МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет України

“Київський політехнічний інститут”

Матеріали наукової конференції студентів, магістрантів та аспірантів

«Інформатика та обчислювальна техніка – ІОТ-2016»

25 – 27 квітня 2016 року

(кафедра «Обчислювальної техніки»)

ТЕЗИ

Київ 2016

Оргкомітет конференції:

Голова:

О.А. Павлов – декан факультету, д.т.н., професор.

Заступник голови :

О .М. Долголенко – заст. декана по науковій роботі, к.т.н., доцент

Співголови організаційного комітету:

Г.М. Луцький – завідувач кафедрою ОТ, д.т.н., професор

Члени організаційного комітету:

Ю.О. Кулаков- д.т.н., професор кафедри ОТ

В.І. Жабін – д.т.н., професор кафедри ОТ

В.П. Симоненко - д.т.н., професор кафедри ОТ

С.Г.Стіренко – д.т .н., професор кафедри ОТ

О.П. Марковський – к.т.н., доцент кафедри ОТ

Секретар конференції:

Н.Є.Куц Пров. інженер кафедри ОТ

ЗМІСТ

УДК 004.771

АГЄЄНКО Ю.М.
КУЛАКОВ Ю.О.

СПОСІБ ДИНАМІЧНОЇ МАРШРУТИЗАЦІЇ ПОШУКУ ІНФОРМАЦІЇ В НЕСТРУКТУРОВАНИХ МЕРЕЖАХ P2P

У роботі проведено огляд існуючих способів динамічної маршрутизації пошуку документів в мережах Peer-to-Peer, розглянуто їх можливості, характерні переваги та недоліки, та основну концепцію. Запропоновано модифікацію алгоритму динамічної маршрутизації в неструктурованих децентралізованих мережах P2Pна основі вивчення зовнішнього локального стану мережі. Представлено переваги модифікованого алгоритму, який дозволяє зменшити час відповіді на запит, збільшити подібність документів до теми запиту та кількість успішних запитів.

The article reviewed the existing methodsof dynamic routing document searching in networks Peer-to-Peer, considered their ability, specific advantages and disadvantages, and the basic concept. Propose the modification of the dynamic routing algorithm in unstructured decentralized P2P networks based on the study of the external local network conditions. Presented the benefits of the modified algorithm, which allows reducetime of a reply to the request, increase similarity of documents to request subject and the number of successful requests.

Вступ

У наш час широкого розповсюдження набув варіант архітектури системи, основою якої є мережа рівноправних вузлів, тобто peer-to-peer архітектура. Одноранговi, або піринговi (peer-to-peer, P2P –рівний до рівного) мережі – це концепція інформаційної мережі, в якій її ресурси розосереджені по усіх системах. В одноранговій мережі всі комп'ютери рівноправні, кожний з них може бути як клієнтом, так і сервером, що дозволяє зберігати працездатність мережі при будь-якій кількості й будь-якій комбінації доступних вузлів [1].

Особливості мереж P2P

В джерелах [1] та [2] описано переваги та недоліки мереж типу peer-to-peer, основні з яких можна представити наступним чином:

Переваги:

– легкість в установці і налаштуванні;
незалежність окремих ПК від виділеного сервера;
– користувачі мають можливість контролювати свої власні ресурси;
– мала вартість та легка експлуатація;
– мінімум устаткування і програмного забезпечення;
– відсутність необхідності в адміністраторі мережі.

Недоліки:

– необхідність пам'ятати паролі до всіх розділених ресурсів;
– необхідність проводити резервне копіювання окремо на кожному комп'ютері, щоб захистити всі дані;
– падіння продуктивності при доступі до розділеного ресурсу, на комп'ютері, де цей ресурс розташований;
– відсутність централізованої схеми для пошуку і управління доступом до даних.

Через те, що кожен мережевий комп'ютер може служити сервером, користувачам важко відслідковувати, на якій машині знаходиться необхідна їм інформація, а це означає, що у мережах P2P важко організовувати зберігання та облік даних. З ростом числа вузлів, на яких повинна відбуватися перевірка, пошук ресурсів ускладнюється через децентралізовану природу мережі.

Сьогодні на технології peer-to-peer засновано величезну кількість популярних мережевих сервісів – від простого обміну файлами до мовного та відеозв'язку.За деякими даними, станом на 2012рік в Інтернеті більше половини всього трафіку припадало на трафік файлообмінних P2P-

мереж, а розміри найбільших з них подолали позначку в 1 млн. одночасно працюючих вузлів. Серед файлообмінних мереж за кількістю вузлів лідирують такі мережі: Bittorrent, eDonkey2000та Gnutella-1,2 [3].

Задача маршрутизації в мережах

Маршрутизація є однією з ключових функцій мережевого рівня і являє собою процедуру визначення одного або множини шляхів (маршрутів) передачі інформації, оптимальних у рамках обраних критеріїв[1, 4]. Таким чином, маршрут – це послідовність мережевих вузлів і трактів передачі, які з'єднують задану пару вузлів мережі.

Основні цілі маршрутизації полягають у мінімізації значень обраних показників якості обслуговування (швидкості передачі, середньої затримки, втрат пакетів й ін.), а також у забезпеченні збалансованого завантаження мережі, її канальних і буферних ресурсів. Тому основними завданнями, які належать до галузі маршрутизації, є: контроль і збір інформації про стан мережі (її топології, завантаження мережевих ресурсів), розрахунок маршрутів і реалізація маршрутних рішень.

Розрахунок шляхів і формування маршрутних таблиць на мережевих вузлах відбувається відповідно до реалізованого алгоритму маршрутизації. Усі методи маршрутизації умовно поділяють на *прості* та *складні*. Прості методи маршрутизації не потребують у вузлах мережі маршрутних таблиць та складного програмного забезпечення. До них належать випадкова та лавинна маршрутизації. Складні методи маршрутизації поділяють на *детерміновані* та *адаптивні*. Методи детермінованої (статичної) маршрутизації у проміжних вузлах передбачають використання таблиць маршрутизації, які не змінюються залежно від стану мережі. Такі методи ефективні для малозавантажених мереж, зі збільшенням завантаження їхня ефективність швидко зменшується. Методи адаптивної (динамічної) маршрутизації гнучкіші, тобто маршрутна інформація може змінюватися залежно від завантаженості окремих ланок мережі, виходу їх з ладу тощо. Динамічна маршрутизація ґрунтується на ідеї пристосування алгоритму до реального стану мережі. Слабкою стороною такого підходу є неможливість передбачити стан мережі. До адаптивних методів маршрутизації належать: маршрутизація за досвідом, маршрутизація якнайшвидшого передавання та локально-адаптивна, розподілена, централізована і гібридна маршрутизації.

Актуальність і постановка задачі

Відповідно о способу організації даних, системи P2P поділяють на дві категорії: структуровані та неструктуровані. У неструктурованих позиція вузла в мережі не є попередньо визначеною, в той час як структуровані мають конкретні правила, в яких роз'яснено, де розташовується вузол, приєднаний до мережі. Беручи до уваги, що в неструктурованих системах P2P, піри з'єднані один з одним в довільному порядку, ці мережі пропонують більш гнучке і автономне середовище, оскільки не потребують значного управління для розміщення ресурсів та контролю топології вузлів.

Так як пошук ресурсів вважається слабким місцем в масштабованості неструктурованих P2P мереж, то основною їх проблемою є створення ефективної стратегії маршрутизації для пошуку, тобто визначення місцезнаходження інформації, необхідної користувачу. Більшість існуючих систем підтримує простий пошук об'єкта по ключу чи ідентифікатору. Деякі системи можуть підтримувати більш складні запити, наприклад, за ключовими словами, які знаходять документи, що містять їх. Одні P2P системи спрямовані на пошук єдиного елементу даних, інші – на пошук всіх елементів даних або якомога більшої їх кількості, що задовольняють умові. Бажані властивості алгоритмів пошуку включають високоякісні результати запиту, мінімальні витрати на підтримку маршрутного стану вузла, високу ефективність направлення запитів, балансування навантаження, стійкість до збоїв вузла та підтримку складних запитів.

Враховуючи, що задача пошуку в піринговых мережах зводиться до швидкого та ефективного знаходження найбільш релевантних відповідей на запит, що передається вузлом у всю мережу, а також той факт, що стратегія маршрутизації пошуку в неструктурованій P2P мережі повинна враховувати її динамічні аспекти, можна зробити висновок, що вивчення, аналіз та дослідження

алгоритмів маршрутизації пошуку в динамічних неструктурованих мережах P2P є дуже важливим завданням. Зокрема, актуальними у наш час є задачі покращення алгоритму маршрутизації пошуку, зменшення мережевого трафіку, що генерується під час обробки запиту, і у той же час отримання швидких і якомога якісніших результатів.

Огляд існуючих рішень

На даний момент з метою успішного знаходження ресурсів разом з низькими накладними витратами і затримками, було запропоновано ряд алгоритмів пошуку для ефективного виявлення ресурсів у децентралізованих мережах P2P.

Існують різні класифікації методів пошуку. Одна з класифікацій базується на пересиланні запитів: детермінована та імовірнісна [6]. У детермінованому підході запит, що пересилається, є детермінованим (наприклад, має локальні індекси), інформація про шлях запиту використовується для маршрутизації. При ймовірнісному підході запит направляється або за певними правилами, або у випадковому порядку (маршрут запитів будується через випадково обрані вузли).

Друга класифікація – це методи сліпого та інформованого пошуку [6, 7]. Ця класифікація базується на інформації про місцезнаходження вузлів чи об'єктів: у сліпому пошуку вузли не тримають інформацію про місце розташування об'єкта, а у інформованому пошуку вузли збирають деякі метадані, які допомагають при операціях пошуку.

Способи сліпого пошуку мають просту реалізацію і не використовують додаткових параметрів, але часто завантажують мережеві канали великою кількістю. Вузли не тримають ніякої інформації про P2P мережу або ймовірні місця розташування об'єктів для маршрутизації запитів. До таких методів відносять: пошук в ширину, випадковий пошук в ширину, ітеративне поглиблення, алгоритм випадкового обходу та алгоритм випадкового обходу рівня k.

Коротко розглянемо кожен з них:

Пошук в ширину (Breadth First Search, BFS) [8, 9]. У той час як пошук є дуже простим, кожен запит потребує багато мережевих і обчислювальних ресурсів через те, що запити розповсюджуються по всіх каналах. Хоча цей метод генерує величезну кількість повідомлень, що дублюються, він гарантує високий рівень успіху.

Випадковий пошук в ширину (RandomBFS, RBFS) [9]. Вузол-запитувач передає запит довільно обраному набору сусідів. При отриманні повідомлення запиту, кожен сусід перенаправляє запит довільно обраному набору своїх сусідів, але виключаючи вузол від якого було отримано повідомлення. Цей підхід має досить високий показник успішності.

Ітеративне поглиблення [10] представляє собою пошук в глибину до фіксованої глибини в дереві. Запит завершується, коли буде отриманий задовільний результат або досягнута максимальна межа глибини (в останньому випадку вимоги запиту не будуть задоволені). Основними недоліками методу є створення великої кількості однакових повідомлень і повільна обробка запитів.

У стандартному *алгоритмі випадкового обходу*[8] вузол запитувач передає повідомлення запиту одному, випадково обраному, сусідові, який в довільному порядку вибирає одного зі своїх сусідів і передає йому повідомлення запиту (ця процедура триває, поки дані не будуть знайдені). Це може значно зменшити кількість повідомлень, що генеруються в мережі, але призводить до більш тривалої затримки пошуку.

В *алгоритмі випадкового обходу рівня k* [11, 12]вузлом запитувачем генеруєтьсяk повідомлень. Тобто вузол запитувач відправляє k копій повідомлення запиту до k довільно обраних сусідів. Кожне повідомлення запиту здійснює свій власний випадковий обхід. Цей алгоритм намагається зменшити затримку маршрутизації, яка, як і очікується, буде в k разів менше.

Найбільш цікавими методами пошуку в неструктурованих мережах є *способи інформованого пошуку*. У інформованих пошукових механізмах вузли зберігають деяку інформацію про маршрутизацію для пересилки запитів до підходящих вузлів. Ця інформація базується на декількох параметрах, таких як популярність об'єктів, ймовірність успішного знаходження інформації тощо. Інформовані підходи мають складнішу реалізаціюі вимагають використання

додаткових параметрів вузла, але є більш ефективними. До таких методів відносять: інтелектуальний пошук, адаптивно-ймовірнісний пошук, пошук на основі індексів маршрутизації та алгоритм експертних груп з використанням індексів маршрутизації.

Інтелектуальний пошук (Intelligent Search Mechanism, ISM) [8, 9, 13]. Кожен вузол в мережі має дані про своїх сусідів, і використовує їх, щоб знайти потрібний вузол, який найуспішніше зможе відповісти на запит (запити пересилаються тільки до цих вузлів). Продуктивність ISM покращується з часом, так як вузли дізнаються все більше інформації про своїх сусідів.

Адаптивно-ймовірнісний пошук (AdaptiveProbabilisticSearch, APS) [7] використовує кількісні дані (у вигляді ймовірнісної інформації) з метою керування пошуковою операцією. Кожен вузол має таблицю з ймовірнісним коефіцієнтом пересилання до кожного сусіда і для кожного ресурсу. Значення кожного елемента таблиці відображає відносну ймовірність сусіднього вузла бути обраним в якості наступного кроку для конкретного ресурсу. Значення індексу оновлюється після кожного запиту.

Пошук на основі індексів маршрутизації[14]. Індекси маршрутизації містять інформацію про теми документів і їх кількість, що зберігають сусіди. Алгоритм з використанням індексів маршрутизації розглядає зміст запитів, ґрунтуючись на змісті документів, а не на імені або ідентифікаторі файлу. Мета індексу маршрутизації (RI) полягає в тому, щоб спростити вузлу процес вибору «кращих» сусідів, для передачі запиту. В цілому, хороший сусід – той, через якого можуть бути швидко знайдені багато документів.

Пошук на основі індексів маршрутизації є модифікацією алгоритму пошуку в глибину (DepthFirstSearch, DFS), всі інші – модифікації BFS пошуку; алгоритм ітеративного поглиблення належить до детермінованих методів пошуку, всі інші – імовірнісні.

Інший підхід класифікації схем пошуку в неструктурованих системах P2P розділяє їх на алгоритми з відправленням запиту всім сусідам і тільки підмножині сусідів. Кращими є алгоритми з використанням підмножини сусідів, адже запит в даному підході відправляється тільки підмножині домінуючих вузлів, пріоритет яких розраховується в різних алгоритмах за своїми критеріями. В ітеративному поглибленні та в класичному методі пошуку в ширину запит відправляється всім сусідам вузла, у всіх інших схемах – лише підмножині сусідів.

Опис модифікованого алгоритму

Враховуючи, що ключову роль при пошуку ресурсів відіграє час затримки та якість результатів, можна зробити висновок, що для вирішення поставленої задачі більше підходять алгоритми модифікованого BFS пошуку. Найкращі результати серед них показує алгоритм експертних груп з використанням індексів маршрутизації, який відноситься до ймовірнісних алгоритмів з відправленням запиту тільки підмножині сусідів.

*Алгоритм експертних груп з використанням індексів маршрутизації*представлено у роботі [15]. Індекс маршрутизації зберігає значення подібності теми поточного вузла до однієї з тем сусіднього вузла і якщо це значення не менше за необхідний поріг подібності тем, то вважається, що цей вузол є “хорошим” сусідом для перенаправлення йому запиту. Варто звернути увагу, що індекс маршрутизації спочатку порожній і заповнюється при додаванні або видаленні документів з вузла. Коли пір приєднується до мережі, він вибирає випадкових сусідів для з'єднання. Після цього він повинен поширити свої знання, щоб отримати знання від інших пірів, широкомовно передаючи свої характеристичні вектори.

Проте цей алгоритм також має недоліки, а саме: неефективне використання інформації в індексі маршрутизації. З метою підвищення швидкості знаходження результату та зменшення кількості повідомлень в мережі було прийнято рішення модифікувати використання та структуру індексу маршрутизації в даному підході.Для забезпечення вищої точності вибору потенційних сусідів було вдосконалено формулуобчислення коефіцієнта подібності тем, яка виглядає наступним чином:

$$P_{sim}(l,q) = \left(\bar{l}\cdot\bar{q}\right) \Big/ \left(|\bar{l}|\cdot|\bar{q}|\right),$$

де l – вектор, що відображає тему вузла, q – вектор відображення теми запиту.

Припустимо, що вектор L={A, B, C, D} є набором ключових слів теми вузла, а вектор Q={B, C, D, E} – набором ключових слів теми запиту, тоді вектор R={A, B, C, D, E} – набір об'єднання всіх ключових слів тем, подобу яких вираховується і l={11110}, q={01111}.Тоді значення коефіцієнта подібності становитиме:

$$P_{sim}(l,q) = \frac{1\cdot 0+1\cdot 1+1\cdot 1+1\cdot 1+0\cdot 1}{\sqrt{1+1+1+1+0}\cdot\sqrt{0+1+1+1+1}} = 0,75.$$

Висновки

У роботі було розглянуто найпоширеніші протоколи динамічної маршрутизації в мережах P2P. Найефективнішим серед розглянутих є алгоритм експертних груп з використанням індексів маршрутизації. Для усунення такого його недоліку, як неефективне використання інформації в індексі маршрутизації, було запропоновано модифікацію структури індексу маршрутизації і алгоритму його підтримки та обробки, а також вдосконалено формулу обчислення коефіцієнта подібності теми вузла і теми запиту, що в свою чергу дозволяє зменшити час відповіді, збільшує ефективність та результативність пошуку при незначному збільшенні повідомлень в мережі. Це показує особливу доцільність практичного використання даного алгоритму в мережах, де вузли виконують пошук інформації, подібної до власного контенту.

Перелік посилань

1. Кулаков Ю.О. Комп'ютерні мережі: Підручник за редакцією Ю.С. Ковтанюка / Кулаков Ю.О., Луцький Г.М. – Київ.: Видавництво «Юніор», 2005. – 397с., іл.
2. Преимущества и недостатки одноранговых сетей – Компьютерные сети[Електронний ресурс]. – Режим доступу: http://mydirectx.ru/seti/preimushchestva_i_nedostatki_odnora.htm
3. Файлообмінна мережа – Вікіпедія. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Файлообмінна_мережа
4. Маршрутизація: мета, основні задачі й протоколи – Електронні засоби навчання. [Електронний ресурс]. – Режим доступу: http://www.znanius.com/3820.html
5. Буров Є. Комп'ютерні мережі. 2-ге оновл. і доповн. вид. Львів: БаК, 2003. – 584 с., іл.
6. Li, X. Searchingtechniques in peer-to-peer networks. Handbook of Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks //Li, X., H. Garcia-Molina // CRC Press, Boca Raton, FL, pp. 120-125, 2010.
7. D. Tsoumakos. Adaptive probabilistic search in peer-to-peer networks. //D. Tsoumakos, and N. Roussopoulos//Proc. of the 3rd Int. Conf. P2P Computing, pp. 102 – 109, 2012.
8. George H. L. Unstructured Peer-to-Peer Networks: Topological Properties and Search Performance //George H. L., Fletcher, Hardik A. Sheth and Katy Borner //Lecture Notes in Computer Science - Agents and Peer-to- Peer Computing, Springer Berlin / Heidelberg, pp. 14-27, 2009.
9. V. Kalogeraki. A local search mechanism for peer-to-peer networks //V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yazti // Proc. of the Eleventh International Conference on Information and knowledge Management, pp. 300 - 307, 2012.
10. B. Yang. Improving search in peer-to-peer networks. //B. Yang and H. Garcia-Molina // In 22nd International Conference on Distributed Computing Systems (ICDCS), July 2012.
11. Z. Zhuang. Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks //Z. Zhuang, Y. Liu, L. Xiao, L. M. Ni // Proc. of International Conference on Parallel Processing, pp.171 - 178, 2013.
12. D. S. Milojicic. Peer-to-peer computing.//D. S. Milojicic, V. Kalogeraki, R. Lukose, and others//HP Lab technical report, HPL-2012-57, 2012.
13. M. Nemade. Intelligent File Search in P2P Network. //M. Nemade and A. Sohaney // Proc. of 4th USENIX Symposium on Internet Technologies and Systems (USITS'13), 2013.
14. K. Hildrum. Distributed object location in a dynamic network. //K. Hildrum, J. Kubiatowicz, S. Rao and B. Y. Zhao//Proc. of 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA), 2012.
15. Achmad Nizar Hidayanto. Adaptive Routing Algorithms in Unstructured Peer-to-Peer (P2P) Systems. //Achmad Nizar Hidayanto and Stephane Bressan // International Journal on Computer Science and Engineering (IJCSE), 2011.

УДК 681.324

В. В. БАЛЬ
В.П. СІМОНЕНКО

ПРОГРАМНІ ІНСТРУМЕНТИ З ВІДКРИТИМ ДОСТУПОМ ДЛЯ ШВИДКОГО РОЗВИТКУ ТИПІЗОВАНИХ GRIDСИСТЕМ

Сервісно-орієнтовані архітектури та програмні додаткинабули широкого поширення в Grid комп'ютерних співтовариствах. Були розроблені ряд інструментів і проміжних систем для підтримки розробки додатків за допомогою GRID архітектури.У цій статті я опишу, програмні додатки з відкритим вихідним кодом, інструменти розширювання, що забезпечують легку розробку та розгортання Web Services ResourceFramework (WSRF).

Ключові слова: *GRID, Grid обчислення, веб сервіси, Grid сервіси, Globus. WSDL*

ВСТУП

Grid обчислення швидко стають новим засобомдля створення розподілених інфраструктур і віртуальних організацій для мульти-інституційних досліджень і корпоративні програм. Grid сисеми перетворилася збудучи платформо орієнтованих на великомасштабних обчислювальних додатків на нову архітектуру парадигми для обміну інформації, даних та програмним забезпеченням, а також обчислювальних ресурсів і ресурсів зберігання даних.

Величезний масив проміжних систем та інструментальних засобів були розроблені для підтримки додатків, в Grid системах. Вони включають в себе проміжне програмне забезпечення та інструменти для розгортання послуг та дистанційного обслуговування виклику [1], безпеки [2], моніторингу ресурсів та планування [3], високої швидкості передачі даних [4], метаданих і управління [5], поєднання компонентів на основі програмного додатка [6], і управління робочим процесом [7].Раніше зусилля були зосереджені на забезпеченні основної функції, необхідних для додатків, щоб використовувати Gridтехнології.Як зросло число ресурсів і додатків які підтримують Grid системи, сумісність стала головною турботою, так як додатки і ресурси неоднорідні, часто погіршувалася автономність, і програмний доступ до ресурсів системи. Найбільш нещодавні зусилля, Web Services Resource Framework (WSRF) стандартизації , відкрили шлях для більш тісної взаємодії та об'єднання між збереженням стану Grid-сервісів і веб-служб.Найбільш широко використовується посилання реалізації OGSA, а реалізується OGSI (Open Grid Services Infrastructure), і WSRF на основі Globus Toolkit (GT) версії 3.2 (GT 3.2) і версія 4.0 (GT 4.0), відповідно.

Основна частина

GT дозволяє Grid архітекторам створюватиGrid з використанням стандартних будівельних блоків. Він також реалізує підтримку для розгортання і виклику Grid-сервісів, а також забезпечує підтримку часу виконання для спільних послуг, таких як індекс сервіс для реєстрації та відкриття послуг та інфраструктури Globus безпеки (GSI) для забезпечення безпеки.Ці компоненти дозволяють постачальникам послуг, розгортатиGrid-сервіси, рекламувати їх, захищати їх від взлому. OGSA, WSRF, і GT забезпечують основу, на якій можуть бути розроблені конкретні безпечні послуги, інструменти і додатки. Крім того, ці майбутні мерееві технології включили багатий набір функцій для забезпечення більшої кількості послуг для клієнтів, і, отже, збільшенню можливості одержання доходу для мережевих провайдерів. Підвищена гнучкість послуг на мережевому рівні також відкриває двері до сторонніх послуг побудованих на вершині інфраструктури. Важливо розрізняти GRID обчислення та Cloud обчислення. GRID обчислення має більшу історію і перш за все був прийнятий публікою обчислювального сектору прогресивної групи користувачів. Ця спільнота має нагальну необхідність в великомасштабних обчислювальних системах та розробка стандартів сприяла цьому. Хмарні обчислення навпаки, виникли в приватному секторі, де технології віртуалізації і великих центрів обробки даних були перетворені в фундамент для продуктів і послуг, які будуть перепродані.Тим не менш, додаткові

інструменти, які зроблять його легше використовувати Grid-технологій в обслуговуванні та розвитку і розгортання додатків, необхідних для більш широкого застосування грід в прикладних областях.Розробка сервісів з Globus Toolkit потребує гарного знання Grid-технологій , сервісних технологій і детальних знань у виеористанні інструментів. Уявіть розробники приховують від користувачів складнощі в обслуговуванніта управління інструментами низького рівня і процесів для розробки та розгортання послуг.Друга мета полягає в забезпеченні більш високого рівня взаємодії в Grid-середовищі. Для цього, вони реалізують підтримку в розвитку сильно типізованих послуг.Сильно типізованих послуг є ті, які споживають і виробляють типи даних, які добре визначені й опубліковані в обладнанні. Використання опублікованих типів дозволяє розробнику створювати сумісні і сумісні Grid-сервіси, без необхідності спілкуватися з іншими розробниками Grid-сервісу.

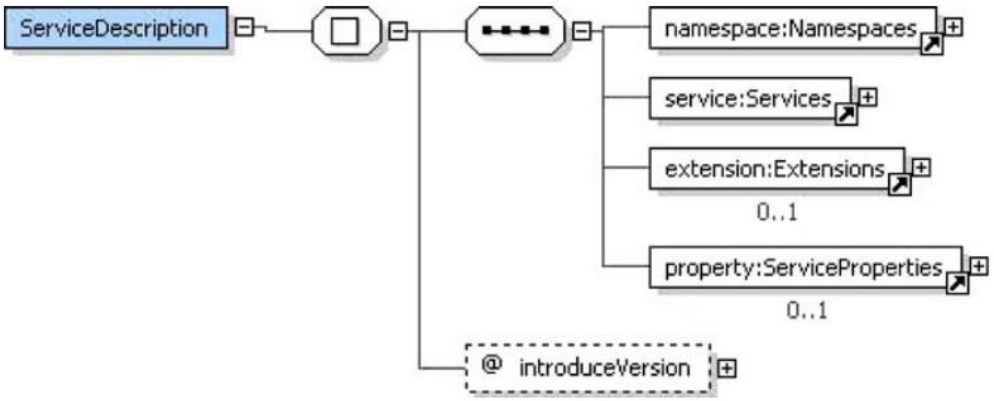


Рис. 1. Базова схема опису надання послуг

Globus забезпечує однорідність доступу до ресурсів. При виклику комунікаційної процедури модуль повинен задіяти одну з наявних функцій. У локальній мережі це звернення по TCP / IP, в рамках паралельного комп'ютера - за внутрішніми високошвидкісних каналах, в глобальній мережі - по ATM.

Для того, щоб визначити, виклик якого методу оптимальний для конкретного випадку, Globus пропонує кілька способів. По-перше, це вибір, заснований на правилах. У кожній точці, де вибір неоднозначний, існує шлях за замовчуванням, закладений розробником (наприклад, «використовувати пакет TCP розміром N») і можливість альтернативного поведінки, яке перевизначається користувачем.

По-друге, користувач може вказати правила, що діють на підставі запиту про властивості ресурсів. Наприклад, «використовувати інтернет-канал, якщо завантаження висока, інакше ATM». Дані про систему надає інформаційна служба Globus. І третій спосіб - це використовувати механізми повідомлення про якість обслуговування. На ліміти якості накладаються певні обмеження; якщо вони порушуються, відбувається зворотний виклик обумовленою функції. Таким чином, наприклад, можна переключатися між мережами, якщо одна з них перевантажена.

Повідомлення обчислювальних вузлів в кластерному системі здійснюється по різних каналах. Необхідно не просто Комунікації в Globus засновані на бібліотеці Nexus, де визначено п'ять головних абстрактних сутностей. Це вузли, потоки, комунікаційні ланки, контексти і видалені запити. Nexus пов'язує початкові і кінцеві пункти обміну повідомленнями, утворюючи комунікаційні ланки. З одним пунктом отримання може бути пов'язано кілька пунктів відправки і навпаки - так утворюється групова передача і збір інформації, в чомусь аналогічна обміну повідомленнями. Комунікаційне ланка підтримує єдину операцію - асинхронне віддалене обслуговування направляється запиту (RSR). У точці створення запиту вказується назва процедури і виділяється буфер під її аргументи. У кожній точці отримання RSR віддалено запускає цю процедуру.

Перманентні дані в кластері фізично, швидше за все, розподілені по декількох комп'ютерів. Для забезпечення прозорого доступу до них у Globus використовується інтерфейс віддаленого

доступу RIO, що базується над абстрактним пристроєм вводу-виводу (ADIO - Abstract Device I / O). У ADIO визначений інтерфейс для відкриття, закриття, читання і запису файлів в паралельному режимі. Звернення програм до локальної файлової системи йде через ADIO, тобто, незалежно від системи, в якій реально зберігаються дані.

У процесі роботи віртуальної організації потрібно інформація про мережу та обчислювальних ресурсах системи в цілому. Це можуть бути такі параметри, як пропускної поріг внутрішніх каналів, початкова завантаження процесорів, кількість обчислювальних вузлів і їх потужність і т.д. Може знадобитися також інформація про додатки - наприклад, про їх ресурсоємності.

Частина цієї інформації використовується глобально, інші частини мають сенс тільки для окремих областей видимості. Її джерела різноманітні - якісь дані надають самі додатки, якісь - стандартні інформаційні служби або спеціалізовані сервіси. Globus об'єднує всі ці різнорідні дані і надає до них однорідний доступ за допомогою сервісу метакомпьютерной директорії (MDS - Metacomputing Directory Service). Інформація організована у вигляді набору структур, кожна з яких є списком пар ключ - значення. У значеннях записуються атрибути, що містять адміністративну та системну інформацію.

На основі базових елементів в Globus конструюються високорівневі механізми. Найбільш значимим прикладом, напевно, є механізми безпеки. Globus забезпечує як глобальну ідентифікацію на основі сертифікату безпеки, так і централізований локальний доступ для ресурсів і користувачів. Ці ж сервіси відповідають за ідентифікацію ресурсів, до яких звернуто віддалений запит.

Іншим важливим прикладом можуть служити адаптації сервісів Globus до інтерфейсів паралельного програмування. Це і відома реалізація інтерфейсу обміну повідомленнями - бібліотека MPICH-G2, і інфраструктура розподілених кластерних обчислень Condor, та інші широко використовувані бібліотеки.

UNICORE (UNiform Interface to COmputing REsources) - однорідний інтерфейс для розподілених обчислень. UNICORE надає інтернет-доступ до вузлів системи, де різниця в апаратних платформах, механізми безпеки і обмін інформацією приховані від кінцевих користувачів.

Архітектура UNICORE тришарова. У неї входять: рівень користувача, UNICORE сервер і цільова система, де буде виконуватися додаток. Компоненти взаємодіють між собою у відкритій мережі через SSL (Secure Socket Layer) протокол. Для встановлення клієнт-серверного з'єднання SSL використовує криптографію з відкритим ключем, так що кожен компонент повинен мати відкритий і секретний ключі. Ключі отримують сертифікат, щоб компонент міг бути ідентифікований як безпечний.Introduce Graphical Development Environment (GDE) є графічним інструментом розробки, який може бути використаний для створення, модифікації та розгортання Grid-технологій.

Висновки

Grid-обчислення стали технологічним вибором, який дозволяє більш ефективно використовувати розподілені дані, аналітичну інформацію, обчислювальння і зберігання ресурсів в скоординованих, мульти-інституційних додатків великого масштабу.

Потреба у висококваліфікованих сумісної інфраструктури стає все більш важливим для вирішення високо гетерогенною і динамічну природу організацій і груп всередині них.СервісніGrid-технології пропонують життєздатну платформу для задоволення цієї потреби та сприяти застосуванню Grid-комп'ютингу в більш широкому діапазоні областей застосування.Строга типізація послуг дозволяє збільшити синтаксичну інтероперабельність послуг і має вирішальне значення для програмного доступу і правильної обробки даних, що повертаються з сервісів.В цілому, концепція роботи з кластером в сполучному ПО зводиться до ряду загальних положень. Звернення до комп'ютера підміняється зверненням до віртуального елементу. Стандартизовані зв'язки між елементами об'єднують їх у загальну структуру. Доступ до ресурсів та управління ними відбувається через абстрактний шар. Як видно з розгляду, всі платформи для створення віртуальних елементів Grid системи на тому чи іншому рівні

реалізують головні завдання розподілених обчислень.Очевидно, що хоча проекти Globus Toolkit, UNICORE і gLite в чому відрізняються по реалізації, всі вони успішно виконують одну й ту ж задачу - надають користувачам повноцінне керування розподіленими ресурсами. Систему Globus можна охарактеризувати як набір інструментів для розробки Grid додатків на основі сервісів. Unicore орієнтований на однорідний доступ до комп'ютерів Таким чином, у користувачів кожної з розглянутих нами віртуальних платформ Grid з'являється потенційний доступ до мереж інших платформ. Grid обчислення стають все більш інтегрованими.

Перелік посилань

1. Foster, I., Kesselman, C.: Globus: a metacomputing infrastructure toolkit. Int. J. High Perform. Comput. Appl.11, 115–128 (1997)
2. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational Grids. In: Proceedings of the 5th ACM Conference on Computer and Communication Security Conference, pp. 83–92. ACM, New York (1998)
3. Wolski, R., Spring, N., Hayes, J.: Network weather service: a distributed resource performance forecasting service for metacomputing. Future Gener. Comput. Syst. 15, 757–768 (1999)
4. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: enabling scalable virtual organizations. Int. J. Supercomput. Appl. 15, 200–222 (2001)
5. Ranganathan, K., Foster, I.: Identifying dynamic replication strategies for high performance data Grids. In: Proceedings of International Workshop on Grid Computing (GRID 2002) Denver, CO, 2002
6. Humphrey, M., Wasson, G.: Architectural foundations of WSRF .NET. JWSR 2, 83–97 (2005)
7. Hastings, S., Langella, S., Oster, S., Saltz, J.: Distributed data management and integration: the Mobius project. Proceedings of the Global Grid Forum 11 (GGF11) Semantic Grid Applications Workshop, Honolulu, HI, pp. 20–38, 2004

УДК 681.324

В. В. БАЛЬ
В.П. СІМОНЕНКО

Grid та Cloudобчислення: можливості для інтеграціїз Next Generation Network

Мережі операторів мобільного зв'язкумайбутнього,зараз стандартизовані і мають назву мережі нового покоління (NextGenerationNetwork). У цій статті, розглядаються можливостіінтегрувати Grid та Cloud обчислення у NGN.Метою NGN є надання більш гнучкою мережевої інфраструктури, яка підтримує не тільки маршрутизацію даних та голосового трафіку, але і сервіси більш високих рівнів і інтерфейси для стороннього удосконалення. Крім того, описується важливість визначення стандартизованих інтерфейсів та їх взаємодії. Запропоновані пропозиції, як методи випробувань, розроблені в Європейському інституті Телекомунікаційних стандартів (ETSI), можуть бути застосовані до підвищення якості стандартів вже представлених.

Ключові слова: Мережі нового покоління, NGN,GRID-системи, Cloud обчислення, Європейському інституті Телекомунікаційних стандартів (ETSI).

Carrier-grade networks of the future are currently being standardized and designed under the umbrella name of Next Generation Network (NGN). The goal of NGN is to provide a more flexible network infrastructure that supports not just data and voice traffic routing, but also higher level services and interfaces for thirdparty enhancements. Within this paper, opportunities to integrate grid and cloud computing strategies and standards into NGN are considered. The importance of standardized interfaces and interoperability testing demanded by carrier-grade networks are discussed. Finally, a proposal how

the testing methods developed at the European Telecommunications Standards Institute (ETSI) can be applied to improve the quality of standards and implementations is presented.

Keywords: Next generationnetwork, NGN, Grid, Cloud, European Telecommunications Standards Institute (ETSI).

Вступ

Мережі операторів мобільного зв'язку є ніщо інше як глобальна комунікаційна інфраструктура, яка підтримує мільйони телефонних дзвінків кожен день, і, що ще більш важливо, масивні глобальні передачі даних, переважно отриманих з Інтернету. Ці мережі експлуатують сотні компаній, вони часто розгорнуті на фізичній інфраструктурі. Ці інфраструктури як правило, не належать оператору, але повинні слідувати різним правилам в кожній країні.Ця глобальна інтегрована система працює з надзвичайно низьким часом відклику і є прозорою для кінцевого користувача.Це було досягнуто через десятиліття розвитку та стандартизації інтерфейсів, процесу який тепер став добре відпрацьований в телекомунікаційній галузі.[1]Остання еволюція глобальних комунікаційних мереж, це мережі нового покоління (NGN), призначені для підтримки конвергентних фіксованих і бездротових мереж, що здійснюють передачу голосу і даних.Крім того, ці майбутні мережі технології включили багатий набір функцій для забезпечення більшої кількості послуг для клієнтів, і, отже, збільшенню можливості одержання доходу для мережевих провайдерів.Підвищена гнучкість послуг на мережевому рівні також відкриває двері до сторонніх послуг побудованих на вершині інфраструктури NGN.[2]Зі збільшенням інтересу до GRID-систем протягом останніх декількох років і останнім часом, Cloud обчислень, головне питання, яке необхідно відповісти, як ці технології, концепції і можливості можуть бути включені в NGN.Взаємодія є однією з ключових характеристик в широкому комерційному успіху технології, що використовуються в телекомунікаційному секторі, у зв'язку з взаємопов'язанимихарактеристиками мереж, і безліччю мережевих операторів.

Різниця між GRID та Cloud обчисленнями

Важливо розрізняти GRID обчислення та Cloud обчислення. GRID обчислення має більшу історію і перш за все був прийнятий публікою обчислювального секторупрогресивної групи користувачів.Ця спільнота має нагальну необхідність в великомасштабних обчислювальних системах та розробка стандартів сприяла цьому.Хмарні обчислення навпаки, виникли в приватному секторі, де технології віртуалізації і великих центрів обробки даних були перетворені в фундамент для продуктів і послуг, які будуть перепродані.GRIDсистемидоповнюють, але не залежать від Cloudсистем.[4]Подібність у тому, що обидва націлені на надання доступу до великих обчислювальнихпотужностей (CPU) або зберігання даних (HDD, SSD).Крім того, хмара використовує віртуалізацію, щоб забезпечити єдиний інтерфейс до динамічної масштабованої системи, з наміром щоб шар віртуалізації приховував фізичні неоднорідність, географічних розподілівта недоліків.Поточні можливостіCloud систем надають тільки пряму підтримку для одного користувача або однієї організації, доступу та поточних моделей, як правило, мають високу вартість, щоб інтегрувати обчислення, дані або передачу даних по мережі за межіCloud системи.[5]Ця модель підходить для середовищ, де обчислювальні ресурси і потреби в даних можуть бути виділені в одному місці і швидкого масштабуються (вгору або вниз) з обчислювальної мережі, і наявність даних важлива.Ефективне управління і використання гетерогенних просторово-розподілених обчислювальних систем цілком залежить від доступності, точності та актуальності інформації щодо її компонентів, їх характеристик, стану ресурсів, а також політики їх використання.

Архітектура мереж нового покоління (NGN)

NGNє глобальною ініціативою щодо телекомунікаційної галузі з використанням стандартів, розроблених ETSI і міжнародного союзу стандартизації електрозв'язку (ITU-T).

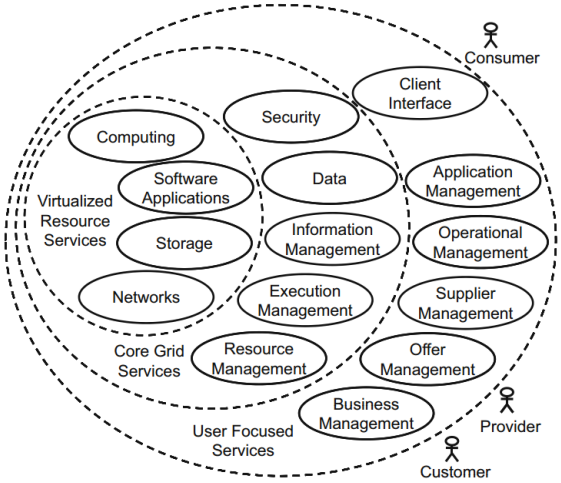


Рис. 1. Концептуальна модель GRIDсистемвід ETSI

Це включає в себе також членів ETSI TC Телекомунікації та Інтернет конвергентних послуг і протоколів для розширеної мережі (TISPAN), які включають операторів зв'язку, таких як BritishTelecom, GermanTelecom та FranceTelecom.Як з іншими телекомунікаційними технологіями, таких як ISDN або GSM, NGN стандарт має намір домогтися сумісності з ними. NGNрозробляється для забезпечення взаємодії збагатьма мережевими стандартами, з поліпшеними мультимедійними можливостями. Існують глобальні зусилля в інформаційній галузі до переходу на NGN, заново розробити стандарти враховуючі існуючі послуги, для того щоб використовувати всі можливостіNGN. Архітектурно, сервісні функції, незалежні від використовуваних технологій. Таким чином, NGN пропонує необмежений доступ користувачів до різних постачальників послуг.Функціональна архітектура NGN що розробила TISPAN [6] виділяє два NGN рівні.Це Сервісний рівень та на IP основі - Транспортний рівень.[7]Обидва рівні складаються з підсистем, які вказаніяк набір функціональних об'єктів та пов'язаних з ними інтерфейсів.

Висновки

Телекомунікаційні оператори очікують, що GRIDсистемиполіпшать їх внутрішню роботу мережі,а також збагатити послуги, які вони пропонують, для своїх клієнтів.Для цього, має бути досягнута взаємодія між GRIDтехнологіями та телекомунікаційними мережами. ETSI і його GRIDмає ключову роль в створенні механізмів пріоритети, стандарти і тестування.Кілька можливих сценаріїв поєднанняGRIDтехнологій і Cloud систем з NGN, кожен має свої переваги і недоліки.Cloud обчислення останнім часом стають популярними, але в даний час відсутні стандарти або перспективи для взаємодії з іншими технологіями, хоча є ознаки розвитку але вони дуже повільні.

Взаємодія між системами може бути досягнута тільки тоді, коли є чіткі стандарти для інтерфейсів і навколишнього середовища, яка підтримує декілька реалізацій архітектурних компонентів. Відсутність широко узгоджених GRID архітектури, що охоплює багато програмного забезпечення, апаратних засобів і послуг, гальмує розвиток узгодженого набору стандартів.Телекомунікаційні галузі отримають цінний досвід роботи з послугами сторонніх систем і підсистем, які пропонують розширену функціональність з NGN. Я очікую, що це розгортання приведе до активізації зусиль з розробки взаємодіючих GRID систем, Cloud систем і телекомунікаційних систем.

Перелік посилань

1.Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement), GFD-R-P.107. Open Grid Forum (2007)

2.Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D.,Ly, A., McGough, S., Pulsipher, D., Savva, A.: JobSubmissionDescriptionLanguage (JSDL) Specification,Version 1.0, GFD-R.136. OpenGridForum (2008)
3.Briscombe, N., Palmer, D., Ntuba, M., Bertram, S.,Boniface, M.: Enablingintegratedemergencymanagement:reapingtheAkogrimobenefits. Tech. rep., IT InnovationCentre, SchoolofElectronicsandComputerScience, UniversityofSouthampton. (Online;http://eprints.ecs.soton.ac.uk/14197/1/Akogrimo_whitePaper_DisasterCrisisMgmt_v1-1.pdf fetched on14-04-09)(2006)
4.ETSI: ETSI ES 282 001: TelecommunicationsandInternetconvergedServicesandProtocolsforAdvancedNetworking (TISPAN); NGN FunctionalArchitecture. EuropeanTelecommunicationsStandardsInstitute (ETSI), Sophia-Antipolis, France(2008)
5.ETSI: ETSI TR 102 659-1 V1.1.1: GRID; StudyofICT Gridinteroperabilitygaps; Part 1: Inventoryof ICT Stakeholders. EuropeanTelecommunicationsStandardsInstitute (ETSI), Sophia-Antipolis, France(2008)
6.Loos, C.: E-healthwithmobilegrids: TheAkogrimoheartmonitoringandemergencyscenario. AkogrimoWhitePaper, (Online; http://www.akogrimo.org/download/White_Papers_and_Publications/Akogrimo_eHealth_white_pap
7.Rings, T., Neukirchen, H., Grabowski, J.: Testing Grid application workflows using TTCN-3. In: InternationalConference on Software Testing Verification and Validation(ICST), pp. 210–219. IEEE Computer Society,Piscataway (2008)

УДК 004.8

БАНІТ В.В.
СТІРЕНКО С. Г.

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ ПРОГНОЗУВАННЯ ФІНАНСОВИХ ЧАСОВИХ РЯДІВ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

В даній статті розглянуто підвищення ефективності прогнозування фінансових часових рядів на основі нейронних мереж на прикладі акцій фондової біржи NYSE. Значну увагу присвячено попередній обробці даних та кодуванню змінних. Розглянуто способи підвищення ефективності прогнозування, додатково врахувавши значення обсягу укладених угод.

In this article consider the increase efficiency of the method of financial time series prediction based on neural networks for example shares Stock Exchange NYSE. Particular attention is endowed preliminary data processing and coding variables. The methods of increased efficiency, additionally taking into account the value of signed contracts.

Ключові слова: *нейронні мережі, машинне навчання, прогнозування фінансових рядів, обробка даних для нейронної мережі, підвищення ефективності прогнозування*

Вступ

В даній статті розглянуто практичнезастосування методу прогнозування часових рядів на основі нейронних мереж. Запропоновано шляхи підвищення ефективності прогнозування.В якості часових рядів буде розглянуто фондовий ринок акцій NYSE.

Гіпотези про можливість прогнозування фондових ринків

Існує декілька теорій про можливість прогнозування фінансових ринків. Гіпотеза ефективного ринку, згідно якої в ціні акції вже врахована вся можлива інформація, і подальше

прогнозування не має сенсу. Продовженням цієї гіпотези виступає теорія випадкових блукань. Інформація може бути передбачуваною, несподіваною, відомою. Передбачена і відома інформація закладена в ринковій ціні, несподівана – являється випадковою, і в ринковій ціні вона не присутня. Таким чином гіпотеза ефективного ринку разом із теорією випадкових блукань стверджують, що зміна ціни на акцію відбувається лише за рахунок несподіваної інформації.[1]

Друга гіпотеза стверджує те, що ринки є неефективними і в ціні акції закладена вся необхідна інформація для прогнозування подальшого курсу. Одним із таких дослідників даного твердження і водночас засновником технічного аналізу був Елліот, який в 30-х роках намагався виявити приховані емпіричні закономірності в своїх публікаціях. В 80-х роках ця точку зору отримала підтримку завдяки теорії динамічного хаосу. За цією теорією протиставляються хаотичність і стохастичність. Хаотичні часові ряди лише виглядають випадковим, незважаючи на те, що представляються в виді детермінованого динамічного процесу. Виходячи з цього вони цілком допускають короткострокові прогнози. Але область прогнозу обмежена в часі горизонтом прогнозування, проте цього може бути вдосталь для отримання прибутку від прогнозу. Звідси випливає те, що перевагу має той, хто має найкращі математичні методи для отримання закономірностей і зашумлених хаотичних рядів. [2]

Постановка задачі прогнозування

З усіх задач для прогнозування курсу акцій найбільш важливими є:

- задача прогнозу майбутньої ціни;
- задача прогнозу напряму руху тренду;
- задача вироблення торгових сигналів на купівлю/продажу;

Дані про поведінку поточної ціни представлені, як вибірка цін за певний проміжок часу з встановленим кроком дискретизації (рік, місяць, день). Для моменту часу (1):

$$t = 1,2, \dots n. \quad (1)$$

дані набувають вигляду часового ряду (2):

$x(t_1), x(t_2) \dots x(t_n)$. (2)

Виходячи з теорії динамічного хаосу, значення ряду до моменту n дозволяють давати оцінку наступним значенням (3):

$$x(t_n + 1), x(t_n + 2) \dots x(t_n + m). \quad (3)$$

Задача прогнозування формалізується через задачу розпізнавання образів.[3]Дані про прогнозуючу змінну за деякий проміжок часу утворюють образ, клас якого визначається значенням прогнозуючої змінної в деякий момент часу за границями даного проміжку часу. Для прогнозування наступного значення часового ряду використаємо метод “часових вікон”. Метод вікон використовує 2 часових вікна T_i, T_o. У кожного задана фіксована ширина n, m відповідно. Перше вікно T_i подається на вхід нейронної мережі, T_o – на вихід. Ці вікна переміщуються по історичній виборці цін, отримуючи необхідні дані для навчання нейронної мережі. Задається фіксований крок переміщення. За кожний крок утворюється пара, яка використовується як елемент навчальної вибірки для мережі.[4]

Нехай часовий ряд $P(t)$ задається значеннями ціни в дискретний момент часу t:

$$P(t_1), P(t_2) \dots P(t_n) \quad (4)$$

Задамо ширину n, m вікон T_i, T_o і крок переміщення s. Вхідні і вихідні вікна, починаючи з першого елементу накладаються на дані часового ряду. Від значень n, m і s залежить якість навчання мережі.

Метод прогнозування на основі нейронних мереж

В якості вхідних даних взято історичні значення ціни акції NYSE Composite. Часовий ряд доволі сильно забруднений випадковими даними, які не несуть закономірностей, сюди можна віднести: короткочасні стрибки цін, що мають випадкову природу, фоновий шум коливання ціни та флет (період часу, за який ціна істотно не змінюється).(Див. рис.1) Тому вхідні дані потребують попередньої обробки.

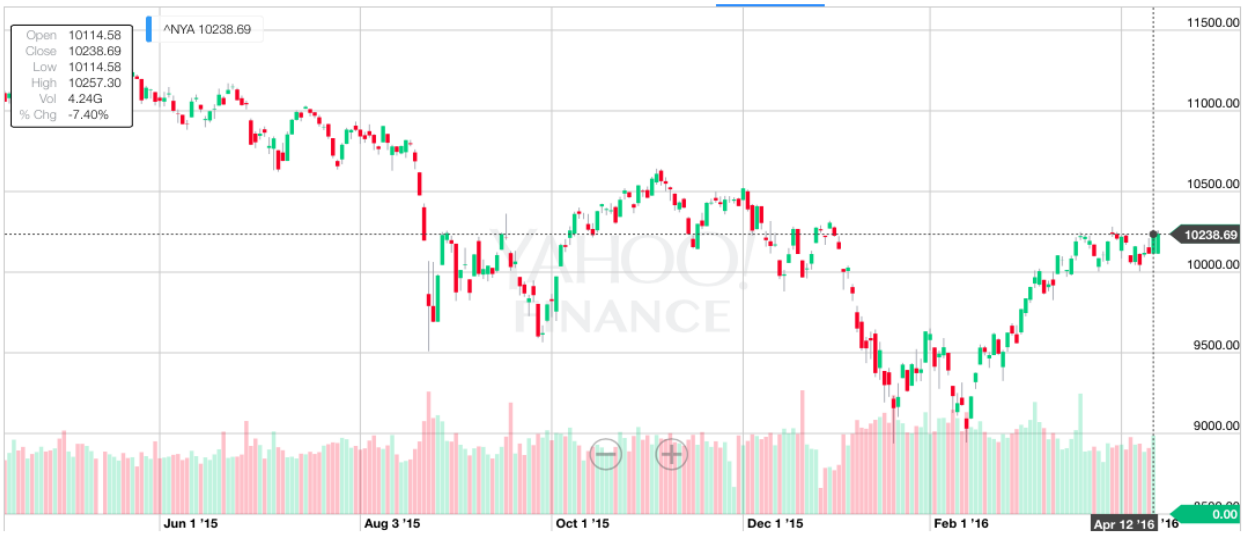


Рис.1 Графік ^NYA у вигляді японських свічок (взято з Yahoo Finance)

Кожна фігура на графіку (рис.3.1) відображає рух ціни за певний проміжок часу.

Якщо ціна за певний період зростає – це бичачий тренд, тіло свічки буде світлим, або прозорим. Якщо ціна падає за певний період – це ведмежий тренд, тіло свічки буде темним, або зафарбованим. Ціна відкриття – це значення ціни на початку проміжку часу. Ціна закриття – це значення ціни в кінці проміжку часу. Максимальна ціна – це максимальне значення ціни на даному проміжку часу. Мінімальна ціна – це мінімальне значення ціни на даному проміжку часу. Індикатор Volume – відображає обсяг укладених угод. Один із найважливіших індикаторів, що показує чи був прорив ціни істинним. На рис.1 його можна побачити у вигляді кольорових стовпців знизу. Якщо при руху ціни різко росте показник Volume – значить рух істинний.[5] На графіку (рис.1) його зростання зображено зеленим стовпцем, а падіння – червоним.

Найсуттєвішими даними для прогнозу являється зміна ціни. Тому потрібно значення ціни перекодувати на коефіцієнти її приросту (5).

$$\boldsymbol{k(t) = x(t)/x(t-1)}. \quad (5)$$

Отримаємо часовий ряд приросту ціни, де

$$\boldsymbol{\max\left(k(t_1), k(t_2), \dots k(t_n)\right)} \quad (6)$$

являється максимальним приростом ціни, а

$$\boldsymbol{\min\left(k(t_1), k(t_2), \dots k(t_n)\right)} \quad (7)$$

являється максимальним від'ємним приростом, тобто падінням ціни.

Завдяки такому перекодуванню максимум та мінімум вибірки являється не історичними максимальними та мінімальними цінами, а історичною зміною ціни. Така вибірка несе більше корисної інформації. Для задач прогнозування навчальна вибірка тим краще, чим менше її суперечливість і більша повторюваність. Отриманий часовий ряд приросту, має малу суперечливість, але й занадто малу повторюваність.[6] Щоб змінити цю ситуацію, перекодуємо значення приросту в класи. Для цього сортуємо вибірку по зростанню (8):

$$\boldsymbol{k_{sort}(t) = \{k(t) | k(t_1) \le k(t_2) \le \cdots k(t_n)\}} (8)$$

Розділяємо на рівні проміжки (9), де p - кількість елементів в одному проміжку, L – довжина вибірки, q – кількість класів, на які розбивається вибірка.

$$\boldsymbol{p = \left[\frac{L}{q}\right], q \in \mathbb{N}}. \quad (9)$$

Тодікласи будуть формуватись так (10):

$$\boldsymbol{Cn = \left[k_{sort}(np), k_{sort}\left((n+1)p\right)\right), 0 \le n \le q, n = q-1 \to k_{sort}\left((n+1)p\right) = k_{sort}(L)} \ (10).$$

Подібним чином перекодовуються значення індикатора Volume. Історична інформація про значення індикатора не несе суттєвої інформації, навпаки – вона буде збільшувати суперечність, адже обсяг угод може бути мінімальним, а стрибки цін хибними і короткочасними. Найбільш важливою інформацією є поточне значення індикатору, що визначає реальну кількість угод і дає

змогу говорити про правдивість чи хибність тренду. Тому до нейронної мережі добавляємо додатковий вхід, що буде відповідати перекодованому поточному значенню індикатора.Зменшуючи вікно, збільшується суперечність і збільшується повторюваність. Зменшуючи кількість елементів навчальної вибірки зменшується суперечність і повторюваність. Зменшуючи кількість класів збільшується суперечність і повторюваність. Необхідно знайти баланс при якому буде найбільша повторюваність і найменша суперечність[7].

Практичний приклад

В якості прикладу прогнозу візьмемо історичні дані акції NYSE Composite фондового ринку NYSE. Дляреалізації візьмемо платформу Node.js, для неї написано безліч бібліотек нейронних мереж. Історичні дані візьмемо підключивши API Yahoo Finance. Бібліотеку нейронної мережі використаємо Brain. Архітектура мережі – багатошаровий персептрон, з одним прихованим прошарком.

Створимо колекцію нейроних мереж. Для кожної нейронної мережі задається довжина навчальної вибірки, кількість класів на які діляться процентні прирости, та глибина вікна. В якості даних візьмемо часовий інтервал за 9 місяців від 01.04.2016 по 01.07.2015 значень акцій ^NYA та значень обсягу укладених угод (Volume). Отриману вибірку розділимо на навчальну і на тестову, навчальну візьмемо за 6 місяців 01.01.2016 – 01.07.2015, а тестову за 3 місяці 01.01.2016 – 01.04.2016.Довжина навчальної вибірки може коливатись для пошуку оптимального значення. Кількість входів прихованого прошарку також змінюється. Тестування показало, що найліпший результат показують нейроні мережі, входи прихованого прошарку якого дорівнюють 18. Без застосування індикаторуVolume, максимальний процент правильного прогнозу на тестовій виборці складав не більше 57 відсотків. Тому подальші тестування відбувалися із введенням в якості додаткової зміни значення індикатору Volume на вхід нейронної мережі. Кількість класів на які розбиваються прирости – 4, а довжина вікна – 5. В таблиці 1 показано 4нейронних мереж, які показали найкращі результати.

Табл.1 Результати тестування нейронних мереж

№	Класи, к-сть класів	Довжина вікна, к-сть одиниць	Прихований прошарок, к-сть входів	Довжина вибірки, к-сть одиниць	Результати на навчальній виборці, %	Результати на тестовій виборці, %
1	5	5	18	180	78,3	57,4
2	4	5	18	180	89,1	65,4
3	3	5	18	180	78	56,7
4	4	4	19	200	79	53,6

Виявлена закономірність, чим ближче тестова вибірка в часі розташована до навчальної - тим вище показники. Це показано в таблиці 2.

Табл.2 Результати тестування

Тестова вибірка, t	Результат, %
01.01.2016 – 01.04.2016	65,4
01.01.2016 – 01.03.2016	68,3
01.01.2016 – 01.02.2016	69,1
01.01.2016 – 10.01.2016	70,2
01.02.2016 – 01.03.2016	64,4
01.03.2016 – 01.04.2016	60,7

Все це свідчить про те, що закономірності змінюються з часом, і нейронна мережа через деякий час потребує перенавчання на нових даних. Введення значення індикатора Volume, в якості додаткового входу в нейрону мережу підвищує якість прогнозу.

Перелік посилань

1.Постулатытехническогоанализа: Техническийанализ [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – “Bull and Bear”поь, 2003-2016. – Режим доступу: http://www.bull-n-bear.ru/technic/?t_analysis=postulates(дата звернення 2.04.2016) – Назва з екрана.

2.НациональныйоткрытыйУниверситет «ИНТУИТ» [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – НОУ “ИНТУИТ”, 2003 - 2016. – Режим доступу: http://www.intuit.ru/department/expert/neurocomputing/8/1.html(дата звернення 3.04.2016) – Назва з екрана.

3.ПРОГНОЗИРОВАНИЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – EreminYakov, 2015. – Режим доступу: http://elanina.narod.ru/lanina/ind/neiro/2.htm (дата звернення 3.04.2016) – Назва з екрана.

4. Калацкая Л. В., Новиков В. А., Садов В. С. Организация и обучение искусственных нейронных сетей: Экспериментальное учеб. пособие. — Минск: Изд-во БГУ, 2003. — 72 с.

5.Индикатор Volumes [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – 2011. – Режим доступу: http://progi-forex.ru/indicator%20volumes.html (дата звернення 3.04.2016) – Назва з екрана.

6. Осовский Станислав. Нейронные сети для обработки информации = Siecineuronowe do przetwarzaniainformacji / Перевод И. Д. Рудинского. — М.: Финансы и статистика, 2004. — 344 с

7.Geektimes[Електронний ресурс] : [Веб-сайт]. – Електронні дані. – 2014-2016. – Режим доступу: https://geektimes.ru/post/144405/ (дата звернення 3.04.2016) – Назва з екрана.

УДК 004.056.53

БІЛАШЕВСЬКА А.В.,
МАРКОВСЬКИЙ О.П

МЕТОД ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ СЕРЕДНЬОАРИФМЕТИЧНОЇ ФІЛЬТРАЦІЇ У GRID-СИСТЕМАХ

Пропонується метод захищеної реалізації середньоарифметичної фільтрації на віддалених процесорних засобах GRID-систем. Метод забезпечу’ захист від доступу до зображень при їх передачі та під час їх обробки на віддалених комп’ютерних системах. Метод базується на використанні адитивного маскування точок зображення. Детально описані процедури шифрування та дешифрування після фільтрації. Та наведено відповідні приклади.

This paper proposes a method for protected implementation of image digital averaging filtering at remote processors of GRID-systems. Proposed methods ensures protection of image against unauthorized access while image network transmission and while image processing on remote computer systems. The methods for averaging filtering is based on using of additional masking. The proposed procedure for image encryption and decryption are described in details, also example for procedure encryption and decryption of filtering are given.

Вступ

Розвиток сучасних технологій обумовлює збільшення можливостей користувачів у просторі вирішення прикладних задач. Основним рушієм такого прогресу є можливість використання обчислювальних ресурсів великої потужності у вигляді віддаленої обробки задач із залученням тисячі багатопроцесорних комп’ютерних систем.

Технології віддаленого надання розподілених обчислювальних ресурсів активно розвиваються в рамках GRID-систем. Динамічний розподіл системою потоку задач користувачів не дозволяє прослідкувати, на якій саме комп’ютерній системі буде відбуватись обробка задачі, тому відкритість GRID-систем спричиняє збільшення можливості реалізації несанкціонованого доступу.

Обробка зображень є однією з найбільш поширених прикладних задач. Чільне місце серед процедур обробки займає фільтрація зображень з метою усунення імпульсних завад. Велика піксельна розмірність робить процедуру обробки занадто ресурсоємною, що робить доцільним використання потужностей GRID-систем. Враховуючи вищевизначену відкритість, виникає потреба у захищеній реалізації фільтраціїї у GRID-системах.

Аналіз методу середньоарифметичної фільтрації

Найбільшого поширення набула середньоарифметична фільтрація. Будь-яка фільтрація виконується над зображенням, яке можна подати в наступному вигляді: $I = \{f(x,y);\ x \in [1;l],\ y \in [1;h]\}$, де I– зображення; $f(x,y)$– функція значення інтенсивності кольору в точці з координатами(x,y); l та h– відповідно ширина та висота зображення. Особливість середньоарифметичної фільтрації полягає у скануванні зображення квадратною апертурою з непарною кількістю рядків та стовбців. Під час сканування знаходиться середнє значення інтенсивності точок апертури, яке замінює значення інтенсивності точки. Обробка розпочинається з лівого верхнього кута, далі апертура переміщується по рядку, а в кінці рядка переходить на першу точку наступного рядка. Приклад на рисунку 1 ілюструє принцип середньоарифметичної фільтрації. За приклад взято зображення 4x4 із застосуванням апертури 3x3. Зліва показано початкове зображення. Далі зліва направо приведена послідовність зміни зображення в процесі його середньо арифметичної фільтрації. На рисунку виділено апертури, які використовуються в процесі фільтрації.

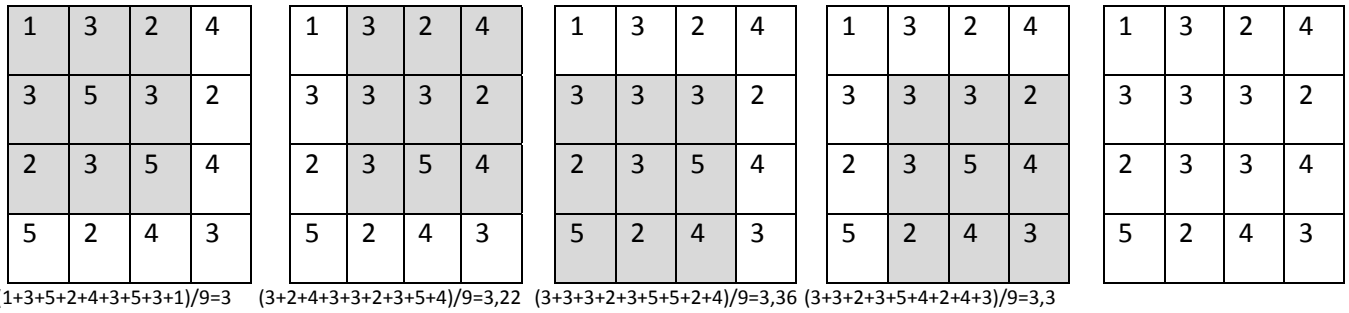


Рис.1 Приклад середньоарифметичної фільтрації

Метод захищеної реалізації середньоарифметичної фільтрації.

В основі середньоарифметичної фільтрації лежить операція обчислення середнього арифметичного точок поточної апертури. Ця операція базується на арифметичному додаванні та діленні. Найпростішим способом шифрування операндів арифметичного додавання є використання адитивного маскування, тобто додавання до кожного із операнду випадкового цілого, кратного певному модулю m. Зняття такої маски може бути доволі просто виконане шляхом знаходження залишку від ділення суми на модуль m, за умови, що модуль m більший за суму.

З урахуванням наведеного, спосіб шифрування, що пропонується зводиться до накладання на точки реального зображення адитивної складової кратної простому числу m, яке за значенням більше будь-якої точки зображення. Якщо позначити значення n реальних точок апертури як $a_1, a_2, a_3, a_4, . a_n$, а через $b_1, b_2, b_3, . b_n$ - відповідно захищені від читання точки цієї апертури, то формально процес шифрування точок апертури можна представити у вигляді: $\forall j \in \{1,.,n\} : b_j = a_j + r_j \cdot m$, де r_j – випадкове ціле число. Для того, щоб результат фільтрації q, що формується як середнє арифметичне значень $b_1, b_2, b_3, . b_n$, міг бути відтворений користувачем, його значення також має бути сумою середнього арифметичного s точок $a_1, a_2, a_3, a_4, . a_n$ та адитивної компоненти кратної m: $q = s + d \cdot m$, де d - ціле число. Таким чином, значення q фактично обчислюється у вигляді:

$$q = \frac{1}{n} \cdot \sum_{j=1}^{n} b_j = \frac{1}{n} \cdot \sum_{j=1}^{n} a_j + \frac{m}{n} \cdot \sum_{j=1}^{n} r_j = s + d \cdot m \quad (1)$$

Відновлення фільтрованої точки s початкового зображення по зашифрованому коду q, як слідує з (1) реалізується через обчислення залишку від ділення q на m: $s = q \bmod m$. Коректність такого

перетворення базується на тому, що $s<m$ (в силу того, що середнє арифметичне менших за m також менше за m), та на тому, що d - ціле.

Так як d має бути цілим, то з (1) слідує, що сума $r_1+r_2+.+r_n$ має бути кратною числу n точок апертури. В процесі фільтрації обчислене значення q замінює центральну $(n+1)/2$-ту точку $b_{(n+1)/2}$ апертури. Так як випадкові коефіцієнти генеруються користувачем до передачі зображення в GRID-систему з урахуванням зазначеної вище умови кратності суми коефіцієнтів апертури її розміру, то для того, щоб при заміні центральної точки апертури середнім значенням q ця умова не порушувалась, коефіцієнт $r_{(n+1)/2}$ центральної. $(n+1)/2$-ї точки має дорівнювати середньому арифметичному коефіцієнтів точок апертури:

$$r_{(n+1)/2} = \frac{1}{n} \cdot \sum_{j=1}^{n} r_j \quad (2)$$

Таким чином, для захищеної реалізації середньоарифметичної фільтрації на віддалених комп'ютерних системах зображення, що містить N точок: $a_1,a_2,.,a_N$, пропонується спосіб, який зводиться до наступної послідовності дій користувача.

1) Генеруються випадковим чином цілі числа $r_1,r_2,.,r_N$, по кількості N точок зображення таким чином, щоб для кожної апертури з n точок їх сума була кратна n і виконувалася умова (2). Вказані дії не прив'язані до конкретного зображення і можуть виконуватися заздалегідь.

2) Для зображення обирається модуль m, що є простим числом, більшим, за значення будь-якої точки зображення: $\forall i \in \{1,2,.,N\}$: $m>a_i$.

3) Для кожної з точок зображення виконується її шифрування накладанням адитивної маски: $\forall i \in \{1,2,.,N\}$: $b_i = a_i + r_i \cdot m$.

4) Зашифровані точки $b_1,b_2,.,b_N$ зображення передаються на GRID-систему для віддаленої фільтрації.

5) В GRID-системі виконується віддалена фільтрація на вільних обчислювальних потужностях і повертається користувачеві в вигляді кодів $s_1,s_2,.,s_N$ точок відфільтрованого зображення.

6) Користувач виконує дешифрацію точок $s_1,s_2,.,s_N$ отриманого зображення шляхом обчислення залишку від ділення: $\forall i \in \{1,2,.,N\}$: $v_i = s_i \bmod m$.

Запропонований метод захищеної реалізації середньоарифметичної фільтрації в GRID-системах може бути ілюстровано прикладом його застосування для фільтрації зображення на рис.1.

Згідно в викладеною вище методикою, випадковим чином формується N=16 цілих коефіцієнтів $r_1,r_2,.,r_{16}$ таких, що їх сума в рамках будь-якої з 4-х апертур розміром 3 х 3 ділиться на 9 і при цьому частка від ділення дорівнює коефіцієнту центрального елементу апертури. Сформовані випадкові коефіцієнти для прикладу наведені в таблиці 1. Легко пересвідчитись, що, наприклад, для правої нижньої апертури, що включає точки 6,7,8, 10,11,12,14,15 і 16, $r_6 + r_7 + r_8 + r_{10} + r_{11} + r_{12} + r_{14} + r_{15} + r_{16} = 5+6+5+8+9+10+5+5+28= 45$; тобто сума ділиться на 9 і коефіцієнт r_{11} центрального елементу апертури дорівнює середньому значенню її коефіцієнтів: $r_{11} = 5 = 45/9$.

Відповідно до п.2 запропонованої методики для заданого зображення випадковим чином обирається модуль m: більше за будь-яке значення просте число, за значенням більше ніж будь-яка точка зображення. Для прикладу, що розглядається m=7. Таблиця 1 відображає шифрування значень точок зображення.

Приклад на рисунку 2 ілюструє принцип фільтрації зашифрованих значень. Зліва показано початкове зображення, що утворюється за допомогою обраного методу шифрування, та поступове обчислення середніх значень. Далі зліва направо приведена послідовність зміни зображення в процесі його середньоарифметичної фільтрації. На малюнку виділено апертури, які використовуються в процесі фільтрації.

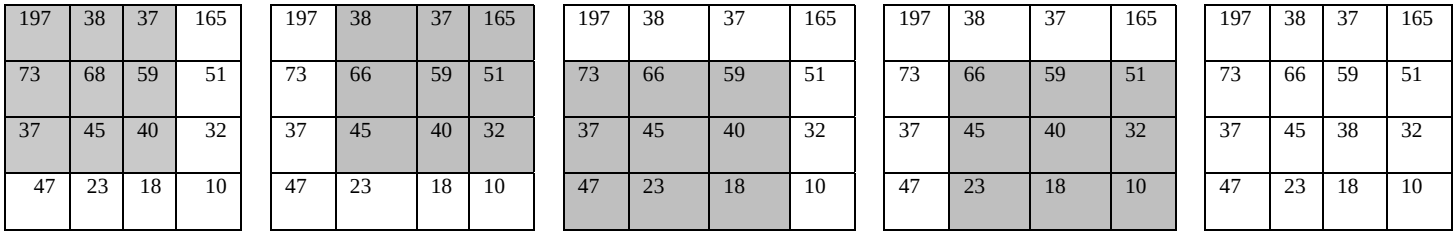
В результаті фільтрації, від системи користувач отримує наступні значення середніх арифметичних : s_1=66 ; s_2=59; s_3=45; s_4=38.

Дешифрування :
$v_1=s_1 \bmod 7=66 \bmod 7=3$;
$v_2=s_2 \bmod 7=59 \bmod 7=3$;

$v_3=s_3 \ mod \ 7=45 \ mod \ 7=3;$
$v_4=s_4 \ mod \ 7=38 \ mod \ 7=3;$
Що абсолютно відповідає вище визначеним реальним значенням.

Табл1. Шифрування точок

Номер точки	a_i	r_i	b_i
1	1	28	197
2	3	5	38
3	2	5	37
4	4	23	165
5	3	10	73
6	5	9	68
7	3	8	59
8	2	7	51
9	2	5	37
10	3	6	45
11	5	5	40
12	4	4	32
13	5	6	47
14	2	3	23
15	4	2	18
16	3	1	10



(197+38+37+73+68+59+37+45+40)/9=66

(38+37+165+66+59+51+45+40+32)/9=59

(73+66+59+37+45+40+47+23+18)/9=45

(66+59+51+45+40+32+23+18+10)/9=38

Рис2. Приклад фільтрації із застосуванням середньоарифметичної фільтрації

Висновки

Основною перевагою запропонованого методу є можливість виконання обробки зображень у GRID-системах з захистом самого зображення від несанкціонованого читання під час передачі та обробки. Основним оцінювальним критерієм методу є рівень захищеності, що досягається за їх застосування. При виконанні середньоарифметичної фільтрації порушення

захисту зображення, що передається для фільтрації у віддалену обчислювальну систему полягає в підборі модуля m, що лежить в інтервалі від k від 2^u. Технологія підбору полягає в тому, що для вибраної точки перебираються всі k можливих її значень. Для кожного з k перебираються можливі значення модуля m з вказаного вище інтервалу. Таким чином, об'єм перебору становить $k\cdot(2^u - k)$. На- приклад, при $k=2^{16}$ і u=32, об'єм вказаного перебору становить $10^{14.4}$, що свідчить про рівень захищеності рівному захищеності відомого алгоритму DES [4].

Перелік посилань

1. Петренко А. Національна GRID інфраструктура для забезпечення наукових досліджень і освіти / А.Петренко Системні дослідження та інформаційні технології.- 2008 - № 4. - С.79-92.
2. Hennessy, John L. Computer architecture: a quantitative approach/ John L Hennessy, David A Patterson. – Amsterdam, Noord-Hooland, Netherlands : Elsevier, 2011. – 824 p.
3. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс; пер. с англ. В.В. Харитонова.- М.:Техносфера, 2005. - 1072 с.
4. Шнаер Б. Прикладная криптография / Б. Шнаер.; пер.с англ. П.В. Семьянов.- М.:Издательство Триумф, 2003.- 816 с.
5. Кнут Д. Искусство программирования для ЭВМ / Д. Кнут.; пер.с англ. Н.И.Вюковой, В.А. Галатенко, А.Б.Ходулева.- М.:Мир, 1978.- 840 с.
6. Брэй Б. Микропроцессоры Intel. Архитектура, программирование и интерфейсы. Шестое издание / Б. Брэй; перю с анг. А.В.Жукова.- Санк-Петербург: БХВ-Петербург, 2005.-1328 с

УДК 004.021

БОЙКО А.П.
КУЛАКОВ Ю.О.

МЕТОД ІНФОРМАЦІЙНОГО ПОШУКУ В ЗАДАЧАХ ОБРОБКИ ПРИРОДНЬОЇ МОВИ

У статті запропонований спосіб виконання глибокого семантичного пошукуз початковим використанням тезауруса.

Ключові слова: обробка природньої мови, інформаційний пошук, семантичний пошук

This article proposes a deep semantic search method with initial use of thesaurus.

Keywords: natural language processing, information retrieval, semantic search

Вступ

Обробка природньої мови – галузь комп'ютерних наук, що займається вивченням та вирішенням проблем роботи комп'ютера з текстами на людській мові. Типовими сферами застосування цієї дисципліни є створення автоматичних перекладачів, синтез мовлення тапобудова інтерактивних діалогових систем, як альтернативи звичному людино-машинному інтерфейсу. Кожну задачу обробки природньої мови можна розглядати в контексті двох процесів – інтерпретації (аналізу) та генерування (синтезу) тексту.

Окремим напрямком досліджень виділяють інформаційний пошук, який об'єднує в собі проблеми обробки природньої мови, а також зберігання та індексування текстової інформації. Завданням інформаційного пошуку є виявлення множини документів або його частин – об'єктів запиту, що відповідають інформаційному запиту – опису потреби користувача в необхідній інформації.

Незважаючи на існування спеціалізованих мов запитів, взаємодія людини з комп'ютерами частіше потребує можливість обробки запиту сформульованого природньою мовою. Такі запити використовуються разом ізметодами пошуку за неформальними ознаками, а саме семантичним та фактографічним.

Семантичний пошук

Семантичний пошук полягає у знаходженні документів, що за змістом відповідають інформаційному запиту. Основною проблемою здійснення семантичного пошуку є неоднозначність слів через полісемію та синонімію, а також складність виявлення та опрацювання стилістичних фігур мови: оксиморонів, метафор, інверсії, порівнянь тощо.

Існуючі методи семантичного пошуку зазвичай є вузько спеціалізованими та працюють з певними обмеженнями. Метод обходу RDF шляхувимагає щоб пошуковий запит був оформлений у вигляді графу із заданими відношеннями між лексемами у вигляді дуг. Метод відображення ключових слів на концепції вимагає складання словника за допомогою машинного навчання, використовуючи тезауруси конкретної мови.Популярні інтерфейси для формулювання пошукового запиту типу SPARQL та GRQL працюють на рівні пошуку шаблонів у моделі документу й не здатні знаходити близькі за змістом фрагменти документу.

Запропонований метод глибокого семантичного пошукуне має вищезазначених недоліків й дозволяє вирішуватизадачі обробки природньої мови кількох різних типів: порівняння текстів, складання стислого змістудокументу (сумаризація), а також пошук частин тексту схожих за тематикою.

Метод глибокого семантичного пошуку

Метод передбачає поетапну обробку частин тексту,які відповідають логічній структурі документу – слів, частин речень, речень, абзаців, розділів (рис. 1).

Етап 1. З'ясування тематики слів речення. Значення конкретного слова може залежати від контексту, в якому воно вжито, тому для отримання можливих початкових значень слова використовується тезаурус.Множина тематик C_iпредставляє собою пари слів чи словосполучень та ймовірностей, з якими ці слова відображають значення речення чи його частини:

$$C_k = \{C_{k\_1}, C_{k\_2}, .., C_{k\_n}\}, C_{k\_i} = (w, p)$$

$$\omega(C_{k\_i}) = w, \qquad (1)$$

$$\rho(C_{k\_i}) = p, p \in [0,1]$$

$$\sum \rho(C_{k\_i}) = 1 \qquad (2)$$

Етап 2. Розбиття речення на частини та виконання синтаксичного аналізу.

Етап 3. Ймовірності тематик слів перераховуються виходячи з виявлених відношень між словами всередині частини речення.

Етап 4. Визначаються можливі тематики всієї частини речення.

Подальший аналіз тексту передбачає ітеративне виконання етапів 2-4, з тією відмінністю, що замість слів аналізуються зв'язки між частинами речення, самими реченнями, абзацами та розділами. В результаті останньої ітерації отримуються тематики усього документу.

Для пошукового запиту попередньо виконуєтьсяаналогічна процедура, результатом якої є множина C_q. Аналіз цільового документу припиняється, якщо на певному рівні пошуку, починаючи з речення, отримані тематики C_k сягнули критичного показника схожостіδ:

$$W_q = \{w(C_{q\_i}), C_{q\_i} \in C_q\}$$

$$P_k = \{p(C_{k\_i}) \mid w(C_{k\_i}) \in W, C_{k\_i} \in C_k\} \qquad (3)$$

$$\forall p \to p \geq \delta, p \in P_k$$

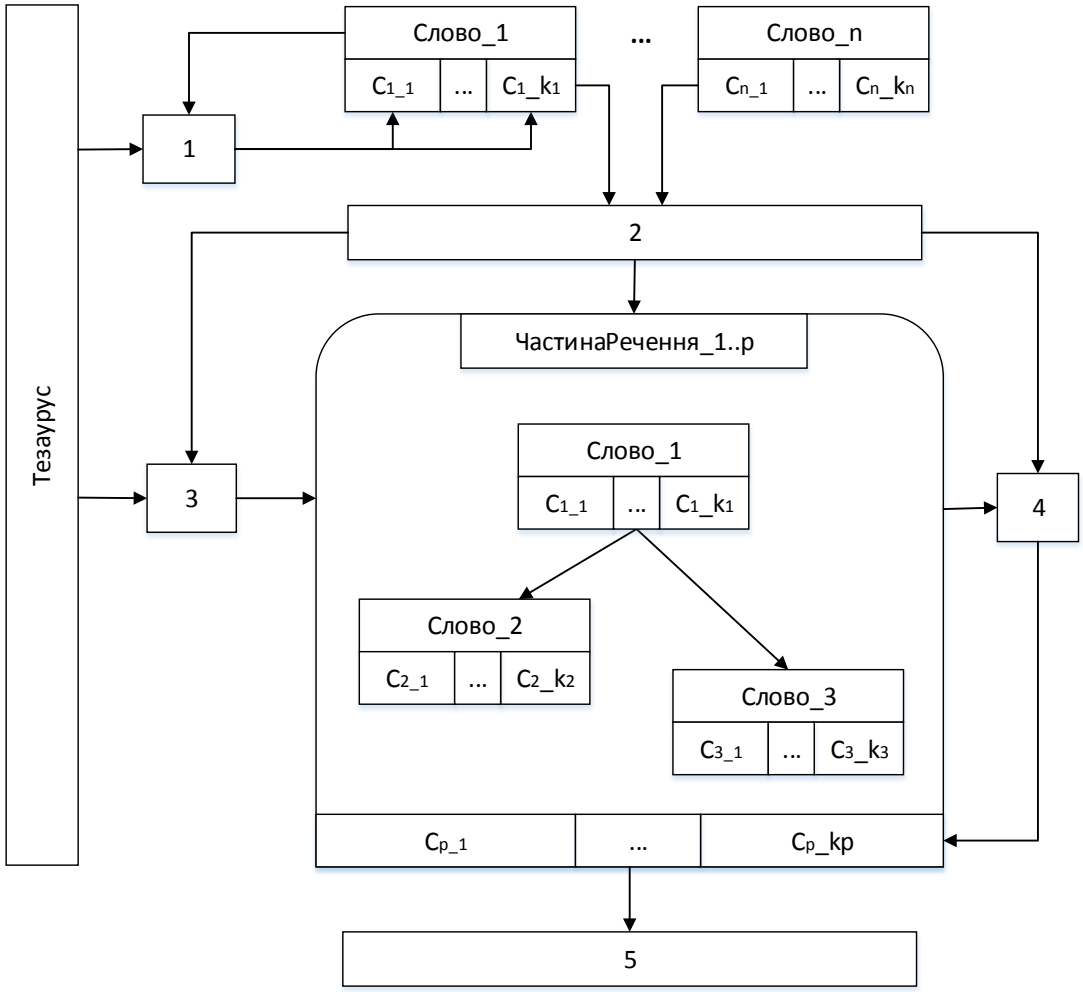


Рис 1. Цикл семантичного аналізу документу

Наприкінці кожної ітерації за необхідності відкидаються тематики, для яких ймовірність відображення правильного сенсу поточної одиниці документу нижча порогового значення θ:

$$C_k = \{C_{k\_i} \mid \rho(C_{k\_i}) < \theta, C_{k\_i} \in C_k\} \quad (4)$$

$$\theta \in [0,1]$$

Після цього ймовірності для отриманих тематикC_k перераховуються для виконання умови (2).

Визначення зв'язків між частинами документу

Робота другого та п'ятого етапів алгоритму передбачає знаходження зв'язків між частинамитексту для їх групування та подальшої роботи вже із сутностями вищого рівня.

Одним із можливих способів виявлення залежностей між словами речення є парсинг на основі даних[2]. Метод використовує поняття зваженої білексичної граматики – скінченного автомату, який при побудові дерева залежностей слів присвоює дугам повного графу ваги на основі бінарних відношень між лексемами.

Для пошуку зв'язків між реченнями та його частинами пропонується використовувати метод додатньо визначених рядкових функцій. Згідно з ним кожна потенційна залежність неявно представляється у вигляду вектору ознак. Кожна ознака відповідає послідовності слів, закріплених за реченнями, між якими був знайдений зв'язок[3].

Слід зазначити, що вищенаведеніметоди не є універсальними, оскільки ефективність їх застосування залежить від синтаксичних особливостей конкретної мови.

Висновок

Запропонований метод передбачається використовувати у типових задачах обробки природньої мови, таких як конспектування документів, спрощення тексту та пошук документів із схожою тематикою. Метод глибокого семантичного пошуку дає такі переваги:

- виявлення кількох змістових значень документу та його частин;
- можливість використання у якості пошукового запитутекстів будь-якого обсягу.

Окрім того в метод закладена можливість реалізації на його основі паралельнихалгоритмів обробки документів.

До недоліків методу слід віднести необхідність генерування тезаурусу для мови документу, звертання до якого даватиме накладні витрати при обробці тексту.Проблема складання тезаурусу вирішується попереднім парсингом тлумачних словників мови[4], а час звертання до словника можна мінімізувати за рахунок використання in-memory баз даних та індексів.

Перелік посилань

1. NitinIndurkhya. Handbook of Natural Language Processing, Second Edition / NitinIndurkhya, Fred J. Damerau. –Chapman & Hall/CRC, 2010. – p. 4.
2. Eisner, J. M. Bilexical grammars and their cubic-time parsing algorithms. // Advances in Probabilistic and Other ParsingTechnologies. – Kluwer, 2000. –pp. 29-62.
3. Anne Kao. Natural Language Processing and Text Mining / Anne Kao, Stephen R. –Springer, 2007. – p. 30.
4. Christopher D. Manning. Introduction to Information Retrieval / Christopher D. Manning, PrabhakarRaghavan, HinrichSchütze. –Cambridge University Press, 2009. – p. 192.

УДК 004.04

Д.В. БУРМІСТРОВ
В.В. ТКАЧЕНКО

ОПТИМІЗАЦІЯ ПОТОКІВ МАРШРУТНОЇ ІНФОРМАЦІЇ В МОБІЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Одним з методів підвищення ефективності в мобільних комп'ютерних мережах є ієрархічний підхід до рішення задачі маршрутизації, який передбачає розподіл мережі на кластери. У статті розглянуті питання впливу структури кластера на ефективність маршрутизації в мобільних комп'ютерних мережах. Досліджена залежність потоку управляючої інформації від кількості вузлів кластера і ступеню їх зв'язності.

Hierarchical approach to routing with fragmentation of the network into the clusters is one of the methods aiming to achieve the efficiency increase of data transfer in wireless network. The article raised an issue of influence of cluster structure to the efficiency of routing in computer wireless network. We have studies the dependence managing date flow from number nodes of cluster and degree of nodes relationship.

Вступ

Мобільні комп'ютерні мережі характеризуються динамічною топологією, обмеженою пропускною здатністю бездротових каналів зв'язку та апаратними обмеженнями мобільних терміналів. Однак сучасні бездротові технології дозволяють все більш широко застосовувати

мобільні мережі в різних галузях для передачі різнорідного трафіка, чутливого до змін затримок і пропускної здатності.

У мобільних комп'ютерних мережах відбуваються часті зміни положення мобільних вузлів, що обумовлює зміни маршрутів передачі даних і різке збільшення об'єму управляючого трафіка, який виникає при оновленні маршрутної інформації. Ефективність передачі даних в мобільних мережах в значній мірі залежить від розмірів мережі, кількості вузлів та швидкості їх пересування. Одним з методів підвищення ефективності передачі даних є ієрархічна структуризація мобільної мережі.

Мета дослідження

Метою дослідження є формування ієрархічної структури мобільної мережі, і дослідження факторів що впливають на вибір розміру і структури кластерів з точки зору оптимізації потоків управляючої інформації.

Постановка задачі

В якості критерію ефективності функціонування мобільної комп'ютерної мережі приймаємо співвідношення об'єму переданих корисних даних (V_n) до загального об'єму переданих мережею даних ($V_n+V_{сл}$), де загальний об'єм переданих даних визначається як сума корисної і управляючої інформації ($V_{сл}$):

$$Э = \frac{V_n}{V_n + V_{сл}} \text{ , де } Э \to 1. \qquad (1)$$

Таким чином, при формуванні структури кластера в мобільній мережі необхідно вирішити наступні задачі:

- оптимізація потоку управляючої інформації, з метою задовольнити критерію (1);
- забезпечення стійкості маршрутизації.

Зважаючи на те, що управляючий трафік в мережі є необхідним з точки зору формування маршрутної інформації, при вирішенні першої задачі виникає задача вибору такої структури кластера, де об'єм управляючого трафіка буде зводитись до мінімуму. В цьому випадку при формуванні кластерної структури мережі необхідно врахувати параметри такі, як кількість вузлів кластера, швидкість їх пересування, частота реконфігурацій кластера, тощо. З іншого боку при виборі, скажемо, мінімального розміру кластера з мінімальною кількістю вузлів збільшується кількість кластерів і різко зростає складність міжкластерної маршрутизації.

Питання формування оптимальної структури кластера розглянуті в роботі [1].

В роботі [1] також проведено дослідження залежності ефективності маршрутизації в мобільній мережі від частоти реконфігурації кластера. Показано, що при реконфігурації кластера виникає лавинне розповсюдження управляючої інформації. При збільшенні частоти реконфігурації об'єм управляючої інформації збільшується, що значно знижує ефективність маршрутизації (1).

В даній статті приведено дослідження залежності об'єму управляючої інформації від таких факторів, як ступень зв'язності вузлів кластера і розмір кластера.

Щодо задачі підвищення стійкості маршрутизації, то для її вирішення можна застосовувати метод підвищення стійкості віртуального з'єднання, запропонований у роботі [1, 7].

Методи підвищення ефективності маршрутизації в мобільних комп'ютерних мережах.

За збільшення розміру мобільної мережі і швидкості переміщення мобільних вузлів в мережі різко збільшується об'єм службового трафіка, час формування шляхів передачі даних і обчислювальна складність алгоритмів маршрутизації [2], тому найбільш ефективним є ієрархічний підхід до рішення задачі маршрутизації [2 – 6].

В такій системі частина маршрутизаторів обмінюється інформацією лише з маршрутизаторами своєї групи, а інша частина крім того і з маршрутизаторами інших груп. В мобільній мережі ця модель дозволяє локалізувати зміни топології всередині кожної логічної групи вузлів. На рис. 1 зображений загальний принцип ієрархічної організації структури мережі, запропонований у роботі [1]. Задача маршрутизації у цьому випадку розкладається на два рівня: внутрішній – передача інформації між вузлами одної логічної групи, зовнішній – між групами вузлів [1, 2, 9]. Це дозволяє на кожному рівні застосовувати найбільш ефективні алгоритми маршрутизації.

В роботі [1] запропонований алгоритм формування динамічної структури кластерів, який є повністю розподіленим алгоритмом для мобільних мереж без інфраструктури класу Ad Hoc. Запропонований алгоритм поділяє мобільну мережу на зони, що називають кластерами. Алгоритм за рахунок динамічної реконфігурації кластерів дозволяє зменшити об'єм управляючого трафіка. При формуванні оптимальної з точки зору зменшення об'єму управляючої інформації структури кластерів в якості вхідних параметрів розглядаються такі параметри, як ступень зв'язності вузлів кластера, максимальний діаметр кластера, максимальна кількість вузлів кластера.

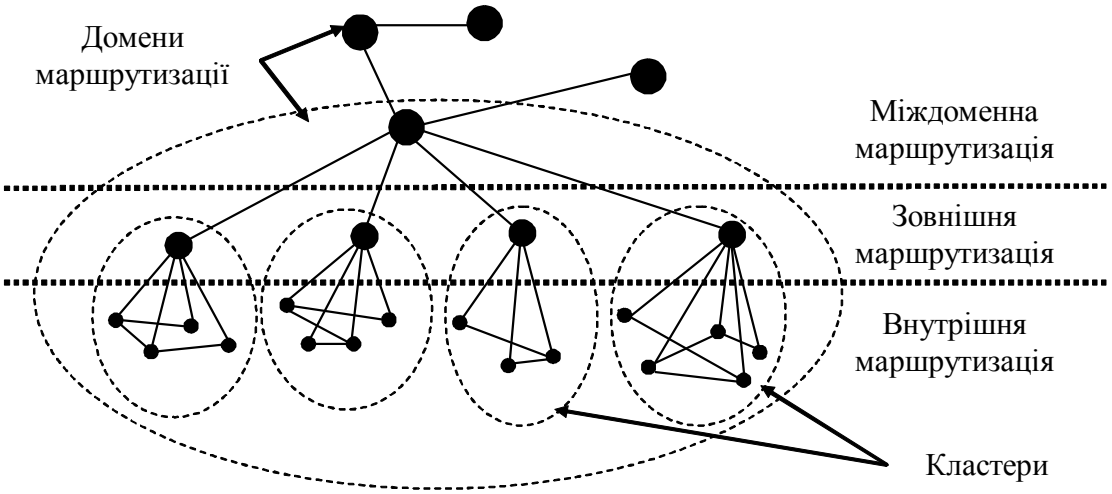


Рис. 1. Ієрархічна структура мобільної мережі.

Дослідження залежності потоку управляючої інформації від кількості вузлів кластера та їх ступеню зв'язності

Розглянемо характер впливу ступеню зв'язності вузлів мережі на об'єм управляючої інформації. Мережа задана регулярним графом G(V,E), де V=(v1,…, v|V|) – множина вершин графу, E=(e1,…, e|E|) – множина дуг, що відповідають |E| каналам зв'язку. Нехай в момент часу t1 вузол v | v∈V лавинним чином розповсюджує інформацію про поновлення свого стану між усіма вузлами мережі.

На рис. 2 наведена регулярна структура мережі із ступенями зв'язності вузлів S=6.

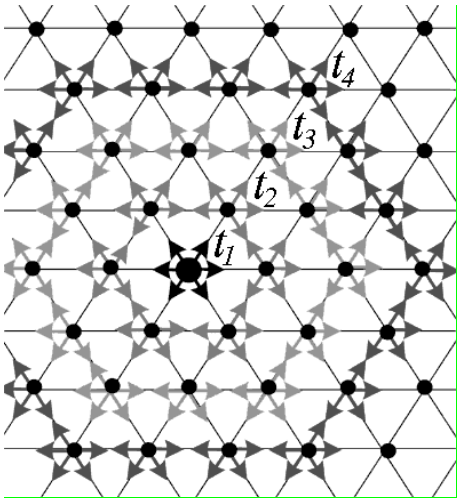


Рис. 2. Регулярна структура мережі (S=6).

Визначимо кількість управляючих пакетів H, що ініційовані вузлом v | v∈V і розповсюджені мережею послідовно його «сусідами» у кожному такті tr роботи лавинного алгоритму, де r – радіус мережі, що співпадає з номером такту:

$$H_{t_1} = S, \qquad (2)$$

$$H_{t_2} = S(S-1), \qquad (3)$$

$$H_{t_3} = S(S-1) + S(S-2), \qquad (4)$$

$$H_{t_4} = S(S-1) + 2S(S-2), \qquad (5)$$

$$H_{t_5} = S(S-1) + 3S(S-2), \qquad (6)$$

$$H_{t_6} = S(S-1) + 4S(S-2), \qquad (7)$$

… .

Виходячи з формул (2 – 7), отримаємо загальну кількість управляючих пакетів, що розповсюджуються мережею в одному такті лавинного алгоритму:

$$H_{t_r} = S(S-1) + S(r-2)(S-2). \quad (8)$$

Перетворюючи формулу (8), отримаємо вираз для визначення кількості службових пакетів, що розповсюджуються мережею в одному такті (S=6):

$$H^6_{t_r} = S^2(r-1) - S(2r-3). \qquad (9)$$

Тоді загальна кількість управляючої інформації, що розповсюджена вузлом у мережі при зміні стану його каналів зв'язку дорівнює:

$$V\text{сл}^6_v = \sum_{r=1}^{\infty} H^6_t = \sum_{r=1}^{\infty} (S^2(r-1) - S(2r-3)) \quad (10)$$

Випадки для мереж із регулярною структурою і ступенями зв'язності вузлів S=3, S=4 и S=8 також розглянуті.

В одному такті лавинним алгоритмом розповсюджується наступна кількість управляючих пакетів (відповідно до ступеню зв'язності вузлів мережі):

$$H_{t_r}^{\text{Знч}} = S^2(r-1) - S(\frac{3}{2}r - \frac{5}{2}), \quad (11)$$

$$H_{t_r}^{4} = S^2(r-1) - S(2r-3), \quad (12)$$

$$H_{t_r}^{8} = S^2(r-1) - S(3r-5). \quad (14)$$

Тоді загальна кількість управляючої інформації, що розповсюджена вузлом у мережі при зміні стану його каналів зв'язку відповідно дорівнює:

$$V\text{сл}_v^4 = \sum_{r=1}^{\infty}(S^2(r-1) - S(2r-3)), \quad (15)$$

$$V\text{сл}_v^8 = \sum_{r=1}^{\infty}(S^2(r-1) - S(3r-5)). \quad (16)$$

На підставі порівняльного аналізу формул (9, 11, 12, 13) можна говорити про усереднену кількість управляючих пакетів, що розповсюджуються у мережі, або її частині, визначеної як кластер, із довільною кількістю вузлів та їх ступенем зв'язності.

Таблиця 1. Кількість управляючих пакетів

Зв'язність вузлів	Радіус мережі (r, вузлів)									
	1	2	3	4	5	6	7	8	9	10
S=3	3	9	18	30	45	63	84	108	135	165
S=4	4	16	36	64	100	144	196	256	324	400
S=5	5	25	60	110	175	225	350	460	585	725
S=6	6	36	90	168	270	396	546	720	918	1140
S=8	8	64	168	320	520	768	1064	1408	1800	2240
S=10	10	100	270	520	850	1260	1750	2320	2970	3700

Після відповідних обчислень отримаємо середню кількість управляючих пакетів, розповсюджених лавинним алгоритмом за один такт:

$$H_{t_r}^{cp} = S^2(r-1) - S(2r-3). \quad (17)$$

Тоді усереднений об'єм управляючого трафіка розповсюдженого у мережі лавинним алгоритмом при зміні положення одного вузла дорівнює:

$$Vсл_v = \sum_{r=1}^{\infty}(S^2(r-1) - S(2r-3)) \quad (18)$$

В табл. 1 наведені численні значення усередненої кількості управляючих пакетів, що розповсюджуються мережею лавинним алгоритмом при зміні положення одного вузла. Кількість управляючих пакетів визначена в залежності від степеню зв'язності вузлів мережі і радіуса мережі на підставі формули (18).

Висновки

Під час реконфігурації в мобільній мережі виникає лавинне розповсюдження управляючої інформації, причому із збільшенням частоти реконфігурацій об'єм управляючого трафіка різко збільшується. Проведене дослідження показало, що при формуванні структури кластера, з точки зору мінімізації об'єму управляючої інформації, необхідно враховувати такі параметри, як зв'язність вузлів у кластері і діаметр кластера.

На підставі проведених розрахунків можна зробити висновок, що об'єм управляючого трафіка, що виникає у мобільній мережі під час реконфігурації кластера, залежить від зв'язності вузлів кластера і радіуса кластера (причому діаметр кластера (D) дорівнює D = 2r). При радіусі кластера більш ніж два із збільшенням ступеню зв'язності вузлів кількість управляючих пакетів збільшується за нелінійним законом.

Перелік посилань

1. Клименко И.А. Способ динамической маршрутизации с поддержкой требуемого уровня качества обслуживания в мобильных сетях без фиксированной инфраструктуры // Проблеми інформатизації та управління: Зб. наук. пр. – К.: НАУ, 2005. – Вип. 15. – С.102–112.
2. Клименко И.А. Способ адаптивной маршрутизации с учетом параметров качества обслуживания в мобильных сетях Ad Hoc // Пр. Наук.-практичної конф. молодих вчених та аспірантів "Інтегровані інформаційні технології та системи" (ІІТС–2005). – Київ: НАУ, 2005. – С.78–80.
3. Mohammad S, Al-Fares., Zhili Sun ., Haitham Cruickshank Hierarchical Routing Protocol for Survivability in Wireless Sensor Network // Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, March 18 - 20, 2009, Hong Kong
4. Dina S. M. Hassan , Hossam M. A. Fahmy, Ayman M. Bahaa Clustering Algorithm for wireless ad hoc networks // Melecon 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference
5. Chen Y.P., Liestman A.L. A zonal algorithm for clustering Ad Hoc networks // International Journal of Foundations of Computer Science. – 2003. – №14, №2. – P.305–322.
6. Nikaein N., Labiod H., Bonnet C. DDR-distributed dynamic routing algorithm for mobile ad hoc networks // Proc. 1st ACM International Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc 2000).– Boston, Massachusetts (USA), 2000. – P.19–27.
7. A New Two-Level Clustering Scheme for Partitioning in Distributed Wireless Sensor Networks
8. // International Journal of Distributed Sensor Networks Volume 2015 (2015), Article ID 435048
9. Chatterjee M., Das S. K., Turgut D. WCA: A weighted clustering algorithm for mobile ad hoc networks // Cluster Computing. – 2002. – Vol.5, №2. – P.193– 204.
10. 9. Xuxun Liu International Journal of Distributed Sensor Networks Volume 2015 (2015), Article ID 435048 // IEEE sensors journal, vol. 15, no. 10, october 2015.

УДК 004.93+510

Василенко В. Г.,
Ширій В. В.,
Баклан І. В.

ІДЕНТИФІКАЦІЯ КОРИСТУВАЧІВ КОРПОРАТИВНОЇ МЕРЕЖІ З ВИКОРИСТАННЯМ ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ

Розглядається використання лінгвістичного моделювання, як одного з напрямків нечисельного моделювання, для ідентифікації користувача у корпоративній мережі. Описується загальна структура системи ідентифікації та алгоритм реалізації методу на базі інтервального підходу, в основі якого лежить процес відновлення формальної граматики.

Ключові слова:Лінгвістичне моделювання, біометрична ідентифікація, розпізнавання образів, інтервальний підхід

Examines the use of linguistic modeling, as one of the areas of numerical modeling, to identify the user on the corporate network. Describes the general structure of the system identification algorithm and the realization method based on interval approach, which is based on the recovery process of a formal grammar.

Keywords: Linguistic modeling, biometric identification, pattern recognition, interval approach

Вступ

В даний час більшість комп'ютерних систем і веб-сайтів онлайн, ідентифікує користувачів за допомогою логіну і паролю. Але зазвичай, хакери можуть легко вкрасти пароль за допомогою великої кількості методів. Деякі з цих методи - напади фішингу, кейлогерів тощо. Також комп'ютер користувача може залишатися невимкнутим в той час, коли хакери зможуть встановити кейлогер на комп'ютер або надіславши деякі посилання на цей комп'ютер, наприклад, привітання або зображення і тд. Якщо користувач натискає на ці посилання, кейлогер починає записувати кожне натискання клавіш, що в подальшому можуть містити логін та пароль користувача, та скріншоти після декількох хвилин роботи. Також вони можуть посилати дані хакерам, не повідомляючи про це користувача.

Недолік ідентифікаційних методів, які засновані лише на основі облікових даних призводять до введення користувальницьких методів ідентифікації і перевірки. Засновані вони на поведінковій і фізіологічній біометрії, які, як передбачається, унікальні один від одного і саме через це, в майбутньому, дані важко вкрасти. Автентифікація виконується лише один раз на початку сеансу, в той час як, перевірка ідентичності виконується безперервно протягом сесії. Перевірка ідентичності може бути досягнута за допомогою одного з двох методів: поведінкової або фізіологічної біометричної системи. Поведінкова біометрична особливість включає особливості взаємодії користувача і пристроїв введення, таких як миша і клавіатура. А фізіологічне використовує людські особливості, які є унікальними для індивіда. Для прикладів: відбитки пальців, візерунки райдужної оболонки ока, обличчя, рухи губ, хода/крок, голос/мова, підпис/почерк тощо.

Найбільш поширені поведінкові біометричні методи верифікації засновані за:

- допомогою руху миші [1] [2] [3] [4], які випливають із взаємодії користувача миші;
- динамікоюнатискань клавіш [6] [7] [8], які отримані від активності натискання користувачем клавіатури;

- взаємодією програмного забезпечення, які покладаються на особливості, взятііз взаємодії користувача з конкретним програмним засобом.

Таким чином системи, що використовують біометричну перевірку користувача, вимагають хакера, який зможе не тільки просочитися в систему, щоб вкрасти облікові дані користувача, але також і наслідувати користувацьку поведінкову або фізіологічну біометрію, що робить викрадення ідентифікаційних даних набагато важчою задачею. Ми зосереджені на поведінковій системі перевірки, тому що вона, на відміну від фізіологічної перевірки, не вимагає додаткових апаратних засобів. Очевидно і те, що вартість такої системи не може бути більшою.

Модель методу перевірки

Розглянемо загальну модель методу перевірки, яказзаснованана русі миші користувача. Цей метод вимагає збереження і обробки десятків координаті дій миші, перш ніж можливо виконатиперевірку. Перевірка кожної окремої дії миші підвищує точність при одночасному скороченні часу, який необхідний для перевірки автентичності користувача. В порівнянні з підходом, який показаний на гістограмі в [1], менше дій потрібно для досягнення певного рівня точності.

Загальну блок-схему запропонованої системи показано на рис. 1

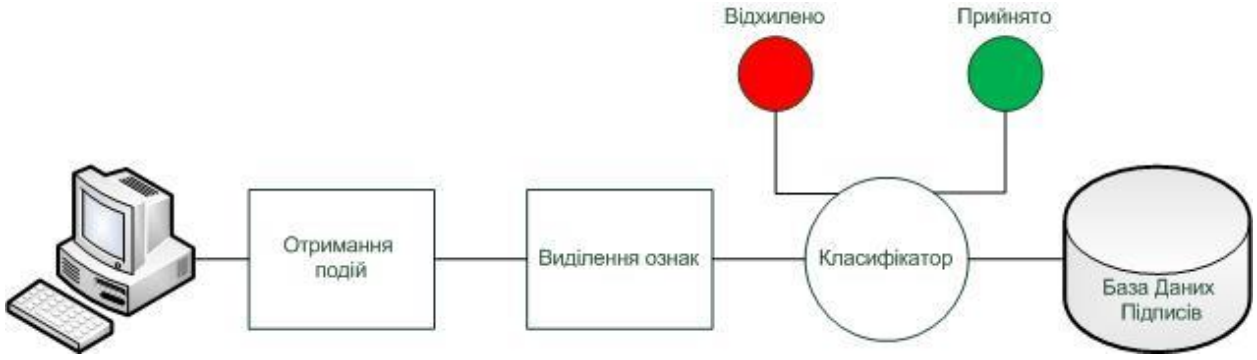


Рис. 1. Типова структура поведінкової системи біометричної ідентифікації

Рис.1 зображує типову архітектуру поведінкової користувальницькоїсистеми перевірки біометрії. Система включає такі компоненти:

- **Отримання подій -** оброблює події, вироблені пристроєм введення. В нашому випадку, це є мишка.Події можуть бути рухом миші (MouseMove, MM), рух вниз ліво (leftdown, LD), вліво вгору (leftup, LU), вправо вниз(rightdown, RD), вправо вверх (rightup, RU), очікування руху (silence, S) і тд.
- **Виділення ознак** - будує підпис, який характеризує поведінкові біометричнідані користувача. Ознаки можутьвключати послідовність переміщення миші (MMS), натискання лівої (LC) та правої кнопки миші (RC) тощо.
- **Класифікатор** - складається з індуктора (наприклад, Векторні Машини Підтримки, Штучні нейронні мережі, тощо), який використовується для побудови моделі перевірки користувача для перевірки підписів. Під час перевірки, модель використовується для класифікації нових зразків підписів, отриманих від користувача.
- **База даних Підписів** - база даних поведінкових підписів, заснованих на навчанні моделі. Якщо кілька користувачів використовують комп'ютер, після введення логіну одного з користувачів, буде відновлено процес перевірки підписів.

Використання лінгвістичногомоделювання

Основним завданням лінгвістичного моделювання є перетворення чисельних рядів, масивів координат до лінгвістичних послідовностей та відновлення за ним формальної

граматики мови відповідного характеру для вирішення проблем які виникли: аналіз та прогнозування часового ряду, автентифікація користувача за його рухами [10].

Лінгвістичне моделювання базується на трьох основних підходах: структурний підхід та математична лінгвістика, інтервальні обчислення та робастні методи, сучасні методи ймовірнісного моделювання.

В основі лінгвістичного моделювання лежить лема існування ізоморфізма відтворення чисельних даних до лінгвістичних послідовностей, на основі яких може бути побудована мова. Як висновок існування унікальної мови, яка фактично уособлюється наборами чисельних даних.

Початок цього був покладений при створенні математичних основ автентифікації користувача складною технічною системою. Автентифікація - шлях встановлення вірогідності інформації, пред'явленої користувачем у разі звернення його до системи та відкриття йому доступу, якщо він має на це право. Дамо загальну постановку завдання. Аналізується кінематика рухів користувача при керуванні складною технічною системою. В нашому випадку – це оператор, що сидить за комп'ютером та взаємодіє з корпоративною мережею. При цьому стоїть необхідність автентифікації користувача - в цей момент керує складною технічною системою оператор А чи не А.

Вхідними даними для моделювання є данні зняті з руху миші довжиною M:$X = \{X_i\}_{i=1}^{M} = \{X_1, X_2, \dots, X_i, \dots, X_M\}$, який описує деякий динамічний процес.

Для початку необхідно побудувати на основі часового ряду $X = \{X_i\}_{i=1}^{M}$різницевих рядів $X_1, X_2, \dots$:

$$X_i^1 = X_{i+1} - X_i$$
$$X_i^2 = X_{i+1}^1 - X_i^1$$
$$\dots.$$

Після чого сортуємо ряд за зростанням $X^1 \rightarrow X^{s1}$ та знаходимо max(X^{s1}) та min(X^{s1}). Розбиваємо відрізки [min(X^{s1}),0] та [0,max (X^{s1})] на N відрізків за правилами інтервалізації, відповідно до свого варіанту. N змінюється від 10 до 33 цей параметр залежності від алфавіту, який буде обраний на етапі лінгвістизації.

Розбиття на інтервали відбувається таким чином, щоб кількість елементів різнецевого ряду в кожний інтервал потрапляла у відповідності до певного розподілу. Тобто частота попадання елементів до інтервалу [a,b] дорівнювала теоретичній ймовірності $P\{x \in [a,b]\} = F(b) - F(a)$, де F — функція відповідного розподілу.

В результаті отримуємо дві множини інтервалів:

$$I_{0,1} = [a_0, a_1], I_{1,2} = [a_1, a_2], \dots., I_{N-2,N-1} = [a_{N-2}, a_{N-1}], I_{N-1,N} = [a_{N-1}, a_N],$$

$$\partial e a_0 = \min(X^{s1}), a_N = 0;$$
$$J_{0,1} = [b_0, b_1], J_{1,2} = [b_1, b_2], \dots. , J_{N-2,N-1} = [b_{N-2}, b_{N-1}], J_{N-1,N} = [b_{N-1}, b_N],$$

$$\partial e b_0 = 0, b_N = \max(X^{s1});$$

В подальшому необхідно відсортуємо символи алфавіту у наступному порядку: a_1=z, a_2=y, …, a_{N-1}=b, a_N=z, a_{N+1}=A, a_{N+2}=B, …, a_{2N-1}=Y, a_{2N}=Z.

Далі будується відображення L: $X^1 \rightarrow Y$ за такими правилами:

$$L(x_i) = \begin{Bmatrix} a_j якщо x_i \epsilon I_{j-1,j} \\ a_{N+j} якщо x_i \epsilon J_{j-1,j} \end{Bmatrix}$$

Застосувавши відображення L, до елементів ряду X1. В результаті чого отримуємо ряд $L(x_1^1), \dots, L(x_M^1)$.

Будуємо матрицю передування для прихованої марковської моделі. Множина станів - це обраний нами алфавіт. Для кожної пари станів, наприклад $< d, S >$ підраховуємо$V_{d,S}$ скільки разів вона зустрічається в лінгвістичному ланцюжку $L(x_1^1)..., L(x_M^1)$. Поділивши $V_{d,S}$ на

загальну кількість входжень літери “d” дотримуємо частоту переходів зі стану “d в стан “S”: $v(d \to S) = \frac{v_{d,S}}{w_d}$.

Далі знаходимо в лінгвістичному ланцюгу повтори переходу від двох, трьох та більше станів. Далі необхідно побудувати розширену матрицю, додавши до станів варіанти двох, трьох та більше станів, що зустрічаються в нашому лінгвістичному ланцюгу. Після чого побудувати по розширеній матриці передування правила ймовірнісної граматики. Тобто для кожної ненульової клітинки будується правило наступного вигляду: $Sax \xrightarrow{0.5} Z$.

Алфавіт та правила передування утворюють лінгвістичну модель ряду X^1. Ту саму процедуру побудови лінгвістичної моделі повторюємо для інших різниць ряду X - X^2, X^3, X^4, X^5, X^6. Будуємо лінгвістичну модель для алфавіту потужності — 10, 15, 20, 26. Та аналізуємо відмінностей результатів лінгвістичного моделювання одного й того ж самого чисельного ряду, які виникають при двох різних правилах інтервалізації.

Висновок

Було розглянуто загальну модель перевірки біометричної ідентифікації користувача, наведено компоненти цієї системи. Кожна з цих компонент виконує свою певну функцію. Наприклад, База Даних Підписів зберігає поведінкові підписи моделі, що дозволяє виявляти сторонніх користувачів чи користувачів, що зареєстровані, однак використовують інший обліковий запис.

Лінгвістичне моделювання використовуючи перетворення чисельних рядів чи багатовимірних данихв лінгвістичні послідовності, перетворює ці послідовності в формальну граматику. За допомогою цього моделювання, можливо ідентифікувати користувача в корпоративній мережі та виявляти зловмисників, які могли вкрасти дані працівників компанії.

Однак, лінгвістичне моделювання можливо використовувати і для визначення емоційного стану користувача, діагностування хвороб, зв’язаних з рухом кінцівок (тремор, тік чи хорея).

Перелік посилань

1.ClintFeher, YuvalElovici, RobertMoskovitch, LiorRokach, AlonSchclar, “Useridentityverificationviamouse dynamics”, Information Sciences 201 (2012) 19–36.

2. Chao Shen, Zhongmin Cai, Xiaohong Guan, Youtian Du, and Roy A. Maxion, User Authentication Through Mouse Dynamics. IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITYVOL. 8, NO. 1, Jan 2013.

3.Zach Jorgensen and Ting Yu, On Mouse Dynamics as a Behavioral Biometric for Authentication. ACM 978-1-4503-0564-8/March 2011.

4.Z. Jorgensen, T. Yu, “On mouse dynamics as a behavioral biometric for authentication, in: Proceedings of the Sixth ACM Symposium on Information, Computer, and Communications Security” (AsiaCCS), March 2011.

5.Saurabh Singh, Dr K V Arya, “Mouse Interaction based Authentication System by Classifying the Distance Traveled by the Mouse” International Journal of Computer Applications (0975 – 8887) Volume 17– No.1, March 2011.

6.Lívia C. F. Araújo, Luiz H. R. Sucupira Jr., Miguel G. Lizárraga, Lee L. Ling, andJoão B. T. Yabu-Uti, “User Authentication through Typing Biometrics Features, IEEE Transactions on Signal Processing”, Vol. 53, No. 2, February 2005

7.S. Cho, C. Han, D.H. Han, H.I. Kim, “Web-based keystroke dynamics identity verification using neural network, Journal of Organizational Computing and Electronic Commerce” 10 (4) (2000) 295–307.

8.L. Ballard, D. Lopresti, F. Monrose, “Evaluating the security of handwriting biometrics, in: The 10th International Workshop on Frontiers in Handwriting Recognition” (IWFHR „06), La Baule, France, 2006.

9.B J Gorad, D. V Kodavade, “User Identity Using Mouse Signature, IOSR Journal of Computer Engineering”, Volume 12, Issue 4, Jul. – Aug. 2013, PP 33-36.
10.Баклан І.В. Лінгвістичне моделювання: основи, методи, деякі прикладні аспекти / І. В. Баклан // Систем. технології. — 2011. — № 3. — С. 10-19. — Бібліогр.: 9 назв. — укр.

УДК 683.519

ГАНЖА І. М.,
СТІРЕНКО С. Г.

МОДИФІКАЦІЯ КЛАСИЧНОГО АЛГОРИТМУ МАРКУВАННЯ ЗВ'ЯЗНИХ КОМПОНЕНТ НА БІНАРНОМУ ЗОБРАЖЕННІ

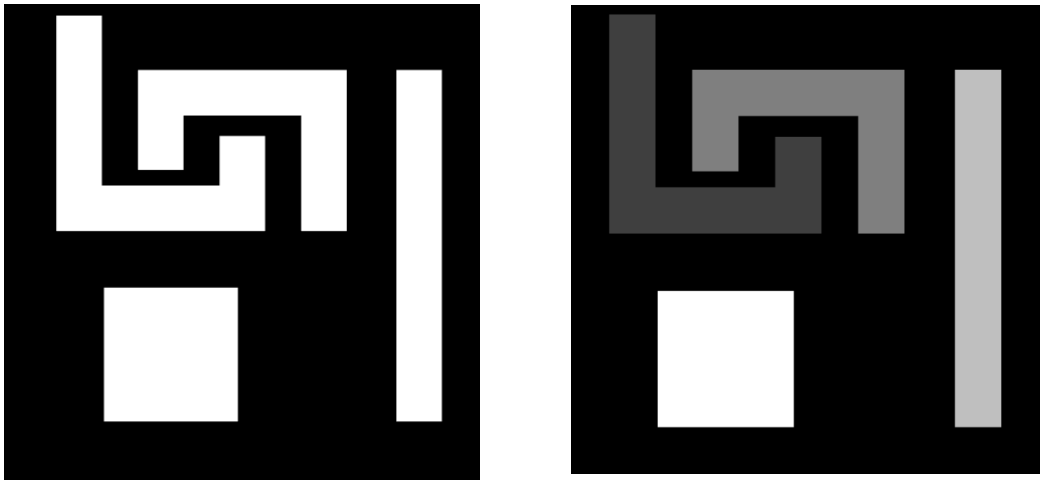
В статті розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, приведено спосіб реалізації алгоритму та запропоновано використання динамічного масиву для збереження класів еквівалентності. Алгоритм маркування зв’язних компонент використовується для виділення зв’язної компоненти, що використовується для подальшого морфологічного аналізу компоненти.

Inthearticleconsideredthetwo-passconsistentconnectedcomponentslabelingalgorithm, a waytoimplementthealgorithmandtheuseof a dynamicarraytostoretheequivalenceclasses. Connectedcomponentslabelingalgorithmusedtoselect a connectedcomponent, whichusedforfurthermorphologicalcomponentanalysis.

Вступ

Двухпрохідний порядковий алгоритм маркування зв'язних компонент або класичний алгоритм маркування зв'язних компонент на бінарному зображенні [2] виявляє зв'язні компоненти(області) переднього плану та надає виявленій компоненті чисельне значення або маркер. В результаті роботи алгоритму, з початкового зображення, в якому всі пікселі компонент мають значення 1, формується маркіроване зображення, в якому пікселі компоненти мають значення маркера компоненти. Нарис. 1а наведено початкове бінарне зображення, в якому всі чотири компоненти мають однакові значення пікселів. На рис. 1б зображено результат алгоритму маркування, кожна компонента отримала свій маркер, всі пікселі компоненти мають значення маркера, внаслідок чого компоненти зображені різними значеннями інтенсивності.

Маркування зв'язних компонент необхідне для подальшого морфологічного аналізу виявлених компонент. Наприклад визначаються ознаки форми сформованої зв'язної компоненти[1].



а. Початкове зображення *б. Маркіроване зображення*

Рис. 1. Приклад роботи алгоритму маркування зв'язних компонент

Опис алгоритму

Класичний алгоритм виконується за два проходи. На першому проході формуються класи еквівалентності та надаються тимчасові маркери зв'язним компонентам або їх частинам. На другому проході тимчасові маркери заміняються маркерами відповідно до кореня структури об'єднання-пошуку класу еквівалентності.

Клас еквівалентності являє собою множину тимчасових маркерів окремої зв'язної компоненти. Клас еквівалентності в класичному алгоритмі маркування представлено в структурі об'єднання-пошуку, яка зберігає тимчасові мітки в деревоподібній структурі, з'єднує дерева між собою та дозволяє швидко знаходити корінь дерева [1]. На рис. 2 наведено приклад структури об'єднання-пошуку у вигляді таблиці та графів дерев.

0	1	2	3	4	5	6	7	8
0	2	6	6	8	6	0	8	0

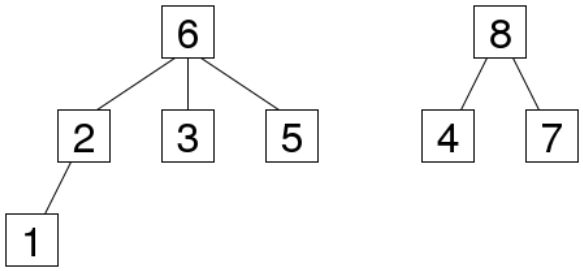


Рис. 2. Структура об'єднання-пошуку у вигляді таблиці та графу

Таблиця на рис. 2 в першому рядку містить значення маркерів від 1 до кількості маркерів, значення 0 не враховується адже відповідає значенню фону зображення. Другий рядок містить значення маркера-батька для маркера з першого рядка, значення 0 означає, що маркер являється коренем дерева. На рис. 2 зображено два графи дерев, що відповідають двом структурам об'єднання-пошуку або класам еквівалентності. Два дерева об'єднують дві множини маркерів - {6, 5, 3, 2, 1} та {8, 7, 4}. Корені дерев - маркери 6 та 8, тому вони мають значення 0 в другому рядку таблиці. Маркер 1 має маркер-батька 2, тому в другому рядку має значення 2, маркери 2, 3, 5 мають маркера-батька 6, тому значення в другому рядку в усіх 6, маркери 4, 7 мають маркера-батька 8, тому в другому рядку міститься 8. Оскільки в першому

рядку значення являють нумерацію, то дану структуру можливо представити у вигляді одномірного масиву, в якому індекси - це дані першого рядка, а значення масиву - це дані другого рядка.

При необхідності об'єднати два маркери в структуру, або об'єднати дві структури, якщо їхні корені не співпадають, виконується операція об'єднання. Для виконання операції об'єднання двох дерев досить в таблиці в другий рядок кореня дерева, що додається замінити 0 на значення кореня дерева, в яке виконується додавання. Якщо необхідно знайти батька для маркера, щоб визначити до якого класу еквівалентності він належить, виконується операція пошуку. Для пошуку кореня в даній структурі виконується перебирання значень батьків маркерів до того моменту доки не буде знайдено 0.

На першому проході алгоритм надає тимчасовий маркер ще не поміченій області та намагається розповсюдити маркер правим та нижнім сусідам компоненти. Якщо виникає ситуація, що два не однакових маркера намагаються розповсюдитись на один і той же піксель то обирається маркер з меншим значенням. Кожен маркер заноситься в структуру об'єднання-пошуку та при наявності двох різних маркерів в одній компоненті маркери об'єднуються за допомогою описаної вище операції об'єднання. Після першого проходу класи еквівалентності сформовані у вигляді структур об'єднання-пошуку та мають унікальний маркер або ідентифікатор, що відповідає кореню дерева структури. На другому проході тимчасові маркери замінюються ідентифікаторами класів еквівалентності, застосовуючи операцію пошуку кореня дерева, яка описана вище.

На рис. 3 показано результат роботи першого та другого проходів алгоритму та наведено структуру об'єднання-пошуку, яка має класи еквівалентності {2, 1} та {4, 3}.

1	1	0	0	0	0	0	0	0	0
1	1	0	0	2	2	2	2	2	0
1	1	0	0	2	2	2	2	2	0
1	1	1	1	1	1	0	2	2	2
1	1	1	1	1	1	0	2	2	2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	0
0	0	0	0	0	0	0	3	3	0
0	4	4	4	4	4	4	3	3	0
0	4	4	4	4	4	4	3	3	0

а. Результат першого проходу

1	1	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	1	1	0
1	1	0	0	1	1	1	1	1	0
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	2	0
0	0	0	0	0	0	0	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0

б.Результат другого проходу

0	1	2	3	4
0	2	0	4	0

в. Структура об'єднання-пошуку після першого проходу

Рис. 3. Результати роботи двох проходів та структура об'єднання-пошуку

Реалізація алгоритму

Для збереження класів еквівалентності або структур об'єднання-пошуку використано динамічний масив, що дозволяє створювати масив відповідно до кількості тимчасових маркерів.

Перший прохід реалізовано двома послідовними циклами, в яких оброблюються пікселі зі значенням 1 вхідного зображення. Відносно поточного пікселя перевіряться наявність сусідів зі значенням 1 зліва та зверху, з задачею ініціювати маркер MC. Якщо є сусід зліва з маркером ML, то MC := ML, якщо сусід зверху також існує і його маркер MT менше MC, то MC:=MT.

Якщо сусідів не виявлено то в МС записується значення наступного маркера по порядку. Маркер МС записується в піксель результуючої матриці.

Якщо маркер МС не дорівнює значенню зліва або зверху то виконується об'єднання маркерів в структурі об'єднання-пошуку.

На другому проході також виконується прохід двома циклами, в яких оброблюються пікселі зі значенням більше 0. Піксель зберігає значення тимчасового маркера, по якому за допомогою операції пошуку знаходиться корінь структури об'єднання-пошуку. Для уникнення збою в порядку нумерації маркерів створюється окремий масив, в якому значенню кореня структури присвоюється порядковий номер. Порядковий номер, що відповідає кореню структури записується в пікселі зв'язної компоненти та є її остаточним маркером.

Порівняння класичного та рекурсивного алгоритмів.

Порівняння класичного та рекурсивного алгоритмів проводиться на основі сформованого вручну зображення в форматі PGM розміром 800x800 пікселів. Щоб отримати множину зображень різного розміру, початкове зображення розміром 800x800 зменшується на 50 пікселів по ширині та висоті. Для кожного типу алгоритму проводиться по 100 експериментів та визначається середній час роботи алгоритму. Час роботи алгоритмів показано на рис. 4.

На рис. 4 видно, що рекурсивний алгоритм працює швидше класичного, однак, при розмірі зображення 500x500 пікселів рекурсивний алгоритм видає помилку StackOverflow , що означає що програма вичерпала кількість викликів рекурсивних методів, тому на графіках значення часу 0. Оскільки кількість пікселів в компонентах збільшилась, а на кожен піксель при обробці необхідно викликати рекурсивну функцію, значить збільшується кількість рекурсивних викликів функцій.

Класичний алгоритм працює при розмірі зображення 500x500 і більше, адже відсутній рекурсивний виклик функцій, маркери назначаються пікселям на основі значень сусідів та структури об’єднання-пошуку. При збільшенні розміру зображення збільшується час роботи алгоритму.

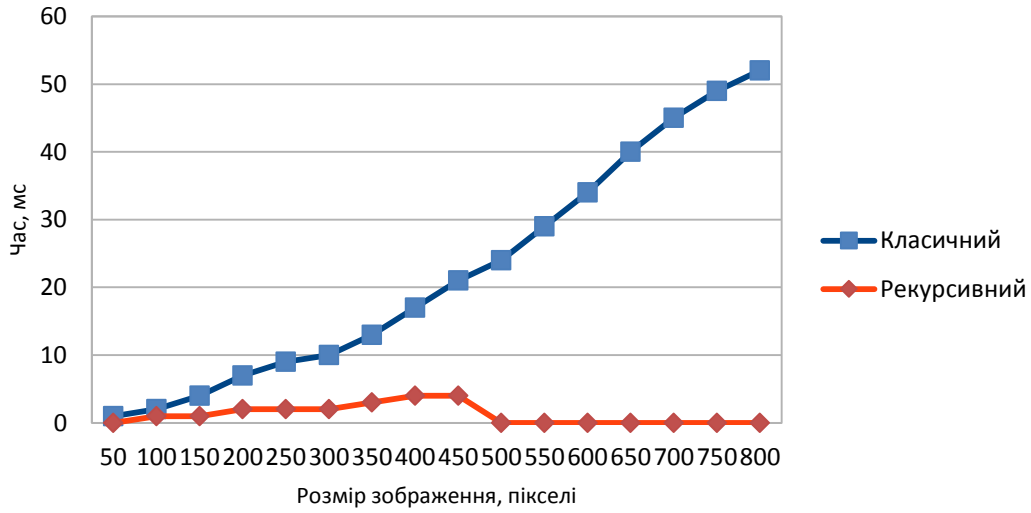


Рис. 4. Час роботи класичного та рекурсивного алгоритмів

Висновок

В статті детально розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, запропоновано модифікацію алгоритму, яка полягає в упорядкуванні маркерів, що полегшує подальший морфологічний аналіз компонент, та приведено опис реалізації алгоритму з використанням динамічних масивів для зберігання структур об'єднання-пошуку. Порівняння часу роботи класичного та рекурсивного алгоритму показало, що класичний

алгоритм програє в швидкості рекурсивному, однак рекурсивний алгоритм при розмірі зображення 500x500 пікселів та більше призводить до переповнення стеку задач. Класичний алгоритм дозволяє обробляти зображення великих розмірів , при цьому збільшується час обробки.

Перелік посилань

1. Л. Шапиро, Дж. Стокман, Компьютерноезрение, М.: Бином. Лабораториязнаний, 2006.
2.A. Rosenfeld, P. Pfaltz, “SequentialOperationsinDigital Picture Processing”, JournaloftheAssociationforComputingMachinery, Vol. 12., 1966.

УДК 004.91

ГНЕННИЙ А.П.

КУЛАКОВ Ю.О.

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ RSS-АГРЕГАЦІЇ

У даній статті розглянуто підхід до побудови веб-додатку для RSS-агрегації з використанням сучасних технологій для веб-розробки. Показано відмінностіміжтрадиційними та односторінковимидодатками, описано проблему збереженняданих на стороніклієнта та вирішенняцієїпроблеми за рахуноквикористаннятехнології локального сховища, щопропонується стандартом HTML5.

This article describes an approach for building RSS-aggregation web-application with usage of modern technologies for web-development. There is shown the difference between traditional and single-page applications and described the problem of data storage on client side and solution for this problem by usage of local storage technology, provided by HTML5 standard.

Актуальність ідеї

У сучасному світі багато людей відвідують сайти, контент на яких часто змінюється. Це, наприклад, сайти новин, ігрові сайти, сайти покупок, медичні портали тощо. Ручна перевірка кожного сайту щодо наявності нового контенту може бути дуже тривалою, втомливою та незручною. Одною із спроб вирішити цю проблему були сповіщення на електронну пошту. На жаль, коли ми отримуємо сповіщення на пошту від багатьох сайтів, листи зазвичай не організовані та можуть засмічувати пошту. Крім того, вони дуже часто сприймаються за спам сучасними поштовими сервісами.

Використання RSS – Really Simple Syndication («Справді Просте Розповсюдження») – набагато кращий спосіб сповіщення про новий чи змінений контент.RSS - це форматданих, який дозволяє відобразити та розповсюдити інформацію у зручний та ефективний спосіб.RSS може забезпечити доступність інформації та фільтрування важливого матеріалу серед величезної кількості ресурсів наявних в Інтернеті [1]. До того ж, користувач може відмовитися від отримання інформації з певного сайту, користуючись лише додатком, а не відвідуючи цільовий сайт.

Існуючі рішення

З точки зору підходу до реалізації RSS-агрегаторів зараз існують два типи. Перший тип представляє собою додатки, які інсталюються на персональний комп’ютер, смартфон чи планшет. Найбільш відомі приклади таких агрегаторів : NetNewsWire, Flipboard, Prismatic та Zite. Перевагою таких рішень є можливість перегляду збережених даних у разі відсутності доступу до мережі Інтернет. Недолік – неможливість доступу до даних з пристроїв, де не встановлено додаток.

Інший тип – агрегатори, що представляють собою веб-додатки, доступні для використання з будь-якого браузера з будь-якого пристрою. Найбільш відомі додатки : Google News, Drudge Report, CityFALCON, Huffington Post, Fark, Zero Hedge, Newslookup, Newsvine та інші.

Недоліком першого типу агрегаторів є те, що ними можна користуватись лише з певного типу пристроїв, для якого написано додаток. Для другого ж типу характерна неможливість перегляду інформації при втраті доступу до мережі. Бажано певною мірою об'єднати переваги та зменшити недоліки таких підходів. Запропонований підхід базується на другому типові – RSS-агрегаторі, що базується на веб-додатках.

Запропоноване рішення

З розвитком технологій в галузі веб-розробки стало можливим зберігати досить значні об’єми інформації локально на пристрої користувача. Мова розмітки HTML5 вводить новий програмний інтерфейс користувача – Local Storage[2], тобто локальне сховище, зі значно більшим об’ємом місця, доступного для збереження даних (мінімум 5 Мб), ніж при використанні куків (4Кб).

Більшість сучасних RSS-агрегаторів, що працюють через браузер, представляють собою традиційні веб-додатки, на яких для перегляду інформації потрібно робити переходи на інші сторінки. У разі зникнення Інтернет-зв’язку подальше користування веб-сайтом неможливе. Рішенням такої проблеми є побудова додатку у вигляді SPA – single-page application, тобто односторінкового додатку. У такому додатку ніколи не відбувається переходу на інші сторінки, а завантаження нових даних відбувається за допомогою таких технологій, як AJAXабо веб-сокети. Більше того, односторінковідодаткиможнавідкривати з локальнихфайлів та користуватисябез доступу до Інтернету. Детальніша інформація про переваги односторінкових додатків над традиційними знаходиться в [3].

На рис. 1 зображено роботу традиційного багатосторінкового веб-додатку.

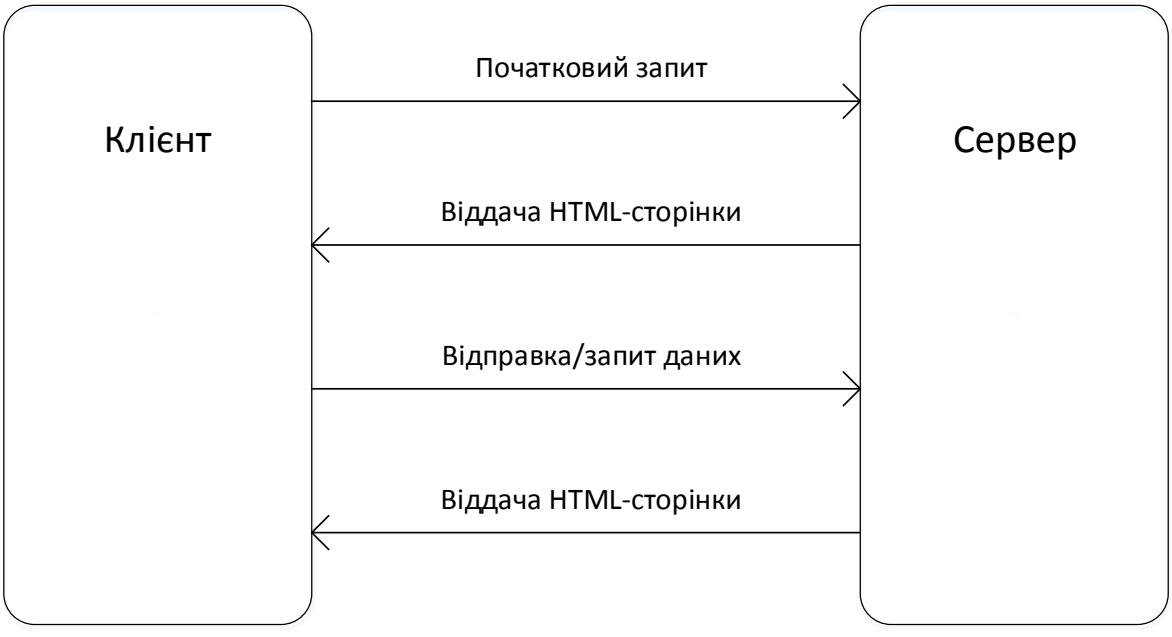


Рис. 1. Робота традиційного веб-додатку

Алгоритм роботи традиційного веб-додатку наступний:

1. Клієнт робить запит до Сервера.
2. Сервер віддає HTML-сторінку.
3. Клієнтвідображаєсторінку.

4. Клієнтробить запит на новусторінку.
5. Сервер віддаєновуHTML-сторінку.
6. Клієнт відображає вже нову сторінку.

Такий підхід має суттєвий недолік – кожного разу на запит Клієнта Сервер відправляє нову сторінку, що зумовлює перезавантаження сторінки. Якщо з'єднання раптово зникне, нова сторінка не завантажиться і користувач не зможе продовжити роботу. Крім того, поточний стан додатку має зберігатися на Сервері, що призводить до додаткового навантаження Сервера.

На рис. 2 зображено роботу односторінкового веб-додатку.

Алгоритм роботи односторінкового веб-додатку наступний:

1. Клієнт робить запит до Сервера.
2. Сервер віддає HTML-сторінку.
3. Клієнтвідображаєсторінку.
4. КлієнтробитьAJAX-запит.
5. Сервер повертаєлишеневеликукількістьданих, необхідних для додатку.
6. Клієнт змінює вигляд сторінки, відображає дані, не перезавантажуючи сторінки.

Рис. 2. Роботаодносторінкового веб-додатку

Отже, односторінковий додаток зменшує час пересилання даних, навантаження на сервер та дозволяє адекватно відреагувати на ситуацію зникнення Інтернет-зв'язку.

Для побудови додатку доцільно використати набір програмного забезпечення MEAN[4], який орієнтований на розробку односторінкових додатків та містить наступні компоненти : MongoDB в якості бази даних, Express.jsв якості сервера, Angular.js у якості фреймворка для клієнтської частини, тобто інтерфейсу користувача, та Node.js – JavaScript платформа для серверної розробки. У таких додаткахвкрайважливовикористовувати описаний ранішепрограмнийінтерфейсLocal Storage, введений у HTML5.

Модель односторінкового додатку з використанням стеку програмного забезпечення MEAN таLocal Storage наведений на рис. 3.

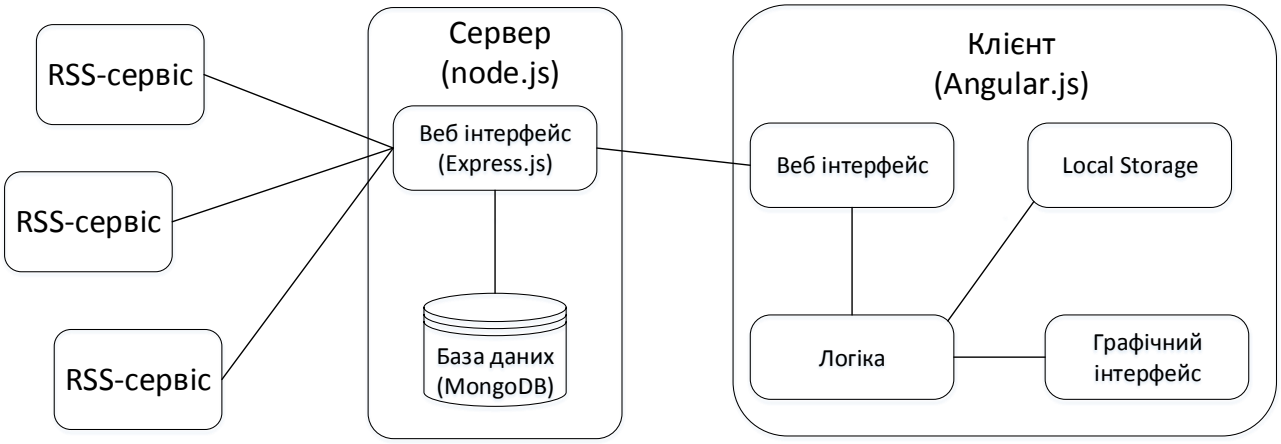


Рис. 3. Модель додатку з використаннямLocal Storage

Алгоритм роботи такого додатку в онлайн режимі (за наявності мережі) наступний :

1. Клієнт робить запит до Серверу.
2. Сервер робить запити до RSS-сервісів.
3. RSS-сервіси надають дані Серверу.
4. Сервер зберігає завантажені дані до бази даних.
5. Сервер надає дані Клієнту.
6. Клієнт відображає дані.
7. Клієнт зберігає дані до Local Storage.

Алгоритм роботи додатку в офлайн режимі (відсутня мережа) наступний :

1. Клієнт перевіряє з'єднання з сервером.
2. Зв'язок відсутній, тому Клієнт завантажує дані з Local Storage.
3. Клієнт відображає дані.

Без використання Local Storage користувач не має змоги переглянути дані та продовжити роботу з додатком.

Висновки

RSS-агрегація –зручнийта потужнийспосіборганізаціїнеобхідних новин в одному місці. Для такоїціліяknайкращепідходитьодносторінковий веб-додаток, забезпечуючинадійний, зручний та швидкий доступ з браузеру користувача. Для написанняклієнтської та серверноїчастинзапропоновановикористовувати одну мовупрограмування – JavaScriptта відповідно набір програмного забезпечення MEAN. Проблему збереження даних вирішує технологія Local Storage.

Список літератури

1.Roger W. McHaney. Web 2.0 and Social Media for Business //Ventus Publishing ApS, 2012. – ISBN 978-87-403-0266-0, c. 109-110.
2.Eric Freeman, Elisabeth Robson // O'Reilly Media, 2011. – ISBN 978-1-449-39054-9, c 413-427.
3.Michael S.Mikowski, Josh C.Powell. Singe Page Web Applications // Manning Publications Co, 2014. - ISBN 9781617290756, c 1-2.
4.Amos Q.Haviv. MEAN Web Development // Packt Publishing, 2014. – ISBN 978-1-78398-328-5, 354 c.

УДК 683.519

ГОНЧАРЕНКО О. Р.,
ВОЛОКИТА А. М.

ТАБЛИЦІ ПОШУКУ ДЛЯ ПІДВИЩЕННЯ ШВИДКОСТІ ОБЧИСЛЕНЬ У КОМПЬЮТЕРНІЙ ГРАФІЦІ

В даному докладі розглянуто таблиці пошуку, використання яких є одним з багатьох методів оптимізації, підвищення продуктивності та швидкодії в сучасних обчисленнях. Демонструються приклади використання деяких таблиць пошуку.

This article describes look-up tables, whose use of is one of the many methods of optimization, increasing productivity and performance in actual computing. Examples of the most common look-up tables usage are demonstrated in this article.

Ключові слова: *таблиця пошуку*. Key words: *look-up table, LUT*.

Таблиці пошуку (англ. lookup table, LUT) – це структура даних, яка використовується для заміни обчислень на операції пошуку. Якщо отримання даних з таблиці пошуку буде швидшим ніж обчислення результатів з нуля (або якщо функція неодноразово отримує на вхід одне і теж саме значення), то використання таблиці дасть значний приріст в продуктивності [1]. Для таблиці обчислюютьсянайбільш поширені вхідні данні. Для запитів, які потрапляють між прикладами з таблиці, алгоритм інтерполяції може генерувати прийнятні наближенні значення шляхом усереднення найближчих значень.

На Рис. 1 зображено приклад таблиці пошуку. Кожна точка набору даних є показником вхідних значень конкретної величини таблиці пошуку. Масив даних таблиці служить в якості дискретного представлення функції, обчисленої в певній точці, яка є індексом даних [2].

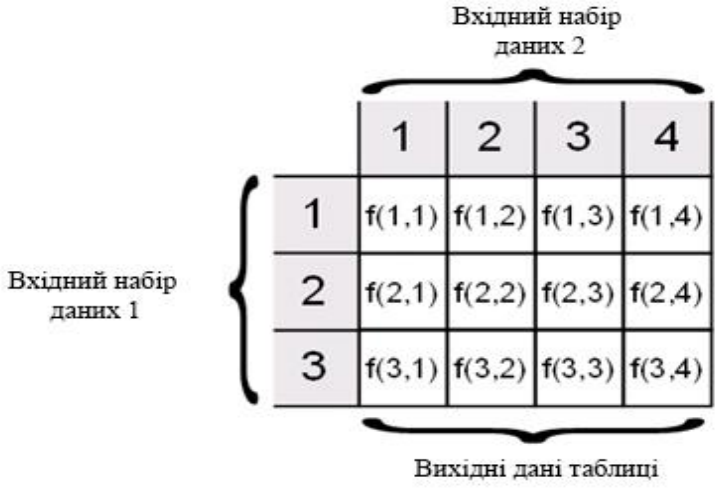


Рис. 1. Приклад таблиці пошуку

Проте, використання таблиць пошуку у простих обчисленнях може призвести до погіршення роботи. Час запиту ресурсів з пам'яті і складність пам'ятіможуть підвищити час виконання програми і складність системи. Можливість росту кеш-пам'яті також може стати проблемою. Запит даних у великих таблицях може призвести до промахів кешу. У деяких мовах програмування (наприклад,Java),звернення до таблиці пошуку може бути навіть більш «дорогим» через обов'язкові перевірки кордонів, що включає в себе додаткові порівняння та розгалуження для кожної операції пошуку.

На приклад – маємо програму, задача якої за зображенням сказати що за об’єкт на ньому знаходиться. На вхідній картинці маємо людину. Спочатку потрібно оптимізувати зображення для подальшої роботи з ним. Потрібно приховати, а по можливості й прибрати непотрібні деталі шляхом збільшення або зменшення яскравості та тіней, привести зображення до сірого спектру і тільки потім запускати алгоритм розпізнавання. Всі попередні етапи можуть бути оптимізовані з використанням таблиць пошуку, потім розглянемо це більш детально.

Розглянемо приклад використання таблиць пошуку у алгоритмі пошуку схожих зображень. Маємо систему з великою кількістю зображень і на вхід отримуємо команду «Знайти схожі зображення». Для цього нам потрібно проаналізувати кожне зображення, зменшити його (зазвичай роблять сітку 8x8), прибрати колір (привести зображення до градацій сірого), створити ланцюг біт, беручи 0 або 1 в залежності від середнього значення кольору в певній точці, перевести отримане значення до простого хешу, отримати таким самим чином хеш вхідного зображення, та на кінець порівняти кількість різних бітів. Час отримання результату пошуку буде помітно зростати з кожним новим зображенням. Але, якщо, ми будемо мати таблицю пошуку з попередньо підрахованими значеннями хешу для кожного зображення, то швидкість формування масиву відповіді значно збільшитися. Окрім швидкого пошуку схожих зображень, ми можемо пришвидшити трансформацію зображення, а саме за рахунок таблиць пошуку оптимізувати приведення зображення до чорно-білого спектру. Потім розглянемо це більш детально.

Таблиці пошуку також широко використовуються в перетворенні (трансформації) зображень. В залежності від складності перетворення збільшення швидкості обчислень може бути здобуте саме за рахунок таблиць пошуку, аж до 15 разів або навіть більше для логарифмічних чи експоненціальних перетворень[3].

На рис.2показані приклади застосування декількох поширених таблиць пошуку[3]. Таблиця пошуку квадратної степені, наприклад, зменшує загальну яскравість зображення, але збільшує контраст, роблячи темні зони зображення більш темними, світлі – яскравішими. З іншого боку таблиця пошуку квадратного кореня, збільшує загальну яскравість, але як тільки ефект стає менш вираженим у яскравих зонах, тоді у темних – зменшується контраст. S-подібна Гауссова або сигмоїдна таблиця пошуку надає світлим і темних зонам гомогенності, відповідності, але збільшує контраст у зонах з середнім освітленням.В додаток до прискорення, набуте за рахунок використання таблиць пошуку, вони також дозволяють виконувати перетворення відтінків сірого, які не можуть бути реалізовані легким шляхом.

Рис.3 показує спеціально визначену таблицю пошуку та її ефект після обробки зображення з рис.2.Таблиця пошуку (рис. 3) повертає чорний колір для дуже яскравих та дуже темних зон, а для зон з середнім освітленням – білий. Таким чином палітру було зменшено до двох рівнів сірого.

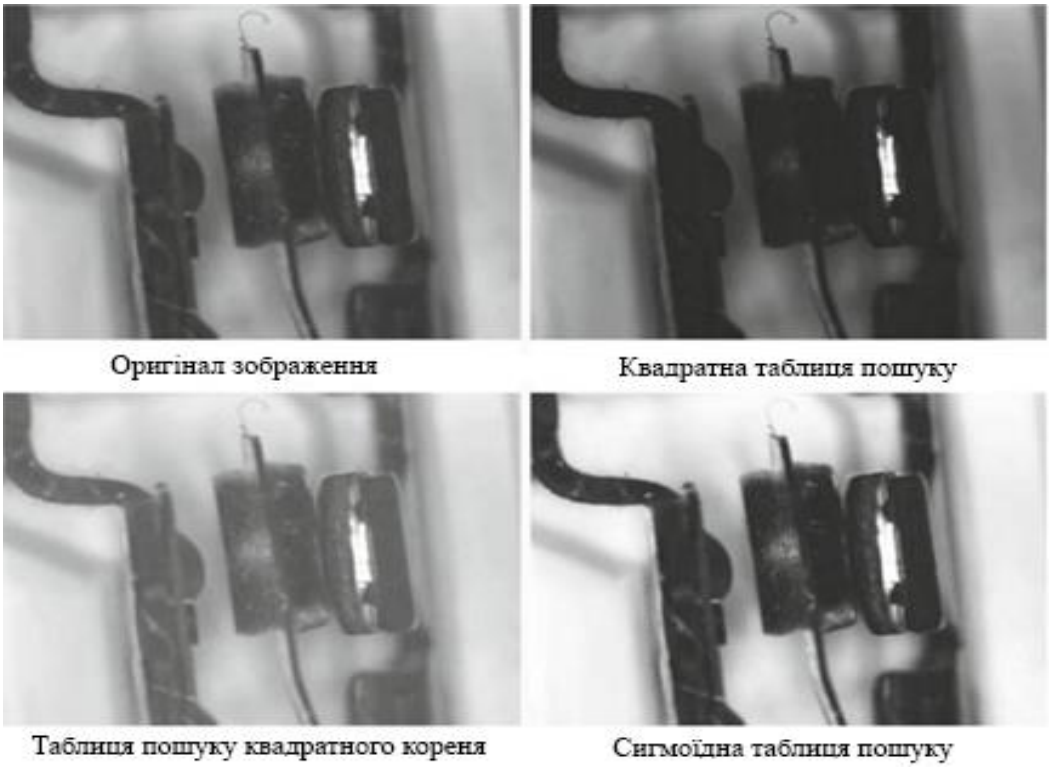


Рис.2. Ефект від поширених таблиць пошуку

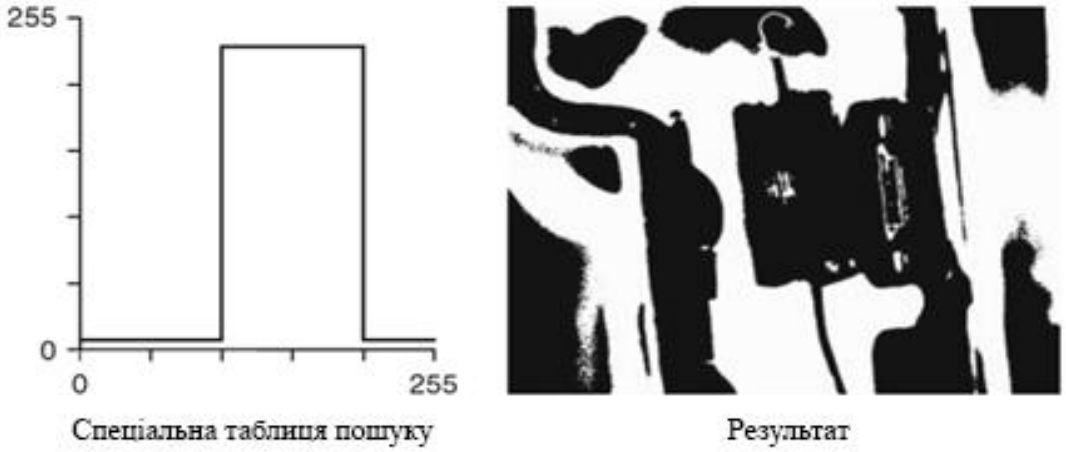


Рис.3. Ефект від спеціально адаптованої таблиці пошуку

Якщо подивитися на це з математичної точки зору, то використання таблиць-пошуку для перетворень у границях сірого, означає, що деяка функція трансформування *f(x)* надана у вигляді таблиці, яка містить одну допоміжну точку функції для кожного можливого значення сірого кольору. Іншими словами таблиця містить певне значення для кожного значення відтінку сірого. Кольорові зображення також можуть бути трансформовані за допомогою таблиць пошуку, але для цього треба буде мати таблицю пошуку для кожного кольорового каналуі така таблиця пошуку називається трьох-вимірною(3D). Прискорення відбувається за рахунок того, що не

потрібно кожен раз обчислювати значення функції для певного параметру, це робиться один раз при першому запуску додатку.

Тривимірні таблиці пошуку індексовані за трьома незалежними параметрами, як показано на рис.4 [4].

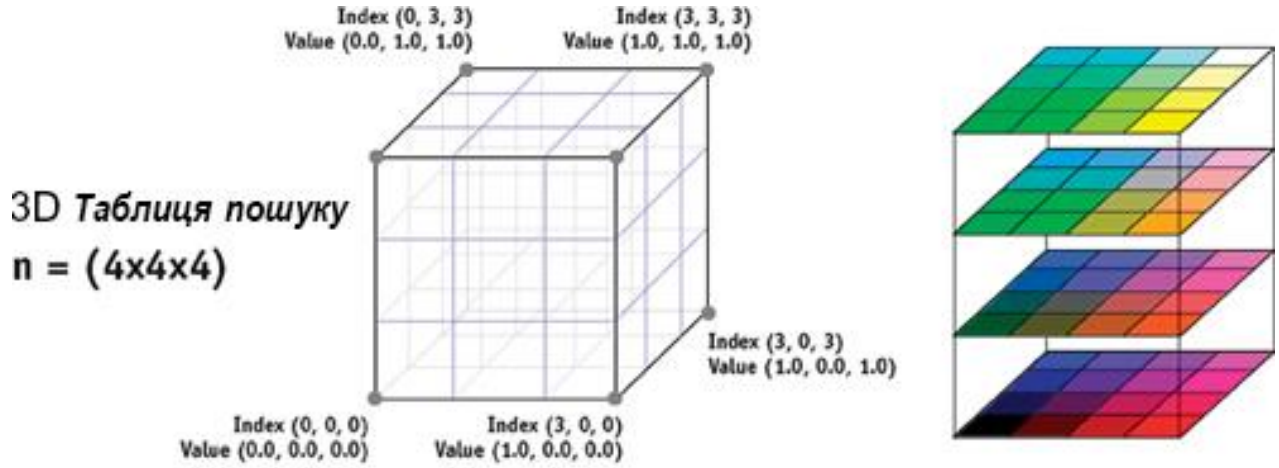


Рис. 4. Трьох-вимірна таблиця пошуку

Тоді як одновимірна таблиця містить тільки 4 елементи в 4-х однакових місцях по кожній осі, відповідна трьох вимірна таблиця пошуку містить 4^3, що дорівнює 64-м елементам. Тому розмірність 3D таблиці пошуку зростає з лінійною частотою дискретизації. Для того щоб помістити в пам'ять 32x32x32 таблицю треба 393 КБ, 256x256x256 таблиця вимагає 200 МБ. Навіть якщо GPU має стільки доступних ресурсів, великі трьох-вимірні таблиці пошуку можуть швидко заповнити весь кеш текстурування, що може погіршити продуктивність. 3D таблиці пошуку є незамінною складовою у системах реального часу, задача яких - відтворювати потокове відео. Саме за рахунок таблиць пошуку відео може бути оптимізоване для перегляду без значного зниження fps шляхом використання певних фільтрів які корегують контраст, яскравість, тон та інші параметри зображення[5].

Висновок

В даній роботі було розглянуто таблиці пошуку, що використовуються для підвищення ефективності обчислень, розглянуто приклади використання таблиць пошуку, приведені приклади використання розповсюджених таблиць пошуку у обробці зображень, а саме: таблиця пошуку квадратної степені, таблиця пошуку квадратного кореня та сигмоїдна таблиця пошуку.За підрахунками NVidia [4] таблиці пошуку підвищують продуктивність більше ніж у 100 разів і це не є межею. В майбутніх дослідженнях можливо використання таблиць пошуку для вирішення задачі розгортання програмного забезпечення в розподілених системах.

Перелік посилань

1.AutomatedMemoizationinC++ [Електронний ресурс] Режим доступу: http://pmcnamee.net/c++-memoization.html(дата запиту: 7.04.2016)

2.Anatomyof a LookupTable - MATLAB [Електронний ресурс] Режим доступу: http://www.mathworks.com/help/simulink/ug/anatomy-of-a-lookup-table.html(датазапиту: 7.04.2016)

3.Christian Demant,BerndStreicher-Abel,CarstenGarnica, " Industrial Image Processing: Visual Quality Control in Manufacturing ", Springer; Softcover reprint of the original; 1 видання 1999 (Листопад 2, 2012) – 353 с.

4.Matt Pharr, "GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation", Addison-Wesley Professional; 1 видання (Березень 13, 2005) – 814 с.
5.Benny Bing, “Next-Generation Video Coding and Streaming”, Wiley; 1 видання (Жовтень 5, 2015)– 334 с.

УДК 004.658.3

ІВАНІЩЕВ Б. В.
ВОЛОКИТА А. М.

ОПТИМІЗАЦІЯ АЛГОРИТМУ РІШЕННЯ ЗАДАЧІ МІГРАЦІЇ ДАНИХ В РОЗПОДІЛЕНИХ СХОВИЩАХ

У цій роботі розглянуто оптимізацію складання плану міграції даних в розподілених масштабованих сховищах. Запропоновано алгоритм складання плану міграції оптимальний за критерієм часу міграції на масштабуючі пристрої зберігання. Доведено, що запропонований алгоритм має поліноміальну складність.

This report reviewed the optimization of drawing up a data migration plan for distributed scalable storage. The algorithm of drawing up a migration plan is optimal by timemigration to scalable storage devices..Itisprovedthattheproposedalgorithmhaspolynomialcomputationalcomplexity.

Ключові слова: розподілені сховища даних, масштабування сховищ даних, балансування даних, міграція даних.

Вступ

Сучасні розподілені сховища даних в більшості є масштабованими, тобто кількість і склад пристроїв зберігання, з яких складаються ці сховища, непостійні і можуть динамічно змінюватися з часом. Постійне підключення і відключення пристроїв зберігання призводить до того, що рівномірний розподіл даних по пристроях зберігання порушується. Деякі пристрої зберігання виявляються перевантаженими, а деякі, нещодавно підключені, виявляються недовантаженими, що знижує ефективність їх використання. Для збереження високої ефективності використання всіх пристроїв зберігання необхідно підтримувати рівномірність розподілу даних. Для цієї мети проводиться процедура балансування даних.

Обмежена пропускна здатність каналів передачі даних не дозволяє одночасно переміщувати великі обсяги даних між різними пристроями зберігання. Тому кожен пристрій зберігання може одночасно брати участь тільки в одній операції переміщення даних.

Для досягнення максимальної ефективності процедури балансування необхідно скласти оптимальний план міграції даних, який отримують в результаті рішення задачі міграції [1]. Задача міграції даних в масштабованому сховищі є окремим випадком задачі міграції і має кілька критеріїв оптимізації. Перший критерій –час міграції на масштабуючі пристрої. Під масштабуючими пристроями маються на увазі такі пристрої зберігання, які в поточний момент часу підключаються або відключаються від сховища даних. Оптимізація за цим критерієм дозволяє не просто зменшити час підключення або відключення пристроїв, а, відповідно, зменшити витрати на їх користування. Другий критерій - оптимізація часу міграції на інших пристроях.

У цій роботі описується алгоритм вирішення задачі міграції даних з поліноміальною складністю, який можна застосувати в масштабованих розподілених сховищах, і який оптимізований за першим критерієм - часом міграції на масштабуючі пристрої зберігання.

Існуючі алгоритми

На першому етапі процедури балансування визначається оптимальне розміщення даних на пристроях зберігання в поточний момент часу, що вирішується задачею розміщення. Визначивши нове розміщення даних, а також знаючи попереднє розміщення, можна побудувати мультиграф вимог G, який описує переміщення даних [1]. Цей мультиграф є направленим і ациклічним:

$$G = (V, E, P) E \subseteq V \times VE \subseteq V \times VP : E \to \mathbb{N} \forall \upsilon, \omega \in V \Rightarrow P(\upsilon, \omega) = 0, \quad (1)$$

$$\text{якщо} (\upsilon, \omega) \notin \text{E},$$

де V - пристрої зберігання, E –процедуриперeміщення даних між пристроями зберігання, P - вагова функція мультиграфу (вартість переміщення даних).

Завдання міграції полягає в складанні плану переміщення даних між пристроями зберігання згідно мультиграфу вимог (1) за мінімальну кількість кроків. У цій роботі в рамках розв'язання задачі міграції приймається, що пристрої зберігання не мають обмеження за обсягом даних, що зберігаються. З практичної точки зору, це не сильне обмеження, оскільки місткість пристроїв зберігання враховується в задачі розміщення, тобто перед розв'язанням задачі міграції.

У такій постанові задача міграції зводиться до задачі розфарбовування дуг мультиграфу.

Відзначимо, що оскільки пристрій зберігання одночасно може брати участь тільки в одній операції переміщення даних, напрямок передачі даних в мультиграфі вимог не має значення. Отже, направлениймультиграф можна замінити ненаправленим (2), що дозволить спростити рішення задачі міграції.

$$G = (V, E, P) E \subseteq V \times VP : E \to \mathbb{N} \forall \upsilon, \omega \in V \Rightarrow P(\upsilon, \omega) = 0,$$

$$\text{если} (\upsilon, \omega) \notin \text{E} \quad (2)$$

$$\forall \upsilon, \omega \in V \Rightarrow P(\upsilon, \omega) = P(\omega, \upsilon)$$

Алгоритми точного рішення задачі розфарбовування дуг графа вимагають повного перебору і мають експоненціальну обчислювальну складність, тобто не є поліноміальними. Відповідно, задачарозфарбовування дуг графа відноситься до класу NP-складних [2]. Однак, існують апроксимаційні поліноміальні алгоритми розв'язання цієї задачі.

Алгоритм 1 – апроксимаційний алгоритм розфарбовування дуг графа з поліноміальною складністю [3].Цей алгоритм заснований на принципі «розділяй і володарюй». Загальне рішення для всього графа вибудовується з об'єднання рішень длярозфарбованихпідграфів.Обчислювальна складність цьогоалгоритму - O(A*(V+δ)),деA-множинадугграфу;V-множинавершин;δ-деякаконстанта.

Алгоритм 2 – поліноміальний алгоритм розфарбовування дуг дводольногографу з поліноміальною складністю. Розфарбування дуг дводольногографу є окремим випадком завдання розфарбовування дуг, але це завдання не належить до класу NP-складних задач. У статті [4] можна знайти опис оптимального алгоритму розфарбовуваннядуг дводольного графу з обчислювальною складністю O(A*logD), де A –множинадуг; D – максимальний ступінь вершин графу.

Алгоритм 3 – поліноміальний алгоритм міграції даних у розподілених сховищах, оптимізований за часом масштабування. У статті [5] наведений алгоритм 3 міграції даних, який має поліноміальну складність max{O(A*logD), O(A*(V+δ))}, де A - множинадугграфу; V - множина вершин; D – максимальний ступінь вершин графу; δ - деяка константа. Цей алгоритм заснований на поділі графу вимог Gна два підграфи: масштабуючий підграф G_Sта залишковий підграф G_R. План міграції складається послідовним об’єднанням планів міграції розрахованих на цих підграфах, відповідно за алгоритмами 1 та 2.

Рішення задачі міграції даних

Запропонований алгоритм заснований на алгоритмі 3. В якості оптимізації цього алгоритму використано заміну послідовного об'єднання двох планів міграції на їх паралельне об'єднання.

Під паралельним об'єднанняммається на увазі наступне: роздільне розфарбування G_S і G_R (як це робиться в алгоритмі 3), але при складанні загального плану міграції G для кожного кольору в розфарбовуванні G_Sпроводиться пошук такого кольору в G_R, щоб ці кольори можна було об'єднати в один колір в загальному розфарбуванні G.

Враховуючи це запропонований алгоритм розв'язання задачі міграції даних виглядає так:

1. виділити підграф G_S з графу G на основі множини масштабуючих вершин S;
2. виділити підграф G_R з графу G на основі підграфу G_S;
3. використовувати алгоритм 2 для знаходження плану міграції для дводольногографу вимог GS;
4. використовувати алгоритм 1 для знаходження плану міграції для графу вимог G_R;
5. отримати загальний план міграції для графу вимог G шляхом паралельного об'єднання планів міграції для підграфів G_S і G_R:

5.1. для кожного кроку в плані міграції підграфа G_Sзнайти такий крок у плані міграції підграфа G_R, щоб ці кроки можна було об'єднати в один крок в загальному плані міграції.

Задача виділенняпідграфів з графу має лінійну обчислювальну складність O(V). Завдання паралельного об'єднання планів міграції має поліноміальну обчислювальну складність$O(V^2)$. Алгоритм 1 і алгоритм 2 мають поліноміальну обчислювальну складність: O(A*(V+δ)) і O(A*logD), відповідно. Оскільки представлений алгоритм являє собою послідовне об'єднання кроків 1-5, його обчислювальна складність дорівнює складності найбільш витратного кроку: $\max\{O(V), O(A*\log D), O(A*(V+\delta)), O(V^2)\} = \max\{O(A*\log D), O(A*(V+\delta)), O(V^2)\}$. З цього випливає, що алгоритм має поліноміальну складність.

Оцінка ефективності алгоритму

Використовуючи традиційні алгоритми міграції, масштабування розподіленого сховища (підключення і відключення пристроїв зберігання) можна здійснити тільки після повного виконання процедури міграції даних. Це пов'язано з тим, що масштабуючі пристрої зберігання можуть бути задіяні в процедурі міграції даних до самих останніх кроків. Розроблений алгоритм виділяє в мультиграфі G масштабуючий підграф G_S, що містить всі масштабуючіпристрої. Це дозволяє швидше виконати процедуру міграції на цих пристроях. Зворотною стороною алгоритму є необхідність виконання залишкової міграції в підграфі G_R після виконання міграції в G_S і послідовне об'єднання крокі23,в міграції, яке збільшує загальний час міграції.

Для оцінки ефективності запропонованого алгоритму було проведено його порівняння з алгоритмом 2, якій використовується для побудови планів міграції оптимізованих за другим критерієм (загальним часом міграції на всіх пристроях зберігання).

Під час проведення експериментів на вхід обох алгоритмів подавалися однакові згенеровані за вхідними параметрами мультиграфи вимог G.

В якості вхідних параметрів при експериментах використовувалися:

- загальна кількість пристроїв зберігання,
- кількість масштабуючих пристроїв,
- відносна кількість пересилок даних в мультиграфі вимог від максимально можливої.

Вихідними параметрами алгоритмів є:

- T(G) – загальний час міграції (у кроках) на всіх пристроях за алгоритмом 2,
- $T(G_S)$ – час міграції (у кроках) на масштабуючих пристроях зберігання, розраховується як час міграції у підграфі G_S на кроці 3 запропонованого алгоритму,
- $T(G_R)$ – час міграції (у кроках) на залишкових пристроях зберігання, розраховується як час міграції у підграфі G_Rна кроці 4 запропонованого алгоритму.

Очевидно, що час міграції на масштабуючих пристроях для алгоритму 2 дорівнює загальному часу міграції T(G), а загальний час міграції для запропонованого алгоритму дорівнює сумі часів міграцій для двох підграфів $T(G_S) + T(G_R)$.

Виходячи з критеріїв задачі міграції даних в розподіленому сховищі, будемо використовувати наступні параметри оцінки:

1. $P = T(G) - T(G_S)$–час (у кроках), на який зменшується процес масштабування (додавання або звільнення пристроїв зберігання);

2. $\Delta P = \frac{P}{T(G)} = \frac{T(G) - T(G_S)}{T(G)}$–це відносний виграш у часі масштабування при застосуваннізапропонованого алгоритму в порівнянні з алгоритмом 2;

3. $L = T(G_S) + T(G_R)$–цезагальний час (у кроках) міграції даних за запропонованималгоритмом;

4. $\Delta L = \frac{L}{T(G)} - 1 = \frac{T(G_S) + T(G_R)}{T(G)} - 1$–це відносний програш в загальному часу міграції даних із застосуванням запропонованого алгоритму в порівнянні з алгоритмом 2.

Було проведено чергу експериментів з наступними вхідними параметрами: відносна кількість масштабуючих пристроїв – 5%, відносна кількість пересилок даних – 25%, кількість пристроїв зберігання від 2 до 200 з кроком 1. Для кожного значення кількості пристроїв було проведено 100 експериментів. Остаточні значення вихідних параметрів отримувались усередненням.

Результати експериментів наведені у таблиці 1та на рисунку 1.

З приведених результатів експерименту видно, що для сховищ з відносно невеликою кількістю масштабуючих пристроїв запропонований алгоритм здатний забезпечувати економію часу масштабування у 50–60% в порівнянні с існуючим алгоритмом. При цьому загальний час міграції даних може збільшитися на 20-25%.

Табл. 1. Результати експериментів

№	Кількість пристроїв зберігання V	Кількість масштабуючих пристроїв S	$T(G)$	$T(G_S)$	$T(G_R)$	P	ΔP	L	ΔL
1	10	1	5 , 1	2,99	3,82	2,11	41,37%	6,95	21,42%
2	50	2	20,5	9,71	15,53	10,79	52,63%	25,68	22,55%
3	100	5	37,13	19,06	26,47	18,07	48,67%	46,49	23,51%
4	150	7	53,59	27,71	38,72	25,88	48,29%	66,43	23,58%
5	200	10	68,98	35,71	49,58	33,27	48,23%	85,29	23,64%

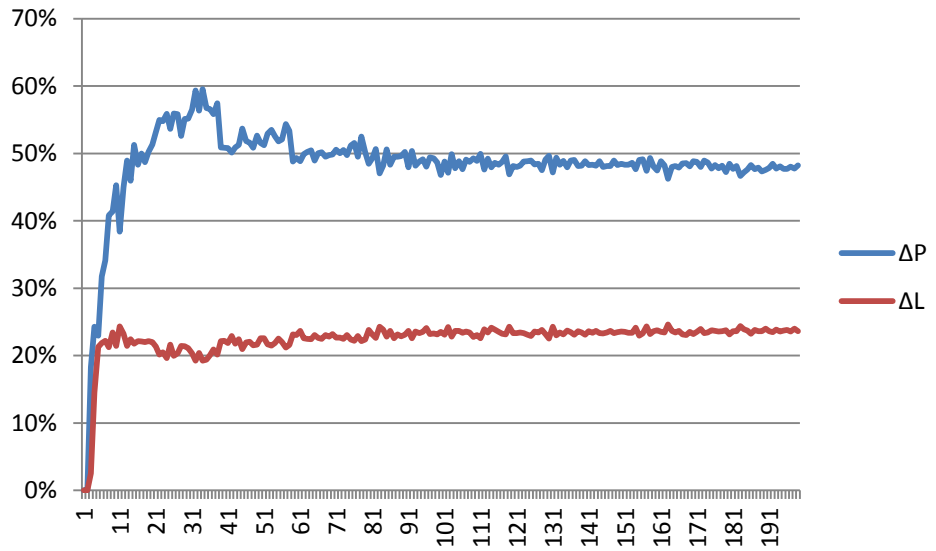


Рис. 1. Графік зміни значень ΔP та ΔL від кількості пристроїв зберігання.

Можливі варіанти подальшої оптимізації алгоритму

Збільшення загального часу міграції на 20-25% для більшості розподілених сховищ не є суттєвою проблемою. Однак, цей ас можна скоротити за допомогою подальшої оптимізації.

Одним з варіантів такої оптимізаціє є заміна запропонованого способу паралельного об'єднання двох планів міграції на інший: розширення дводольного підграфа G_Sвершинами та дугами з підграфа G_Rтак, щоб кількість кроків в плані міграції підграфа G_Sне збільшилась, а кількість кроків у плані міграції підграфа G_Rзменшилась.

Зазначені вище поліпшення вимагають додаткового опрацювання і докладно не розглядаються в цій роботі.

Висновок

В цій роботі було розглянуто проблему рішення задачі міграції даних в розподілених масштабованих сховищах. Було представлено оптимізований за часом міграції на масштабуючі пристрої зберігання алгоритм складання плану міграції даних, який має поліноміальну складність. Цей алгоритм дозволяє зменшити час масштабування сховища, тобто час необхідний на повне підключення або відключення пристроїв зберігання від сховища, за рахунок збільшення загального часу міграції даних.

Перелік посилань

1.Hall J., Hartline J., Karlin A., Saia J., Wilkes J. On Algorithms for Efficient Data Migration // ACM Symposium on Discrete Algorithms. – 2001. – P. 620–629.
2.Holyer I. The NP-completeness of Edge-Coloring // SIAM J. Comp. – 1982. №11. – P. 117–129.
3.Hochbau D. S., Nishizeki T., Shmoys D. B. A better than "Best Possible" algorithm to edge color multigraphs // J. off Algorithms. – 1986. – №7. – P. 79–104.
4.Cole R., Ost K., Schirra S. Edge-coloring bipartite multigraphs in O(E log D) time // Combinatorica. – 2001. – №21. – P. 5–12.
5.Петров Д. Л. Оптимальний алгоритм міграції даних у масштабованих хмарних сховищах // Управління великими системами. – 2010. – №30. – С. 180–197.

УДК 683.519

КИРИЛЕНКО С.Ю.,
МАРКОВСЬКИЙ О.П,

Організація захищеного обчислення модулярної експоненти на віддалених комп'ютерних системах

В статье предложен способ удаленного модулярного экспоненцирования в облачных системах с защитой данных и кода экспоненты. Предложенный способ позволяет разделить процедуру модулярного экспоненцирования на две составляющие, одна, меньшая часть которых выполняется пользователем, а другая, большая - в удаленных мощных вычислительных системах. Подробно описаны математические принципы предлагаемого разделения вычислений, изложена методика организации вычисления. Приведены числовые примеры удаленного защищенного вычисления экспоненты.

In this paper a method for the performing the calculations required for modular exponentiation using remote or distant computational resources. This method is targeted to the paradigm of cloud computing resources and provides security by avoiding the disclosure to the cloud resource, of either the data or the user secret key. The proposed method operates by separating the procedure for modular exponentiation in two components. The first component which is computationally simple is performed on the user

terminal and the second and computationally complex component, is executed on powerful cloud computational resources. The details of the proposed distributed calculation are presented. Hence, the methodology for the organization of the calculations is analyzed. The calculation is illustrated by means of a simple numerical example.

Введение

Опережающее развитие средств телекоммуникаций и глобальных сетей стимулировало появление новой технологии компьютерной обработки информации - облачных вычислений.

Облачные технологии предполагают удаленное предоставление на коммерческой основе любому пользователю компьютерных ресурсов из некоторого пула. При этом, в качестве предоставляемых ресурсов могут выступать вычислительные мощности суперкомпьютеров, программное обеспечение, долговременная память. Выделение ресурса выполняется исходя из наличия свободных ресурсов в планетарном масштабе, так, что пользователь не знает на каких вычислительных мощностях решается его задача или где хранятся его данные. Именно непрозрачность для пользователя процесса предоставления ресурса и обусловило название - облачные технологии.

Использование облачных технологий позволяет разрешить извечную проблему для пользователя ограниченности вычислительных ресурсов: каждому из них может быть предоставлена вычислительная мощность тысяч процессоров и практически неограниченный объем памяти.

Вместе с тем, появление и расширяющееся использование облачных технологий существенно влияет на безопасность компьютерной обработки информации во всех областях человеческой деятельности. Предоставляемые облачными технологиями возможности использования мощных вычислительных ресурсов современных суперкомпьютеров позволяют потенциальным злоумышленникам многократно увеличить эффективность взлома существующих систем защиты информации. Большая часть существующих методов нарушения защиты в той или иной степени использует перебор, который весьма просто распараллеливается и, соответственно, эффективно реализуется на многопроцессорных компьютерных системах, доступ к которым предоставляют облачные технологии.

Таким образом, распространение облачных технологий, делающих доступными для любого пользователя значительные по объему вычислительные ресурсы, которые потенциально могут быть использованы нарушения защиты, имеет следствием снижение надежности существующих механизмов информационной безопасности. Это требует принятия специальных мер для повышения устойчивости протоколов криптографической защиты компьютерной обработки информации во всех сферах человеческой деятельности.

Большая часть современных протоколов защиты информации основана на использовании криптографии с “открытым” ключом. В свою очередь, математической основой большинства алгоритмов с “открытым” ключом является операция модулярного экспоненцирования $A^E \bmod M$. На практике использования криптографии с “открытым” ключом модуль M является частью открытого ключа, E - является компонентой закрытого ключа, а A - представляет собой информационную компоненту. Соответственно, задачей взлома является получение кода E.

Уровень защищенности напрямую зависит от разрядности n используемых в операции модулярного экспоненцирования чисел. В большинстве практических применений, разрядность составляет 1024, 20148 или 4096. Вычислительная сложность модулярного экспоненцирования составляет $O(n2)$ [1]. Это означает, что при удвоении разрядности используемых чисел, объем вычислений возрастает примерно в четыре раза. При этом вычислительная сложность подбора кода экспоненты $O(2n)$.

Как уже отмечалось выше, быстрое распространение облачных технологий компьютерной обработки данных позволяет злоумышленникам сконцентрировать значительные вычислительные мощности для взлома алгоритма, то есть для подбора кода экспоненты E. Фактически это означает снижение в современных условиях криптостойкости систем, в основе которых лежит операция модулярного экспоненцирования. Естественный путь восстановления

криптостойкости систем защиты информации рассматриваемого класса в этих условиях состоит в увеличении разрядности n используемых чисел. Однако, как указывалось выше, увеличение разрядности заметно замедляет вычисления, связанные с функциями защиты информации.

Выходом из создавшейся ситуации может быть использование для модулярного экспоненцирования вычислительных ресурсов облачных систем, но делать это так, чтобы при вычислении AE mod M не были в явном виде использованы секретный код экспоненты E и обрабатываемое число A.

Анализ известных методов защиты удаленной реализации модулярного экспоненционирования

Исследователи в области компьютерных наук уделяют в последние годы пристальное внимание проблеме закрытой реализации удаленных различных видов ресурсоемких вычислений [2].

Основная трудность состоит в том, что не существует универсального подхода, способного защитить данные во время их обработки. Способ защиты существенным образом зависит от операций, используемых при обработке. Иными словами, технологии защиты требуют использования гоморфного по отношению к процедуре удаленной обработки шифрования данных [3].

Поэтому, к настоящему времени предложено большое число различных решений для защищенной удаленной реализации задач разных классов, например, линейной алгебры, обработки изображений [4].

Часть подобных исследований посвящена проблеме защищенной удаленной реализации модулярного экспоненцирования - базовой операции большинства современных протоколов защиты информации.

В частности, в работе [4] предлагается алгоритм удаленного вычисления модулярной экспоненты с проверкой правильности вычислений, выполняемых удаленно. Две модификации удаленного вычисления модулярной экспоненты предложено в работах [3,5]. Основной акцент в этих исследованиях сделан на том, чтобы пользователь мог не только удаленно выполнять модулярное экспоненцирование в закрытом режиме, но и косвенно контролировать правильность выполненных операций.

При этом за рамками внимания авторов остается тот факт, что основным среди критериев эффективности использования удаленных компьютерных мощностей является достижение высокой скорости вычисления. Для этого, необходимо так организовать защищенное удаленное вычисление экспоненты, чтобы можно было использовать основное преимущество удаленных вычислительных систем - наличие значительного числа процессоров, способных работать параллельно [2]. Соответственно, метод защищенной удаленной реализации модулярного экспоненцирования должен прямо организовать возможность параллельного решения этой задачи на многих процессорах.

Целью исследований является разработка метода защищенного выполнения операции модулярного экспоненцирования в облачных системах с возможностью ее распараллеливания.

Метод защищенной параллельной реализации модулярного экспоненцирования

Для постижения поставленной цели предлагается m-разрядный код экспоненты *E={ e0, e1, ..., em-2, em-1},* $\forall i \in \{0,1,..,m\text{-}1\}: ei \in \{0,1\}$, так, что $E = \sum_{i=0}^{m-1} e_i \cdot 2^i$, случайным образом разделить на *h* групп δ0, δ1,...,δh-1 смежных разрядов, содержащих соответственно n0,n1,...,nh-1 двоичных разрядов. При этом *n0+ n1+...+nh-1= m* и группа $\delta 0$ содержит первые n0 двоичных разрядов экспоненты: $\delta_0 = \{e_0, e_1, ..., e_{n_0-1}\}$, вторая группа $\delta 1$ включает следующие *n1* разрядов: $\delta_1 = \{e_{n_0}, e_{n_0+1}, ..., e_{n_1-1}\}$ и так далее. Тогда, разряды группы $\delta 0$ соответствуют числу *g0*, разряды группы $\delta 1$ соответствуют числу *g1*, разряды группы $\delta h\text{-}1$ соответствуют числу *gh-1*:

$$\forall l \in \{0,1,...,h-1\} : g_l = \sum_{j=0}^{n_l-1} e_{n_0+n_1+...+n_{l-1}+j} \cdot 2^j$$

Тогда код экспоненты E может быть представлен в виде суммы:

$$E = g_0 + g_1 \cdot 2^{n_0} + g_2 \cdot 2^{n_0+n_1} + ... + g_{h-1} \cdot 2^{m-n_{h-1}} = \sum_{l=0}^{h-1} g_l \cdot 2^{\sum_{j=0}^{l-1} n_j}$$

Если ввести обозначения w0=1, $w_1 = 2^{n_0}$, $w_2 = 2^{n_0+n_1}$,..., $w_{h-1} = 2^{n_0+n_1+n_2+...+n_{h-2}}$, то экспонента *E* может быть представлена в виде: $E = \sum_{l=0}^{h-1} g_l \cdot w_l$.Тогда *AE mod M* может быть представлено в виде произведения:

$$A^E \bmod M = ((A^{g_0} \bmod M)^{w_0} \cdot (A^{g_1} \bmod M)^{w_1} \cdot ... \cdot (A^{g_{h-1}} \bmod M)^{w_{h-1}}) \bmod M = (\prod_{l=0}^{h-1} (A^{g_l} \bmod M)^{w_l} \bmod M) \bmod M$$

Исходя из изложенного предлагается следующий порядок вычисления модулярной экспоненты *AE mod M*.

1. Пользователь вычисляет

$R_0 = A^{g_0} \bmod M$, $R_1 = A^{g_1} \bmod M$,..., $R_{h-1} = A^{g_{h-1}} \bmod M$.

Практически эти вычисления выполняются по следующему алгоритму:

1.1. Находится максимальная разрядность фрагментов экспоненты $\eta = max\{n0,n1,...,nh-1\}$.
1.2. Определяются начальные значения: индекс j=0; B = A; если e0=0, то R0=1, иначе R0=1, Для всех i=0,1,2,...,h-1: если младший разряд i-той группы $e_{q_i} = 0$, то Ri=1, иначе Ri=B, где qi - порядковый номер разряда экспоненты E, с которого начинается i-тая группа gi: q0 = 0, q1 =n0, q2 =n0+n1, q3=n0+n1+n2,..., qh-1 = n-nh-1.
1.3. Вычисляется B = A·B mod M, индекс j увеличивается на единицу: j=j+1. Для всех i=0,1,2,...,h-1: если j-тый разряд i-той группы равен единице: $e_{q_i+j} = 1$, то Ri = Ri·B mod M.
1.4. Если j<η, то возврат на п.1.3.

2. Значения R1,...,Rh-1 и w1,...,wh-1 отсылаются в облако.
3. В облаке параллельно вычисляются $D_1 = R_1^{w_1} \bmod M$, $D_2 = R_2^{w_2} \bmod M$ ..., $D_{h-1} = R_{h-1}^{w_{h-1}} \bmod M$ и вычисленные значения возвращаются пользователю.
4. Пользователь вычисляет

$$A^E \bmod M = (R_0 \cdot \prod_{i=1}^{h-1} D_i \bmod M) \bmod M$$

Предложенный способ удаленного вычисления экспоненты может быть иллюстрирован следующим примером для малых разрядностей *A*, *E* и *M*.

Пусть значения параметров *M*=719, *A*=596. *E*= 71310 = 10110010012. *AE mod M* = 596713*mod* 719 = 83. Пусть, 10-разрядный код экспоненты E делится на три фрагмента: 5-разрядный, двухразрядный и 3-хразрядный: *n0* = 5, n1=2, n2 = 3 δ0={1,0,0,1,0} δ1={0,1}, δ2={1,0,1}. Тогда *g*0=9, *g1*=2, *g*2=5, *w1*=25 = 32, *w*2= 25+2=128. $\eta = max\{5,2,3\}=5$.

Порядок вычисления значений *R0,R1 и R2* по описанному алгоритму иллюстрируется таблицей 1.

Таблица 1. Вычисление компонент R0,R1 и R2

Индекс *j*	***B***	R_0	R_1	R_2
1	A = 596	596	1	596
2	A^2 mod 719 = 30	596	30	596
3	A^4 mod 719 =181	596	30	26
4	A^8 mod 719 =406	392	30	26
5	A^{16} mod 719 = 185	392	30	26

Вычисленные значения *R1*=30 *R2*=26 а также значения *w1*= 32, *w2*= 128 отсылаются в облако. Там вычисляются значения *D1* = 3032 *mod* 719 = 403 и *D2* = 26128 *mod* 719 = 130 и возвращаются пользователю, который вычисляет значение экспоненты в виде: *AE mod M* = (*R0·D1·D2*) *mod M* = (392·403·130) *mod* 719 = 83.

Оценка эффективности

При вычислении экспоненты AE mod M по классическому алгоритму [1] выполняется m шагов, на каждом из которых выполняется модулярное возведение в квадрат и умножение на A, если текущий двоичный разряд экспоненты равен единице. Соответственно, среднее число *N0* операций модулярного умножения составляет *N0= 1.5·m*.

При вычислении пользователем *h* кодов *R0, R1, ..., Rh-1* по предложенному способу, средняя длина разрядность каждого из фрагментов равна *m/h* и можно считать также, что η≈ m/h. Предложенный способ предполагает выполнение η операций модулярного возведения в квадрат и, в среднем, *0.5·m/h-1* операций модулярного умножения для каждой из групп разрядов экспоненты. Таким образом, среднее число *N* операций модулярного умножения, выполняемых пользователем составляет:

$$N = \frac{m}{h} + h \cdot (\frac{m}{h \cdot 2} - 1) = \frac{m}{2} + \frac{m}{h} - h \approx \frac{m}{2} .$$

Следовательно, количество операций модулярного умножения, выполняемых пользователем уменьшается в *N0/N≈3* приблизительно в три раза за счет того, что остальные операции выполняются в облаке параллельно.

При наличии у пользователя *h* процессорных ядер вычисление кодов *R0, R1..., Rh-1* может выполняться параллельно. В этом случае, число *N1* операций модулярного умножения, которые лежат на критическом пути составляет *2·m/h.* Соответственно, количество операций модулярного умножения по сравнению с прямым вычислением экспоненты уменьшается в *N0/N1* ≈ 0.75·h раз.

Вместе с тем, по передаваемым в облако кодам *R1, R2,...,Rh-1* практически сложно выполнить восстановление секретных кодов экспоненты *E* и числа *A*.

Заключение

В результате проведенных исследований предложен простой и технологичный способ ускорения вычисления базовой операции большинства протоколов защиты информации - модулярного экспоненцирования за счет использования значительных вычислительных ресурсов, предоставляемых облачными технологиями.

Способ предполагает выполнение части вычислений непосредственно пользователем, а части - на удаленных процессорных средствах. При удаленной обработке обеспечивается защита данных и секретных компонент ключа пользователя.

Теоретически и экспериментально доказано, что объем вычислений, выполняемых пользователем сокращается примерно в три раза. Это позволяет существенно ускорить реализацию криптографических преобразований при большой длине ключа за счет использования мощных процессорных средств, предоставляемых современными облачными технологиями.

Разработанный способ ориентирован на применение пользователем портативных мобильных вычислительных устройств малой производительности, подключенных к сети Интернет.Список литературы

1. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography.- CRC Press.- 1996. - 780 p.
2. Boroujerdi N., Nazem S. Cloud Computing: Changing Cogitation about Computing // IJCSI International Journal of Computer Science Issues, - Vol. 9, - Issue 4. -2012.- No 3.- PP. 169-180.
3. Xiaofeng Chen1, Jin Li, Jianfeng Ma3, Qiang Tang, Wenjing Lou. New Algorithms for Secure Outsourcing of Modular Exponentiations // ESORICS 2012, LNCS 7459, - 2012.- PP. 541–556.
4. Hohenberger, S., Lysyanskaya, A.: How to Securely Outsource Cryptographic Computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005),
5. Markovskyi O.P. Secure Modular Exponentiation in Cloud Systems./ Oleksandr P. Markovskyi, Nikolaos Bardis, Nikolaos Doukas, Sergej Kirilenko // Proceedings of The Congress on Information Technology, Computational and Experimental Physics (CITCEP 2015), 18-20 December 2015, Krakow, Poland, C. 266-269.

УДК 004

КОСТЕНКО В.А.
ГОРДІЄНКО Ю.Г.

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБРОБКИ ДАНИХ ВЕЛИКОГО ОБСЯГУ В КОНТЕКСТІ ІНТЕРНЕТУ РЕЧЕЙ

Інтернет речей здійснив великий вплив на розвиток даних великого обсягу. Основою революційної ідею інтернету речей є те, що майже кожен об'єкт або пристрій буде мати IP- адресу і буде з'єднаний один з одним. Тепер, з огляду на той факт, що мільйони пристроїв буде підключено до мережі і вони будуть генерувати величезні обсяги даних, ефективність механізму збору та обробки даних буде знижуватися. Запропонований підхід, який покликаний підвищити ефективність обробки даних із у мережах із використанням інтернету речей.

Internet of things (IoT) make huge impact on Big Data concept. The main revolutionary idea of IoT is almost every object or device have IP address and is connected to the Internet. Now, considering fact that millions of devices will be connected and will generate enormous volumes of data. That is why contemporary data collection mechanisms will lose their performance and efficiency. In current article proposed method for increasing Big Data proceeding performance in IoT networks.

Ключові слова: дані великого обсягу, інтернет речей.

Вступ

Наступна хвиля у розвитку інформаційних технологій буде знаходитися поза сферою традиційного персонального комп’ютера [1]. Інтернет речей - парадигма, яка полягає у тому, що оточуючі нас об’єкти будуть під’єднанні до мережі Інтернет. Радіочастотна ідентифікація і датчики мережевих технології будуть розвиватися, щоб впоратися з новим викликами, де інформаційні і комунікаційні системи будуть повсюдно інтегровані у навколишнє середовище.

Це призведе до утворення величезних обсягів даних, які повинні бути збережені, оброблені і представлені в доступній формі для легкої інтерпретації. Ця модель буде складатися з послуг, які є товарами і поставляється у вигляді, подібному до традиційних товарів. Хмарні обчислення можуть надати віртуальну інфраструктуру для обчислювальної мережі, яка об'єднує пристрої контролю, пристрої зберігання даних, аналітичні інструменти, платформу візуалізації і доставки клієнту.

Приклад використанні інтернету речей в медицині

Неінвазивні системи моніторингу, створенні на основі інтернету речей, використовуються для госпіталізованих пацієнтів, фізіологічний стан яких вимагає постійної пильної уваги [2]. Ці системи моніторингу використовують датчики для збору фізіологічної інформації, яка аналізується і зберігається за допомогою шлюзів у хмарних інфраструктурах. Після чого ця інформація надсилається по бездротовому каналу до осіб, які забезпечують догляд для подальшого аналізу і розгляду, отже, медичним фахівцям уже не потрібно буде постійно спостерігати за станом хворого при безпосередньому контакті. Замість цього, він забезпечується безперервний автоматизований потік інформації. Таким чином, якість медичного обслуговування поліпшується за рахунок постійної уваги, що в свою чергу, знижує вартість догляду та позбавляє від необхідності лікарів активно брати участь в зборі та аналізі даних.

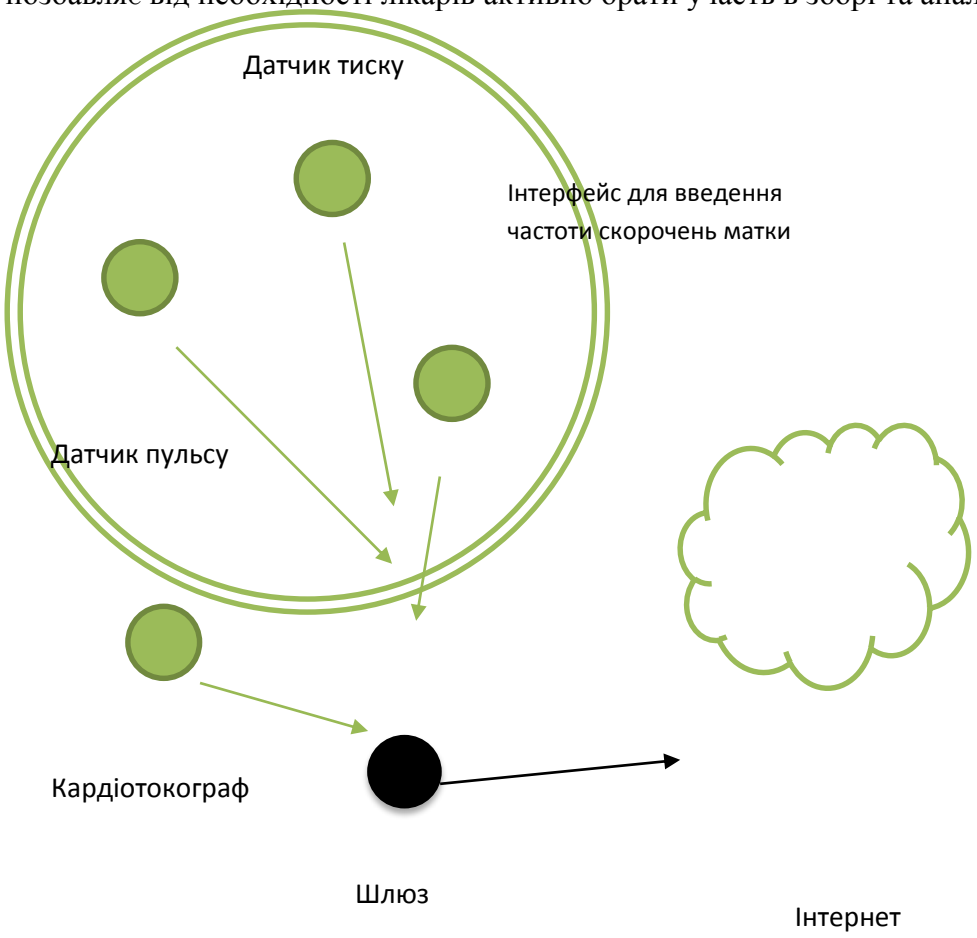


Рисунок 1. – Схема з’єднання датчиків та шлюзу із мережею інтернет

У сучасних пологових будинках використовується практика при якій під час пологів у пологовому залі можуть бути присутні чоловік і/або родичі жінки, що народжує. Для цього кожна майбутня мати розташовується у окремій палаті. Але одночасно лікар може приймати пологи у декількох жінок. Тому йому необхідно завжди володіти інформацією про стан пологів

кожної жінки для ефективного визначення пріоритетів і вчасного надання допомоги. Тому у пологовому залі для оцінки стану плоду і жінки, що народжує оцінюють такі показники:

1. Частоту пульсу жінки.
2. Артеріальний тиск жінки.
3. Частоту скорочення матки.
4. Кардіотокограма на якій аналізують такі показники як: базальна часто уд/хв, амплітуда осциляцій уд/хв, реакція на перейми [3].

Перші три показники оцінюють стан жінки, тоді як четвертий, кардіотокограма, оцінює стан плоду. Це дає лікарям картину стану плоду за якою вони можуть приймати рішення про кесаревого розтину.

Зараз усі ці показники у т.ч. графік кардіотокорами аналізуються вручну. Тому залишається актуальним завдання автоматизації даного процесу. Його зручно провести із використанням концепції Інтернету речей. Пристрої стеження за станом жінки і шлюз, який буде з'єднано із мережею Інтернет представлено на рис. 1.

У якості шлюзу можна використовувати пристрої на основі платформи Arduino. Зокрема, Arduino MKR1000, який володіє рядом переваг:

- Зручне середовище розробки програмного забезпечення.
- Можливість створення інтерфейсів для роботи із різними типами пристроїв (датчиків пульсу, токографа і т.д.)
- Низька вартість.
- Низьке енергоспоживання.
- Наявність вбудованого Wi-Fi модуля.
- Компактність.

На обчислювальних пристроях, що виступають у якості шлюзу можна виконувати оброку даних, необхідну для встановлення точного діагнозу і тим самим збільшити ефективність медичного персоналу під час процесу пологів.

Висновок

Даний підхід дасть доступ до великого обсягу даних, які супроводжують процес пологів. Також знизиться навантаження на серверну частину системи за допомогою перенесенню частини обчислень на клієнтську часину. Завдяки комбінації двох потужних технологій обробки даних великого обсягу та інтернету речей створить можливість для зберігання та обробки значних масивів даних. Аналіз даних великого обсягу у майбутньому дасть змогу не тільки спостерігати за процесами, але й передбачати наслідки.

Перелік посилань

1.Jayavardhana G. Internet of Things (IoT): A vision, architectural elements, and future directions / G. Jayavardhana, B. Rajkumar. // Future Generation Computer Systems. – 2013. – №7. – С. 1645–1660.
2.Kulkarni A. Healthcare applications of the Internet of Things: A Review / A. Kulkarni, S. Sampada. // International Journal of Computer Science and Information Technologies. – 2014. – №5. – С. 6229–6232.
3.Воскресенський С.В. Оценка состояния плода / Воскресенський С.В.. – Минск: Книжный дом, 2004. – 302

УДК 51.004

КОСТЕРНИЙ О.В.

МЕТОД PARALLAX OCCLUSION MAPPING

В даній статті буде розглянутий метод Parallax occlusion mapping. Parallax occlusion mapping — програмна техніка (методика) у тривимірній комп'ютерній графіці, удосконалений варіант

техніки «parallax mapping». Parallax occlusion mapping використовується для процедурного створення тривимірного опису текстурованої поверхні з використанням карт зміщення замість безпосереднього генерування нової геометрії.[1] Методику «Parallax occlusion mapping» умовно можна назвати «2.5D», тому що вона дозволяє додавати тривимірну складність у текстури, не створюючи при цьому реальні тривимірні графічні структури.

This article will consider the method Parallax occlusion mapping. Parallax occlusion mapping - programming technique (method) in a three-dimensional computer graphics, improved version of technology «parallax mapping». Parallax occlusion mapping is used to create a three-dimensional description of the procedural textured surface using displacement maps instead of directly generating new geometry. [1] The technique «Parallax occlusion mapping» can be called «2.5D», because it allows you to add complexity in a three-dimensional texture without creating thus the real three-dimensional image structures.

Вступ

Parallax occlusion mapping[1] - це метод, який зменшує складність геометричної моделі шляхом кодування інформацію деталей поверхні в текстуру. Інформація поверхні, яка зазвичай використовується, - це карта висот, представлення заміненої геометрії. Коли модель візуалізується, деталі поверхні відновлюються в піксельному шейдері з інформації текстури карти висот.

Огляд алгоритму

Так що ж таке parallax occlusion mapping? Перш за все, давайте подивимося на зображення стандартної поверхні, до якої ми б хотіли застосувати нашу техніку. Припустимо, що ця многокутну поверхня являє собою куб, що складається з шести граней, з двох трикутників кожний, в цілому дванадцять трикутників. Ми встановимо координати текстури з кожної вершини таким чином, що кожна грань куба буде включати в себе повну копію даної текстури. На малюнку 1 показана ця проста многокутна поверхня, з нормальним відображенням, яка використовується для забезпечення простого дифузного освітлення.

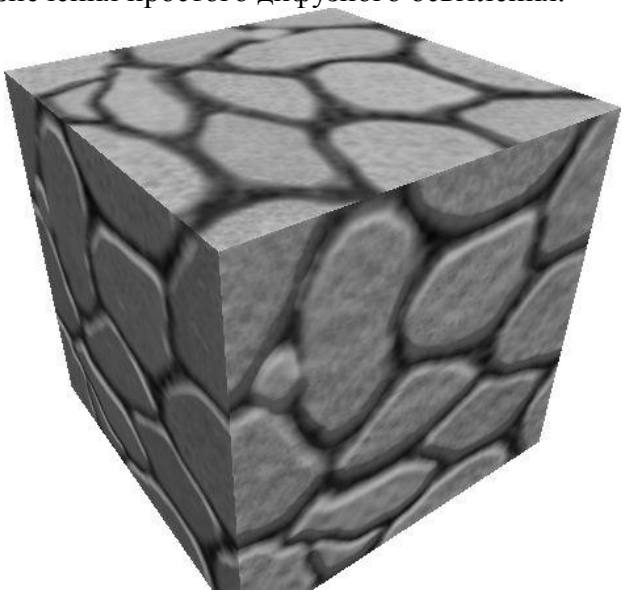


Рис. 1: Плоска поверхня

Основна ідея parallax occlusion mapping є відносно простою. Для кожного пікселя багатокутної поверхні, ми моделюємо складну об'ємну форму. Ця форма представлена картою висот[2], закодовану в текстуру, яка наноситься на поверхню. Карта висот в основному додає компоненту глибини до плоскої поверхні. На малюнку 2 представлені результати моделювання цієї поверхні з картою висот на тестовому кубі.

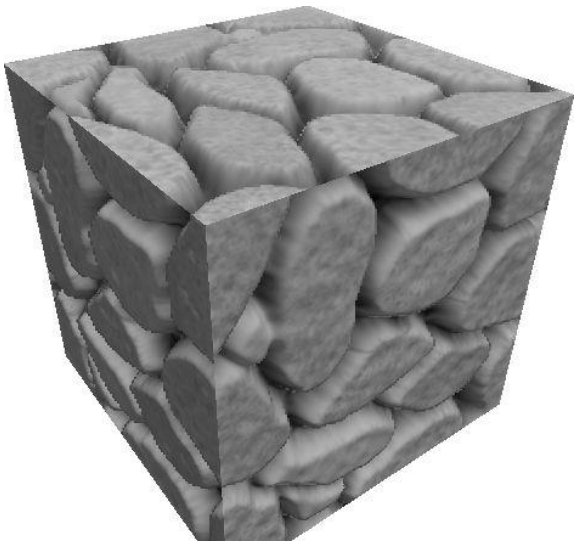


Рис. 2: Плоска поверхня, яка апроксимує об'ємну форму

Модифікація положення поверхні може бути візуалізована більш чітко, якщо спроектувати сітку на моделюємий об'єм. Це показує різні контури, які створюються шляхом зміни текстурних координат поверхні. Малюнок 3 демонструє такий шаблон контуру.

Будемо вважати, що карта висот буде варіюватися в значення від [0.0,1.0], де значення 1,0, представляє полігональну поверхню, а 0.0 представляє найглибшу можливу позицію об'ємної форми. Для того, щоб мати можливість правильно відтворити об'ємну форму, представлену картою висот, напрямок «погляду» повинен бути використаний в поєднанні з даними карти висот для обчислення, які частини поверхні будуть видні на кожному екранному пікселі поверхні для даного напрямку «погляду».

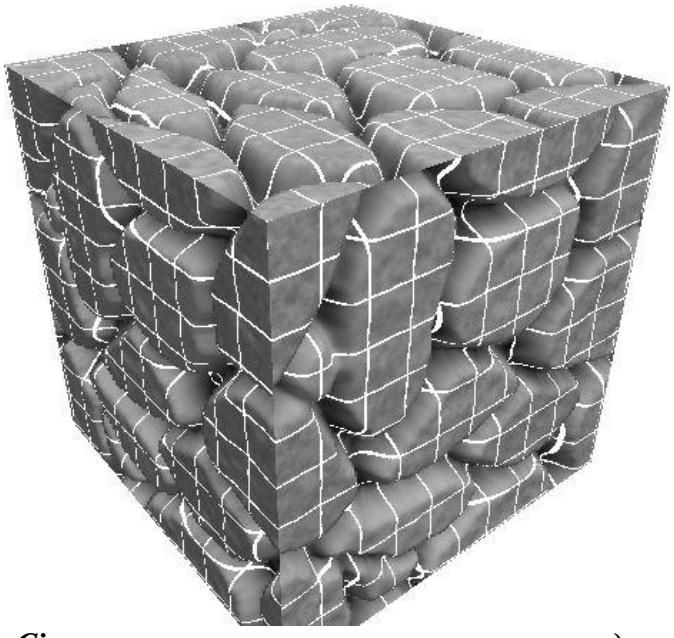


Рис. 3: Сітка, спроектована на поверхню, що моделюється

Це досягається за рахунок використання спрощеного алгоритму ray-tracing[3] (трасування променями) в піксельному шейдері. Луч, який ми будемо трасувати, формується з вектора розташування ока (або камери) до поточного растерізуємого пікселя. Уявіть собі цей вектор, що проколює поверхню, і подорожує далі, поки не досягне нижньої частини віртуального об'єму. На малюнку 4 показаний бічний профіль цього перетину.

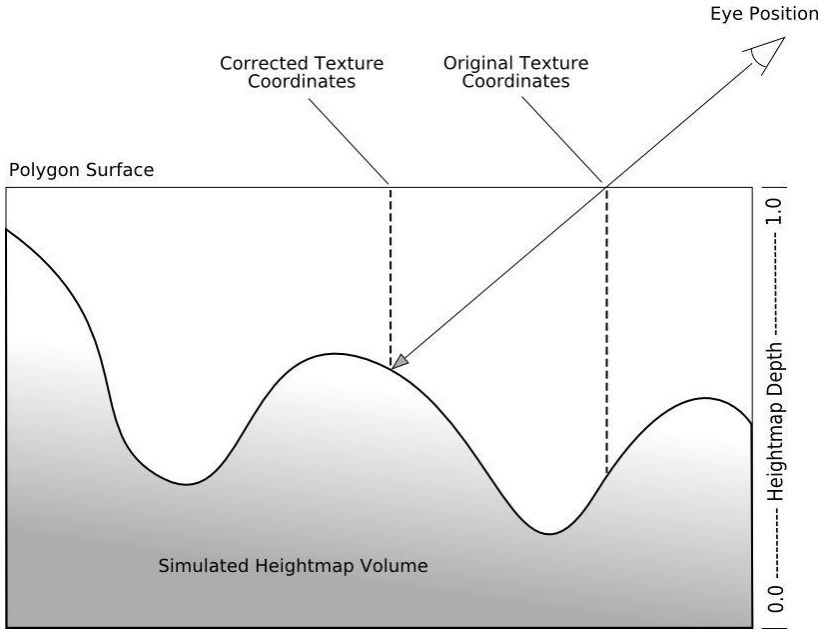


Рис. 4: Вектор «погляду», що перетинається з віртуальним об'ємом

Відрізок від багатокутної поверхні до нижньої частини віртуального об'єму являє собою лінію «видимості» для нашої поверхні. Завдання полягає в тому, щоб з'ясувати першу точку на цьому сегменті, який перетинається з нашою картою висот. Ця точка – це те, що буде бачити «глядач», якщо ми візуалізуємо повно геометричну модель нашої поверхні з картою висот.

Так як точка перетину між нашим відрізком і поверхнею карти висот являє собою видиму точку поверхні в цьому пікселі, вона також неявно описує скориговані зміщені текстурні координати, які повинні бути використані для дифузного кольору, карти нормалей, або будь-якої іншої текстури, яку ви використовуєте для освітлення поверхні[4]. Якщо ця корекція здійснюється на всіх пікселах поверхні, то загальним ефектом буде реконструйована об'ємна поверхня.

Висновок

На простому прикладі був розглянутий метод Parallax occlusion mapping, який зменшує складність геометричної моделі шляхом кодування інформацію деталей поверхні в текстуру. А візуалізована поверхня відновлюється з точок пересікання вектору «погляду» та уявної поверхні, яка утворюється з карти висот.

Перелік посилань

1. Brawley, Z., and Tatarchuk, N. 2004. Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing. In ShaderX3: Advanced Rendering with DirectX and OpenGL, Engel, W., Ed., Charles River Media, pp. 135-154. http://books.google.ca/books?id=DgMSb_10l7IC&pg=PA135&dq=parallax+occlusion

2. Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows - Tatarchuk http://ati.amd.com/developer/techreports/2006/I3D2006/Tatarchuk-POM-SI3D06.pdf

3. Kaneko, T., et al, 2001. Detailed Shape Representation with Parallax Mapping. In Proceedings of ICAT 2001, pp. 205—208.

4. Наталия Татарчук, 2005. Practical Dynamic Parallax Occlusion Mapping презентация на SIGGRAPH

УДК 004.771

КУРІЛО С.А.
КУЛАКОВ Ю.О.

СПОСІБ БАГАТОШЛЯХОВОЇ МАРШРУТИЗАЦІЇ В МЕРЕЖАХ GMPLSНА ОСНОВІ VPN

У роботі розглянуто існуючі протоколи безпечної маршрутизації в мережах VPNта способи їх побудови, а також основні переваги технологіїGMPLS. Запропоновано алгоритм зустрічної хвилі для пошуку непересічних шляхів у мережі, що дозволить забезпечити безпечну передачу інформаційних повідомлень та рівномірно завантажити канали зв'язку.

Ключові слова: GMPLS, VPN, багатошляхова маршрутизація.

In the article are considered the existing secure routing protocols in the VPN networks and methods of their construction, and the main benefits of GMPLS technology. Propose the algorithm of the meeting waves to find disjoint paths in the network, which will provide secure transfer of information messages and evenly loadchannels.

Keywords: GMPLS, VPN, multipath routing.

Вступ.В даний час при організації GMPLS (Generalized Multi-Protocol Label Switching)мереж широко використовуються віртуальні приватні мережі (Virtualprivatenetwork, VPN)[1, 2]. Вузли VPN з'єднуються між собою за допомогою віртуальних каналів або тунелів, що представляють собою послідовність каналів передачі данихмережі. Мінімальною кількістю віртуальнихканалів $k = N - 1$, де N– кількість вершин, характеризується VPN з топологією мінімального покриваючого дерева. Найбільш універсальною є повнозв'язна топологія VPN з кількістю каналів $k = N \cdot (N - 1)$.При організації передачі інформації в розподілених комп'ютерних системах широко використовується багатошляхова маршрутизація, при якій кілька фізичних каналів об'єднуються в одинбагатоканальний віртуальний шлях. Це дозволяє підвищити ефективність процедури конструювання трафіку, забезпечитизадані параметри якості сервісу (QualityofService, QoS) при зміні параметрів системи передачі інформації, зокрема при зміні метрики каналів, шляхомзаміни одного фізичного каналу іншим врамках одного багатоканального віртуальногошляху.[2]

Багатошляхова маршрутизація є однією з найбільш важливих напрямків вобласті маршрутизації. Дана маршрутизація заснована на одношляховіймаршрутизації між вузлом джерелом і вузломпризначення, де з великою ймовірністювибирається шлях з мінімальною вартістю, хоча по різних цінових показниках можуть бути різні шляхи. Таким чином, в сильнозв'язаній мережі можеіснувати кілька шляхів між пароювузлів джерела і призначення. Сенс багатошляхової маршрутизації полягає в тому,щоб надати вузлу джерела вибіродного з декількох маршрутів в будь-якийчас до конкретного вузла призначення, або виконати пересилання частини інформаційного повідомлення по всім шляхам для забезпечення безпеки передачі даних, використовуючи переваги надлишкової зв'язаності мережі. Кілька шляхів можутьвикористовуватися як по черзі (трафікпроходить по одному із шляхів за одиницю часу),так і одночасно (трафік проходить одночасно по декількох шляхам).[3]

Так як використання багатошляхової маршрутизації для передачі інформації дозволяє знизити ризик перехоплення повідомлення, що свідчить про безпечність мережі, то задача формування множини непересічних шляхів в мережах GMPLSє актуальною і потребує нових рішень.

Огляд існуючих рішень

За типом використовуваного середовища VPNподіляються на захищені та довірчі. Захищені мережі є найпоширенішим варіантом віртуальних приватних мереж. Вони дозволяють створити

надійну і захищену підмережу на основі ненадійної мережі, зазвичай, Інтернету. Прикладом захищених протоколів VPN є: IPSec, SSL та PPTP. Довірчі мережі використовують у випадках, коли середовище передачі даних можна вважати надійним і необхідно вирішити лише завдання створення віртуальної підмережі в рамках великої мережі, а питання забезпечення безпеки стають неактуальними. Прикладами подібних VPN рішень є: Multi-ProtocolLabelSwitching (MPLS) і L2TP (Layer 2 TunnellingProtocol).

Конфігурація і характеристики VPN багато в чому визначаються типом VPN-пристроїв. За способом технічної реалізації розрізняють наступні групи VPN:

- VPN на основі маршрутизаторів;
- VPN на основі брандмауерів;
- VPN на основі програмних рішень;
- VPN на основі мережевих ОС;
- VPNна основі спеціальних апаратних засобів з вмонтованими шифропроцесорами.

VPN на основі маршрутизаторів передбачає застосування маршрутизаторів для створення захищених каналів. Крім простого шифрування інформації, що проходить, маршрутизатор може підтримуватий інші функції VPN, такі як ідентифікація при встановленні тунельного з'єднання і обмін ключами. При цьому тунель може бути встановлений, ґрунтуючись на адресах джерела і приймача, номера порту TCP (UDP) і зазначеної якості сервісу. [**Ошибка! Неизвестный аргумент ключа.**]

Всі реалізації VPN на основі брандмауерів засновані на тому, що трафік, що проходить через брандмауер, автоматично в ньому і шифрується. До програмного забезпечення брандмауера додається модуль шифрування.

Наступним підходом до побудови VPN є програмні рішення. При реалізації такого рішення використовується спеціалізоване програмне забезпечення, яке працює на виділеному комп'ютері і в більшості випадків виконує роль proxy-сервера. Шифрування здійснюється на базі 56 або 128 бітових ключів Rivest-Cipher 4, отриманих в процесі встановлення з'єднання. Далі зашифровані пакети інкапсулюються в інші IP-пакети, які в свою чергу відправляються на сервер. У ході роботи Tunnel здійснює перевірку цілісності даних за алгоритмом MD5. Крім того, дане ПЗ кожні 30 хвилин генерує нові ключі, що значно підвищує захищеність з'єднання.

На базі будь-якої сучасної ОС можливо побудувати мережу VPN. Якщо говорити про сімейство ОС від компанії Microsoft, то для створення VPN Microsoft використовує протокол PPTP. При підключенні до PPTP-серверу користувач проходить процес аутентифікації по протоколах PAP, CHAP або MS-CHAP. Передані пакети інкапсулюються в пакети GRE/PPTP. Аналогічні засоби присутні і в операційних системах на базі Linux та MacOS.

Головною перевагою VPN на основі спеціальних апаратних засобів є їх висока продуктивність. Більш висока швидкодія спеціалізованих VPN-систем обумовлена тим, що шифрування в них здійснюється спеціалізованими мікросхемами. Спеціалізовані VPN-пристрої забезпечують високий рівень безпеки, проте мають велику вартість.

За типом протоколу існують реалізації віртуальних приватних мереж під TCP/IP, IPX і AppleTalk. На сьогоднішній день спостерігається тенденція до загального переходу на протокол TCP/IP, і абсолютна більшість VPN рішень підтримує саме його. Адресація в ньому найчастіше вибирається згідно зі стандартом RFC 5735 з діапазону приватних мереж TCP/IP. За рівнем мережевого протоколу на основі зіставлення з рівнями еталонної мережевої моделі ISO/OSI до VPN належать:IPSec (IP security); PPTP (Point-to-Point Tunneling Protocol); PPPoE (PPP (Point-to-Point Protocol) over Ethernet); L2TP (Layer 2 Tunnelling Protocol); L2TPv3 (Layer 2 Tunnelling Protocol ver. 3); OpenVPN SSL –VPN з відкритим вихідним кодом, підтримує режими PPP, bridge, point-to-point, multi-client server.

Найбільш універсальним способом побудови VPN є використання технології інкапсуляції, або тунелювання. У загальному випадку тунелювання застосовується для того, щоб передавати пакети однієї мережі (первинної) по каналах зв'язку іншої (вторинної), протоколи яких не сумісні. Для цього пакет первинної мережі (дані та протоколи) інкапсулюються в пакет вторинної мережі

у вигляді даних. Таким чином, пакет просувається маршрутизаторами ядра мережі тільки на підставі зовнішнього заголовка, без інспекції вмісту оригінального пакета.[5]

У технології GMPLS використовується розподілений принцип організації управління, що дозволяє їй швидко реагувати на зміни в мережі. Однак через це GMPLS має більш обмежені можливості в накладеній мережі. Таким чином періодична переоптімізація з використанням централізованого набору засобів може вимагати гарантій, що мережа не буде занадто відхилятися від її оптимального стану.

У контексті віртуальних приватних мереж, GMPLS допускає існування оптичних VPN, які забезпечують з'єднання між різними ділянками VPN за допомогою оптичних з'єднань. Використовуючи оптичні VPN, клієнтські пристрої отримують більш функціональне і деталізоване управління над їх віртуальної мережею. Вони можуть більш легко адаптувати смугу пропускання і політику функціонування до їхніх вимог, використовуючи властивості самоналаштування (тобто без втручання з боку оператора).Набір протоколів GMPLS реалізують механізми, необхідні для задоволення вимог наступного покоління оптичних транспортних мереж. Шляхом узагальнення широко відомої концепції MPLS-TE, набір протоколів GMPLS обслуговує єдину площину управління, що функціонує на всій оптичній мережі, автоматизованим, гнучким і ефективним чином.

GMPLS надає необхідний рівень автоматизації, що, в свою чергу, дозволяє швидко і ефективно надавати клієнтам широкосмугові послуги. Завдяки реалізації засобів для автоматизації можливості обслуговування як пакетних, так і оптичних мереж. Набір протоколів GMPLS призначений для реалізації розподіленої площини управління, так як єдина площина управління не вносить побічних ефектів у всю магістральну мережу, це забезпечує гнучкість і масштабованість, оскільки подальше додавання GMPLS-сумісних вузлів і устаткування в існуючу мережу не викликає складних операцій по конфігурації або обслуговуванню. Це вигідно відрізняється від вживаних в даний час ручних методів. Технологія VPN відповідає основним критеріям збереження інформації: цілісності, конфіденційності, авторизації доступу. Такерішення гарне тим, що при мінімальних витратах і високому ступені захисту можна отримати працездатний загальний простір, об’єднавши декілька локальних мереж в загальну VPN мережу. Весь процес віртуалізації мережі проходить абсолютно непомітно для користувача і нічим не відрізняється від роботи в локальній мережі.До недоліків технології VPN, в розглянутих реалізаціях, можна віднести необхідність закупівлі великої кількості устаткування і програмного забезпечення, а також збільшення об'ємів зовнішнього трафіку. Втім ці витрати досить невеликі, враховуючи величезну кількість переваг VPN.

GMPLS еволюціонувала від MPLS шляхом розширення існуючої парадигми комутації по мітках від технологій комутації пакетів/фреймів до технологій, орієнтованих на встановленні з'єднання. Фундаментальна концепція GMPLS застосована для усунення недоліків існуючих технологій для багаторівневих мереж. Завдяки оптимізованому розподілу ресурсів відновлення, і ефективним (багаторівневим) механізмам відновлення можна знизити капітальні витрати. До того ж, автоматичне відновлення –з виключенням необхідності в ручних втручаннях – може знизити експлуатаційні витрати та кількість помилок. [2]

Отже, на основі існуючих рішень з ціллю підвищення ефективності та продуктивності передачі даних мережею VPN,було вирішено розробити спосіб багатошляхової маршрутизації з використанням GMPLS,як засіб транспортування зі швидкою комутацією пакетів, так як ця технологія гнучка та легка в експлуатації і не потребує дорогих витрат.

Опис способу багатошляхової маршрутизації в мережах GMPLS

Для визначення множини шляхів мережі на початковому етапі необхідно визначити число непересічних маршрутів з метою оптимізації процедури формування та знаходження кількості ідеально непересічних шляхів в заданій мережі.Задача визначення максимально можливого числа непересічних маршрутів між відправником та одержувачем зводиться до задачі знаходження мінімальної розділяючої множини між двома вершинами.

Під мінімальною розділяючою множиною мається на увазі мінімальна підмножина вершин графа, видалення яких призводить до поділу графа на два підграфи. В теорії графів таку мінімальну підмножину вершин прийнято називати підмножиною зчленування зв'язаного неорієнтованого графа. Потужність даної підмножини визначає максимальну кількість непересічних шляхів між двома вершинами, розташованими в суміжних підграфах. Стосовно до даної задачі спосіб визначення мінімальної розділяючої множини полягає в наступному: відбувається розбиття графа G (рис. 1) на два підграфи, перший підграф GS містить лише вершину S відправника інформації, а другий GD включає вершину D одержувача інформації та інші вершини.

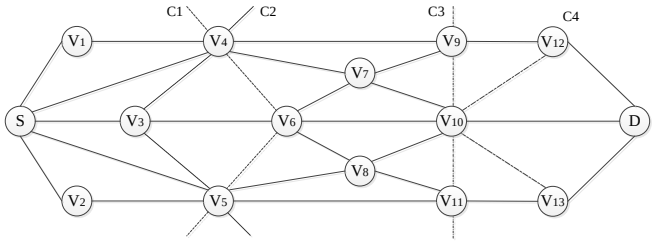


Рисунок1. Мінімальні розділяючі множини графа

На початковому етапі множина вершин VS підграфа GS включає тільки одну вершину S, тобто VS = {S}. Після цього формуємо мінімальну розділяючу множинуVmin з вершин, пов'язаних з вершинами підмножини VS, тобто з вершиною S. В даному випадку такими вершинами є: V_1, V_2, V_3, V_4, V_5.Вершини S, V_1, V_2 є внутрішніми вершинами підграфа GS, так як не пов'язані з вершинами підграфа GD. Вершини V_3, V_4, V_5 підграфа GS пов'язані з вершинами підграфа GD і утворюють мінімальну розділяючумножинуVmin = {V_3, V_4, V_5} між підграфом GS і GD. На наступному кроці розглянемо вершини підмножини Vmin = {V_3, V_4, V_5}. Вершина V_4 пов’язана з двома вершинами V_7 і V_9 підграфа GD. У цьому випадку підмножина зчленування VSDпідграфів GS і GD не є мінімальною, так як утворюється з 4 елементів VSD = {V_3, V_5, V_7, V_9}. У свою чергу вершина V_3 пов'язана тільки з однією вершиною V_6, що належить підграфу GD. У цьому випадку підмножина вершин {V_4, V_5, V_6} також буде мінімальною розділяючоюмножиною. Остання вершина V_5 зв’язана з двома вершинами V_8 і V_{11} підграфа GD. У цьому випадку підмножина зчленування VSDпідграфів GSi GD не є мінімальною, бо утворюється з 4 елементів VSD = {V_4, V_6, V_8, V_{11}}. На наступному кроці розглядаєтьсямножина вершин $Vi \in GD$, пов'язаних з вершинами мінімальної розділяючоїмножиниVmin = {V_4, V_5, V_6}, серед яких формується новамножина.

Процес формування мінімального розділяючоїмножини триває поки в підграфі GD не залишиться тільки одна кінцева вершина D. При цьому залежно від топології графа і середнього значення ступеня його вершин в графі може існувати кілька мінімальних розрізів між двома довільними вершинами, наприклад, для графа, представленого на рис. 1, між вершинами S і D існує 4 мінімальних розрізи графу:C_1 ={V_4, V_3, V_5}; C_2 = {V_4, V_6, V_5}; C_3 = {V_9, V_{10}, V_{11}}; C_4 = {V_{12}, V_{10}, V_{13}}.

Мінімальний перетин дорівнює 3, тому в мережі може бути сформовано не більше трьох непересічних маршрутів, що проходять через вершини {V_3, V_4, V_5, V_6, V_9, V_{10}, V_{11}, V_{12}, V_{13}} мінімальних перетинів.

Для формування множини непересічних шляхів пропонується використати модифікований алгоритм пошуку шляху в графі, заснований на методі пошуку в ширину і відомий як алгоритм зустрічної хвилі. Ініціаторами пошуку є кожен вузол системи. Після зустрічі хвилі формується шлях по довжині, що складається з вузлів які були пройдені зустрічними хвилями.

Алгоритмом також передбачається формування запасних шляхів. Суть методу полягає у формуванні запасного шляху на підставі вибору чергової суміжної вершини, максимально близької до однієї з вершин основного шляху.

При використані цього алгоритму вершинами є маршрутизатори мережі GMPLS. Отже хвилі-пакети при зустрічі в проміжних маршрутизаторах конкатенуються, загортаються в пакет та відправляються до кінцевих маршрутизаторів. Кінцеві роутери записують їх до своєї таблиці маршрутизації.

Використовуючи описаний вище алгоритм для графу, зображеного на рис. 1 було побудовано такі непересічні маршрути: основний шлях $R_0 = \{S, V_3, V_6, V_{10}, D\}$ та 2 запасних: $R_1 = \{S, V_4, V_7, V_9, V_{12}, D\}$ і $R_2 = \{S, V_5, V_8, V_{11}, V_{13}, D\}$.

Висновки.

У роботі було розглянуто способи побудови VPN мереж, приведено переваги використання технології GMPLS. Запропоновано спосіб визначення мінімальної підмножини зчленування зв'язаного неорієнтованого графа, що необхідно для знаходження кількості непересічних маршрутів.На основі цих даних було запропоновано спосіб безпечної багатошляхової маршрутизації, що дозволяє забезпечити максимально безпечну передачу інформаційних повідомлень при рівномірному завантаженні каналів передачі даних.

Перелік посилань:

1. N. Mahesh Kumar. Proposed Architecture For Implementing Privacy In Cloud Computing Using Grids And Virtual Private Network // N. Mahesh Kumar, K. Senthilkumar // International Journal Of Technology Enhancements And Emerging Engineering Research, Vol 1, ISSUE 3 2013. p. 12-15.

2. Hiroshi Matsuura. GMPLS-Based VPN Service to Realize End-to-End QoS and Resilient Paths//Hiroshi Matsuura, Kazumasa Takami, Young-Tak Kim, Makoto Takano // Management of Convergence Networks and Services// Springer Berlin Heidelberg, Volume 4238, 2006, p. 302-311.

3. Кулаков Ю.О. Комп'ютерні мережі: Підручник за редакцією Ю.С. Ковтанюка // Кулаков Ю.О., Луцький Г.М.– Київ.: Видавництво «Юніор», 2005. – 397с., іл.

4. Lemeshko A.V. Research on Tensor Model of Multipath Routing in Telecommunication Network with Support of Service Quality by Greate Number of Indices / Lemeshko A.V., Evseeva O.Yu., Garkusha S.V // Telecommunications and RadioEngineering, Vol.73, No 15. –P. 1339-1360.

5. Mahalakshmi.C Multipath Data Transfer Scheme for Virtual Private Networks/ Mahalakshmi. C, Ramaswamy. M// International Journal of Computer Applications (0975 – 8887) Volume 44– No.8, April 2012 P.27-31.

УДК (004.237 : 621.337.2)

ЛАЗОРСЬКИЙ Е.О.
НІКОЛЬСЬКИЙ С.С.

РЕАЛІЗАЦІЯ КОНТРОЛЕРА ПРІОРИТЕТНИХ ПЕРЕРИВАНЬ ДЛЯ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ З ВІДКРИТОЮ АРХІТЕКТУРОЮ

Вступ

Підвищення продуктивності обчислювальних систем (ОС) реального часу є важливою задачею в області застосування автоматичних систем управління. Особливістю таких систем є функціонування в режимі реального часу, що вимагає забезпечення гарантій виконання жорстких часових вимог. Складність виконуваних задач підвищують вимоги до сучасних ОС реального часу і обумовлюють розробку нових апаратних та програмних засобів для підвищення ефективності обробки та обміну даними.

Огляд літератури в даній області показав, що підвищення продуктивності паралельних обчислювальних систем реального часу досягається застосуванням методів і засобів зменшення витрат часу на ініціалізацію і синхронізацію процесів обміну даними. Так в роботі [1] запропоновані методи автоматичного динамічного розпаралелювання обчислень на рівні команд,

програмних модулів і програм у системах з різною архітектурою, метою яких є підвищення реакції систем управління на зовнішні події та розширення області їх застосування. В роботі [2] були запропоновані формули апаратно-орієнтованих алгоритмів швидких перспективних перетворень, адаптовані для програмно-апаратної реалізації процесорів реального часу, що задовольняють критеріям точності та швидкодії. Окрім того, однією з функціональних особливостей систем управління реального часу є необхідність обробки переривань від великої кількості зовнішніх пристроїв. Це потребує ефективної реалізації системи переривань. Н**а програмному рівні зазвичай питання обробки переривань вирішуються за допомогою спеціалізованих операційних системи реального часу, які базуються на** системі пріоритетів процесів і алгоритмів планування [3].

В даній роботі проблема підвищення продуктивності систем реального часу вирішується за рахунок модифікації архітектури системи переривань, що забезпечить гарантоване обслуговування переривань від великої кількості зовнішніх пристроїв на протязі фіксованого часу циклу управління.

Постановка задачі

Цілі дослідження: забезпечення гарантованого обслуговування переривань від кожного зовнішнього пристрою та уникнення тупикових ситуацій, характерних застосуванню географічного пріоритету, за рахунок чого підвищення продуктивності обчислювальних систем реального часу. Реалізація розподіленого контролера пріоритетних переривань (КПП) з фіксованими та динамічними рівнями пріоритетів запитів на ПЛІС, виконання моделювання в САПР Quartus II.

Особливості реалізації КПП з фіксованими та динамічними пріоритетами в ОС

З точки зору швидкодії обробка переривань на апаратному рівні є більш ефективною в ОС розглянутого класу [3]. Для обробки переривань на апаратному рівні використовуються спеціалізовані пристрої – централізовані та розподілені КПП. Перевагами розподілених контролерів переривань є невелика кількість ліній зв’язку у шині управління та простота нарощування зовнішніх пристроїв. Недоліками такої системи є велика частота звернень процесора до системної магістралі під час ініціалізації системи, а також використання фіксованих рівнів пріоритетів запитів, які не забезпечують гарантованого обслуговування заявок від зовнішніх пристроїв.

Досліджена обчислювальна система, структурна схема якої зображена на рис. 1. Система складається з процесора *P* (*Processor*) і зовнішніх пристроїв *PU* (*Peripheral Unit*), пов’язаних між собою загальною шиною *GB* (*Global Bus*). У системі реалізована обробка зовнішніх векторних переривань за допомогою розподіленого контролера переривань, у склад кожного *PU* входить блок обробки переривань *IB* (*Interrupt Block*). Готовий *IB* до обміну даними видає сигнал запиту переривання *IR* (*Interrupt Request*) на загальну лінію вимоги переривань *ID* (*Interrupt Demand*). Відповідний сигнал процесора підтвердження переривання *IA* (*Interrupt Acknowledgement*) поширюється послідовно через усі *IB*, що утворюють так називаний пріоритетний «ланцюжок» (*Daisy Chain*), пріоритети обробки запитів переривань від зовнішніх пристроїв залежать від їх географічного розташування по відношенню до процесора. В КПП з послідовним передаванням пріоритету в кожному такті максимальний пріоритет, тобто початок ланцюга, передається наступному у ланцюгу процесору. За цього максимальний час очікування обслуговування дорівнює $(n-1)$ тактів, де n – кількість зовнішніх пристроїв. За такого способу реалізації динамічних пріоритетів максимальний пріоритет (початок ланцюга) отримує зовнішній пристрій, наступний за тим, що був обслугований у поточному такті.

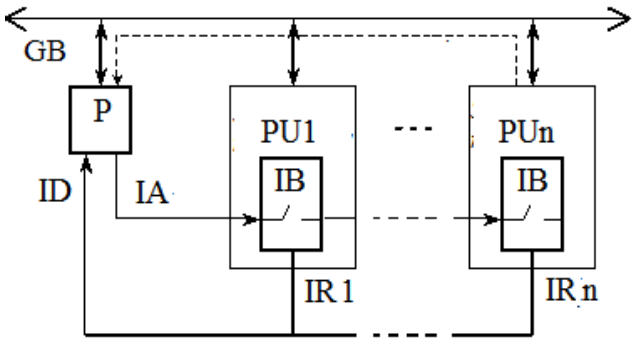


Рис. 1. Структурна схема обчислювальної системи з фіксованими пріоритетами

Обчислювальна система (рис. 2) містіть процесор *P*, *n* зовнішніх пристроїв *PU*, загальну шину *GB*, до якої підключені процесор *P* та зовнішні пристрої *PU*. До складу кожного зовнішнього пристрою входить блок формуванняпереривань *IB*, тригер *D*, елемент І (&) та елемент АБО (*I*) (апаратура зовнішніх пристроїв, яка не стосується реалізації переривань на рис. 2 умовно не показана). Вихід тригера *D* підключений до першого входу елемента І, вихід якого зв'язаний з першим входом елемента АБО, який своїм виходом підключений до входу блока переривань *IB*. Вихід *NEXT_TR* кожного тригеру *D* приєднаний до інформаційного входу *DIN* тригера наступного зовнішнього пристрою, при цьому тригер *D* останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, тригери *D* об'єднані у кільце. Вихід *NEXT_PU* кожного блоку переривань приєднаний до входу елемента АБО наступного зовнішнього пристрою, при цьому блок переривань останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, блоки переривань *IB* через елементи АБО об'єднані у кільце. Виходи *IR* блоків переривань *IB* об'єднані у єдину лінію і підключені до входу вимоги переривань (*ID*) процесора *P*, вихід підтвердження переривання (*IA*) якого підведений до керуючих входів тригерів *D* та других входів елементів *I*.

Обчислювальна система працює наступним чином. У кожний момент часу тільки в одному із тригерів *D* записана одиниця (під час ініціалізації – у тригері першого в пріоритетному ланцюзі *PU*). Всі інші тригери встановлені в нуль. Найвищій пріоритет має зовнішній пристрій *PU*, тригер пріоритету якого встановлений в одиницю. Готовий до обміну інформацією з процесором *P* будь-який зовнішній пристрій *PU* формує сигнал запиту переривання (*IR*). За наявності такого сигналу розривається пріоритетний ланцюжок між входом та виходом *NEXT_PU* блоку переривань *IB*. Якщо є хоч один сигнал на виходах *IR*, формується загальний сигнал вимоги переривань на вході *ID* процесора. Після закінчення чергової команди процесор у відповідь на сигнал *ID* формує сигнал підтвердження переривання на виході *IA*. Цей сигнал потрапляє у пріоритетний ланцюжок, замкнутий у кільце. При цьому початок ланцюжка визначає логічний елемент І, який відкривається високим рівнем сигналу в тригері збереження пріоритету, як вже було зазначено в такому стані знаходиться тільки один зовнішній пристрій системи. Сигнал *IA* розповсюджується по ланцюжку до першого на його шляху блока переривань *IB*, який виставив сигнал запиту переривання *IR*. В цьому блоці *IB* формується сигнал на виході *NEXT_PU*. За зрізом сигналу *NEXT_PU* формується сигнал дозволу видачі вектору переривання, який через загальну шину *GB* надходить в процесор *P*. Процесор переходить на виконання програми обслуговування переривання після чого знімає сигнал *IA*. Після зняття процесором сигналу *IA* і за фронтом сигналу *IA* одиниця з виходу *NEXT_TR* зовнішнього пристрою *PU* з найвищим пріоритетом переписується у тригер *D* наступного *PU*. У цьому випадку пріоритети передаються послідовно від одного зовнішнього пристрою до іншого. Що гарантує обробку переривань від всіх зовнішніх пристроїв на визначеному проміжку часу.

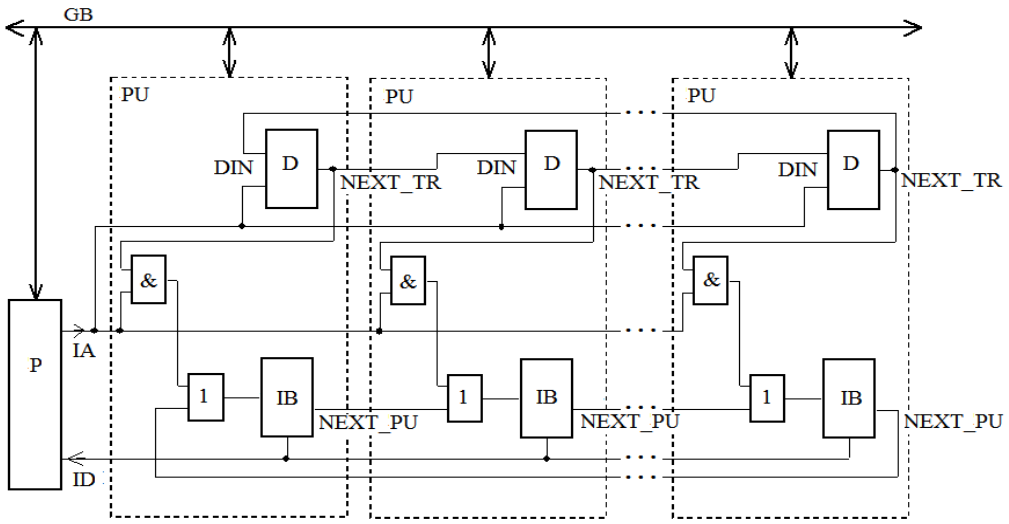


Рис. 2. Обчислювальна система з послідовним передаванням пріоритету

В обчислювальній системі з динамічним передаванням пріоритету (рис. 3) на відміну від обчислювальної системи з послідовним передаванням пріоритету, кожний блок переривань має третій вихід *NEXT_TR*, який приєднаний до інформаційного входу *DIN* тригера наступного зовнішнього пристрою, при цьому блок переривань останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, блоки переривань *IB* через елементі АБО об'єднані у кільце. Виходи *IR* блоків переривань *IB* об'єднані у єдину лінію і підключені до входу вимоги переривань (*ID*) процесора *P*, вихід підтвердження переривання (*IA*) якого підведений до керуючих входів тригерів *D* та других входів елементів *I.*

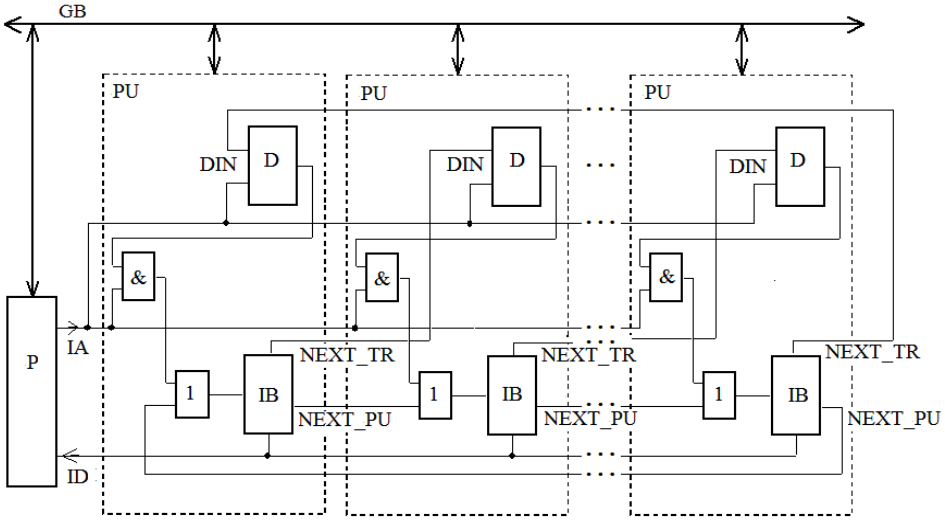


Рис. 3. Обчислювальна система з динамічним передаванням пріоритету

Результати моделювання

Виконано моделювання розподілених контролерів пріоритетних переривань з послідовним передаванням пріоритету та динамічним передаванням пріоритету в САПР ***ModelSim***. В якості результати досліджень наведені часові діаграми роботи контролерів переривань.

Моделювання модулів КПП показало, що за наявності сигналу запиту переривання *IR* зовнішній пристрій *PU* отримує сигнал підтвердження переривання *IA*. Якщо на тригері *D* встановлено нульове значення, зовнішній пристрій *PU* переходить у стан очікування за чергою.

Висновки

Застосування розподіленого КПП відповідає модульному принципу організації ОС і ефективно вирішує задачу масштабування обчислювальних систем з відкритою архітектурою, що до кількості зовнішніх пристроїв.

Застосування системи з динамічними пріоритетами гарантує обслуговування переривань від кожного зовнішнього пристрою на визначеному проміжку часу, що надає можливість уникнути тупикових ситуацій та простоїв і підвищити швидкодію системи.

Застосування контролерів з послідовним передаванням пріоритетів ефективно в обчислювальних системах з однорідним циклом функціонування. Тут кожний зовнішній пристрій на протязі (n-1) тактів роботи системи гарантовано отримає обслуговування. В системах що вирішують різного роду задачі управління в тому числі і траєкторні задачі, цикли управління характеризуються своєю неоднорідністю. Таким чином у визначений момент часу виконується опитування та обробка переривань від певної групи ЗП, що впливають на стратегію управління. Інші ж ЗП знаходяться у пасивному стані. В таких системах застосування КПП з послідовним передаванням пріоритетів буде обумовлювати затримку початку обслуговування переривань, що буде визначатись довжиною пріоритетного ланцюжка. Застосування КПП з динамічним передаванням пріоритету дозволяє підвищити ефективність обробки переривань в системах з неоднорідним циклом управління і великою кількістю зовнішніх пристроїв за рахунок „обминання” зовнішніх пристроїв, які не приймають участі на даному етапі управління.

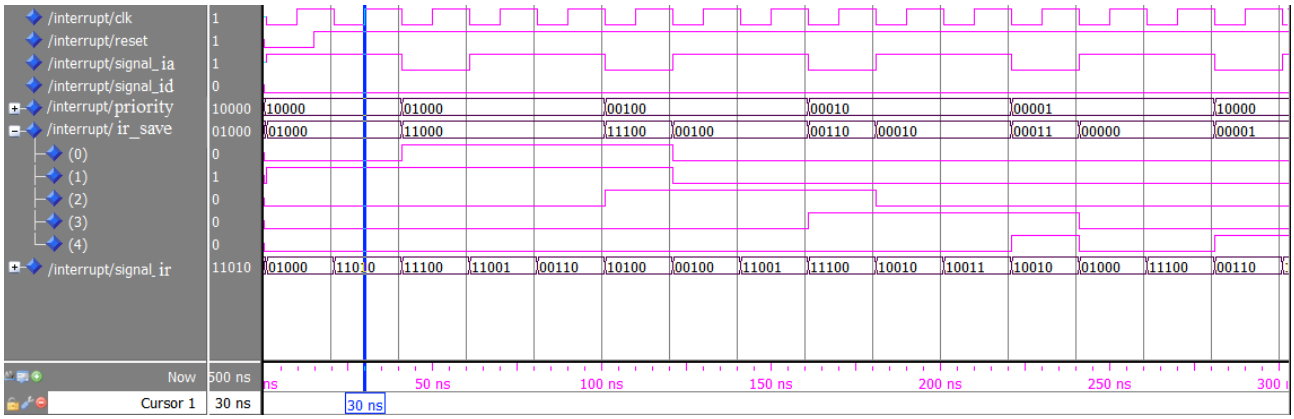


Рис. 4. Часова діаграма обчислювальної системи з послідовним передаванням пріоритету

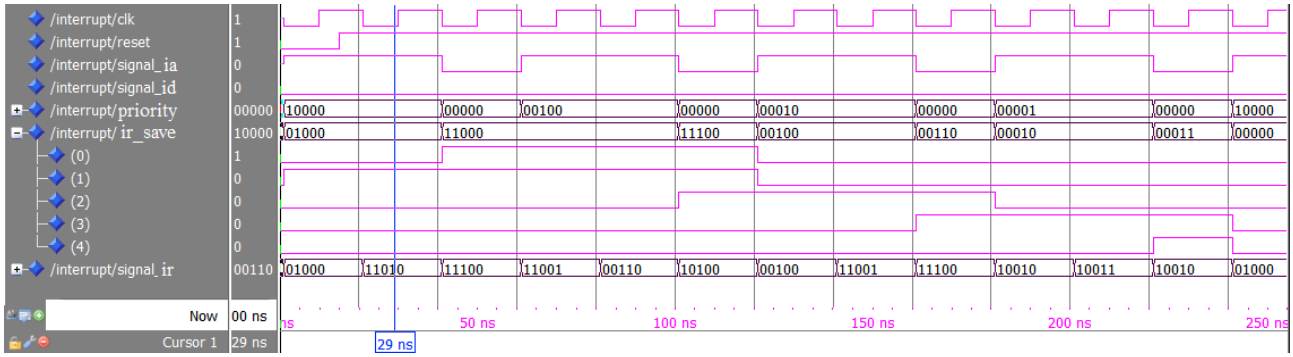


Рис. 5. Часова діаграма обчислювальної системи з динамічним передаванням пріоритету

Перелік посилань
1. Автореф. дис. д-ра техн. наук: 05.13.13 / В.І. Жабін; Нац. техн. ун-т України "Київ. політехн. ін-т". — К., 2006. — 36 с. — укр.
2. Автореф. дис. канд. техн. наук: 05.13.13 / М.В. Васильєв; Харк. нац. ун-т радіоелектрон. — Х., 2003. — 19 с.: рис. — укр.
3. Жабин В.И. Архитектура вычислительных систем реального времени. – К.: ВЕК +, 2003. – 176.

УДК 004.93(622.2)

МОЖАРОВСЬКИЙ А.С.,
ТЄЛИШЕВА Т.О.

СПОСІБ ОПТИМІЗАЦІЇ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ БУРІННЯ НАФТОВИХ СВЕРДЛОВИН НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ

Наведений аналіз способів керування процесом буріння під час поглиблення свердловини і обґрунтована можливість і актуальність розробки нечіткого способу оптимізації вартості метра проходки свердловини з використанням метода адаптивного керування процесом поглиблення. Розглянута базова модель процесу поглиблення свердловини і критерій оптимального буріння свердловини. Обґрунтоване введення нечіткості при визначенні часу спуско-підйомних операцій, що впливає на досягнення оптимального значення критерія. Вказані шлях розв`язання задачі нечіткої оптимізації і вибір алгоритму рішення.

Ключові слова: буріння, поглиблення свердловини, математична модель , нечіткість, нечіткий критерій оптимальності , нечітка змінна, принцип розширення, нечітка логіка.

The subject of the article is the analysis of methods of drilling process control whilewell deepening and proved the possibility and urgency of developing a fuzzy method for optimizing the costper meter of the well penetration using the method of deepening process adaptive management. We consider the basic model of the process of deepening wells and the optimal drilling conditions. Reasonable introduction of ambiguities in determining the time of dip and lift operations,which has influence for the achievement of the optimal value of the criterion. Shown the way of solving the problem of fuzzy optimization and choosing the algorithm of solutions.

Keywords: drilling, deepening wells, mathematical model, fuzzy, fuzzy optimality criterion, fuzzy variable, extension principle, fuzzy logic.

Вступ

Поглиблення свердловин на нафту і газ є найдорожчим етапом з проекту. побудови свердловин. Оптимізація процесу поглиблення свердловин суттєво впливає на техніко-економічні показники будівництва свердловин. Складність оптимізації процесу поглиблення полягає в неоднорідності порід, що буряться і неповноті інформації про взаємодію породоруйнівного інструменту з породою.Всилу природної складності більшість технологічних та геофізичних параметрів в процесі поглиблення мають нечіткий характер.

Для оптимізації процесу поглиблення розроблено математичні моделі[1,2], які в різному сенсі задовольняють потреби виробництва, але останнім часом багато уваги приділяють сучасним способам оперативного управління на основі нечіткої логіки[2], які дозволятьпідвищити ефективність управління і рентабельністьза рахунок інтелектуалізації моделей і алгоритмів оптимізації.

Таким чином, розробка нечіткої моделі поглиблення свердловини для оптимізації вартості метра прохідки і побудова на цій основі способуоптимізації процесу буріння є актуальною.

Постановка задачі

Процес механічного буріння є стохастичним, нестаціонарним і таким, що розвивається у часі. Це значно ускладнює визначення цілого ряду параметрів, що впливають на процес руйнування породи. Базовими параметрами технологічного процесу поглиблення при роторному бурінні є осьове навантаження на долото, частота обертання долота, витрата промивної рідини,зношення опор та озброєння долота та ін.

Оскільки вище наведені параметри є нечіткими при вимірюванні їх безпосередньо в процесі поглиблення, то в роботі розглядається питання розробки нечіткої моделі на основі розробленої диференційної моделі[3]і можливість використання нечіткої моделі в способі нечіткого оптимального керування процесом поглиблення свердловини.

Мета дослідження – скласти нечітку модель процесу поглиблення свердловини, на базі якої є можливий спосіб нечіткого оптимального керуванняпроцесом поглиблення свердловини з мінімальною вартістю метра прохідки свердловини.

Для цього необхідно скласти нечітку модель для оптимального керування процесом поглиблення, перевірити чи є вона адекватна існуючій диференційній моделі [3]і чи дає це дослідження право на використання її в способі нечіткого керування.

Розробити алгоритми та програмне забезпечення для способу нечіткого керування, перевірити на адекватність результати при порівнянні способу нечіткого керування і керування на базі диференційної моделі процесу поглиблення свердловини.

Розгляд існуючих рішень

Існує багато математичних моделей, які розроблені вченими на протязі багатьох років Аналіз моделей з точки зору їх використання в оперативному керуванні процесом поглиблення свердловин на момент 1988 року наведений в роботі [3] . За наступний період до аналізу існує більше робіт, однак по суті поставленої мети найбільшу увагу визивають наступні роботи.

Модель Елдреда[1]призначена для оптимізованого виконання різних операцій буріння, вона заснована на свердловинних вимірах. Модель процесу буріння представляє комбінований вплив умов на забої свердловини і роботи колони бурильних труб. На основі реакції зворотного зв'язку система використовує змінні моделі процесу буріння і моделі пластів для визначення оптимальної механічної швидкості проходки для пласта що в даний момент буриться. Результати вимірювань як реакції зворотного зв'язку одночасно використовуються для підтвердження достовірності діючої в даний момент моделі пластів. Якщо виявлені відхилення, модель пластів оновлюється так, щоб вона відображала нові результати вимірювань. Цей процес відбувається безперервно протягом всього процесу буріння пласта.

Недоліки: спосіб передбачає побудову в ході роботи в свердловині адаптивної моделі буріння, яка містить параметри моделі пластів і параметри гідравлічної моделі промивання, що дозволяє підлаштовувати її під умови на забої, проте модель не враховує можливості вибору оптимального режиму буріння – вона призначена для роботи в рамках одного режиму.

Відома також модель процесу буріння Самсоненко-Бойченко[4], що представляє собою експонентний тренд механічної швидкості проходки, яку отримують в результаті шести і більше експериментальних даних буріння в інтервалі пласта однаковою буримості з подальшою їх апроксимацією методом найменших квадратів. Далі коефіцієнти моделі буріння підставляють в критерій «мінімум вартості метра проходки», що містить вираз залежності часу буріння t від осьового навантаження на долото G, частоти його обертання n і витрати бурового розчину Q у вигляді полінома першого ступеня, і виробляють пошук мінімуму критерію методом штрафних функцій . Отримані оптимальні параметри встановлюють на буровій установці і подальше регулювання буріння виробляють на оптимальних уставках G, n і Q.

Недоліки: спосіб вимагає попереднього проведення факторного експерименту для отримання поліноміальної залежності часу буріння від навантаження на долото, частоти його обертання і витрати бурового розчину; використовує показник роботи долота на неоднорідних пластах, що не є найбільш оптимальним.

Базова математична модель

Для розробки способу нечіткого керування найбільш прийнятна математична модель, яка була розробленав Івано-Франківському інституті нафти і газу колективом дослідників під керівництвом професора Семенцова Г.Н.[5]і використана для метода адаптивного оптимального керування процесом поглиблення свердловини[3].Вона дозволяє визначити оптимальне осьове навантаження на долото в процесі його роботи з урахуванням мінливих умов. Експериментально і теоретично підтверджено, що математична модель справедлива при різних формах зносу зубів долота.

Математична модель:

$$\begin{cases} \dfrac{dh}{dt} = \dfrac{V_0(F,n)}{\varepsilon} \\ \dfrac{d\varepsilon}{dt} = K_\varepsilon(F,n) \\ \dfrac{d\xi_{\text{оп}}}{dt} = K_{\xi_{\text{оп}}}(F,n) \end{cases} \qquad (1)$$

пригранічних умовах:

$$\begin{gathered} h(0) = \xi_{on}(0) = 0, h(t_\delta) > 0; \\ \varepsilon(0) = 1{,}1 \le \varepsilon(t_\delta) \le (1+m)^2; \\ 0 \le \xi_{on}(t_\delta) \le 1; \\ F_{\min} \le F \le F_{\max}, \\ n_{\min} \le n \le n_{\max}, \\ Q = const \end{gathered} \qquad (2)$$

де h - проходка; F -осьове навантаження на долото, n - частота обертання долота; $\xi_{\text{оп}}$ - відносне зношення опор долота; t_δ - час буріння; Q - витрата промивальної рідини; ε - оцінка відносного зносу озброєння долоті; K_ε - швидкість зміни оцінки відносного зношення озброєння долота.

$$V_0(F,n) = K_1 F^{\alpha_1} n^{\beta_1} \qquad (3)$$

$$K_\varepsilon(F,n) = K_2 F^{\alpha_2} n^{\beta_2} \qquad (4)$$

$$K_{\xi_{\text{оп}}}(F,n) = K_3 F^{\alpha_3} n^{\beta_3} \qquad (5)$$

де K, α, β – залежать від типу долота та фізико-механічних властивостей гірських порід.

При цьому $K_\varepsilon(F,n)$описує ситуацію коли зношується озброєння долота,$K_{\xi_{\text{оп}}}(F,n)$ – коли зношуються опори .

Оптимізація режимів роботи бурової установки проводиться за умов підтримки осьового навантаження Fпротягом деякого проміжку часу Δt і встановленої частоти обертання $n = const$.Припускається, що за час Δt властивості породи не змінюються або змінюються в незначній мірі, тому початкова швидкість проходки $V_0(F,n) = V_0 = const$.

Для рішення поставленої задачі оптимізації був вибраний критерій мінімальної вартості метра проходки.

$$q = \frac{C_\delta(t_\delta - t_{\text{сп}}) + d}{h} \qquad (6)$$

де C_δ – вартість години роботи бурової установки,t_δ – загальний час буріння, $t_{\text{сп}}$ – час, що витрачається на спуско-підйомні операції, d–вартість долота.

Введення нечіткості

Значення $t_{\text{сп}}$ зазвичай передбачається відповідно до експериментальних вимірів на інших свердловинах або в попередніх рейсах на тій самій свердловині. Таке передбачення є досить складним якщо додатково до технічних характеристик свердловини та обладнання додати такі фактори як кваліфікація бригади, погодні умови та ін.

Постановка задачі: знайтиосьове навантаженняF із допустимої області A_U,таке що буде незмінним протягом часу Δtта критерій оптимальності набуватиме мінімального значення.

Тому розглядатимемо $t_{\text{сп}}$ як нечітку величину, що має функцію належності $\mu(t_{\text{сп}})$.

$$\mu(t_{\text{сп}}) = \exp\left\{-\frac{(t_{\text{сп}} - m)^2}{2\sigma^2}\right\} \tag{7}$$

де $m,\ \sigma$ – числа, що характеризують розмитість нечіткої величини $t_{\text{сп}}$.

Критерій оптимальності (6) зручно подати так:

$$q = \frac{C_\delta t_\delta + d}{h} + \frac{C_\delta}{h} t_{\text{сп}} \tag{8}$$

Оскільки $\mu(t_{\text{сп}})$ – функція належності нечіткої величини x і за умови, що якщо між множинами X і Y існує однозначна залежність $y = \varphi(x)$ така, що $x \in X, y \in Y$ – визначається $\varphi^{-1}(y) = \{x: x \in X, \varphi(x) = y\}$звідки отримаємо:

$$\mu(y) = \mu\left(\varphi^{-1}(y)\right) \tag{9}$$

Для випадку з критерієм що розглядається (9) отримає вигляд:

$$\mu(q) = \mu\left(\varphi^{-1}(q)\right) \tag{10}$$

Із рівняння (8) $\varphi^{-1}(q) = t_{\text{сп}} = \frac{q-a}{b}$, де $a = \frac{C_\delta t_\delta + d}{h}$, $b = \frac{C_\delta}{h}$

Підставимо у (7):

$$\mu(q) = \exp\left\{-\frac{(q - a - mb)^2}{2\sigma^2 b^2}\right\} \tag{11}$$

Виберемо певне значення $\mu(q) = \alpha$ та йому відповідне $q = q^*$яке можна визначити з (7) при умові $\alpha \in [0,1]$.

Врахувавши попередні умови та допущення можна визначити:

$$q = \frac{C_\delta(t_\delta + m) + d}{h} + \frac{\sigma C_\delta}{h}\sqrt{\ln\frac{1}{\alpha^2}} \tag{12}$$

за умови що q, h, t_δвизначені на універсумі Uдля якого виконуються обмеження $F_{min} < F < F_{max}, n_{min} < n < n_{max}$

Час буріннявизначається відповідно до чіткої моделі коли зношення озброєння долота випереджує зношення опор$t_\delta = \frac{\varepsilon^* - 1}{K_\varepsilon}$.

Тоді маємо критерій оптимальності для випадку колиозброєння долота зношується швидше ніж його опори:

$$q = \frac{C_\delta A_\varepsilon + K_\varepsilon(C_\delta m + d)}{V_0 \ln(A_\varepsilon + 1)} + \frac{\sigma K_\varepsilon C_\delta}{V_0 \ln(A_\varepsilon + 1)}\sqrt{\ln\frac{1}{\alpha^2}} \tag{13}$$

де $A_\varepsilon = \varepsilon^* - 1$, де$\varepsilon^*$ – значення оцінки ε в кінцевий момент часу.

Результати

Задача (13) є задачею нелінійного програмування з обмеженнями (2). Такі задачі, як правило, розв'язуються числовими методами, які для своєї реалізації вимагають, щоб критерій оптимальності і обмеження мали похідні до другого порядку включно; критерій оптимальності повинен бути унімодальною функцією, а обмеження – випуклими. Тільки у випадку виконання цих умов існують необхідні і достатні умови існування мінімуму скалярної функції на множині значень її аргументів, які задовольняють певним обмеженням.

Крім того критерій оптимальності (13) може мати складну топологію, наприклад, у вигляді яру, що також затрудняє розв'язок задачі оптимізації за допомогою класичних методів. Як альтернативу таким методам можна використати генетичні алгоритми.

Висновки

Показано, що через технологічні та суб'єктивні фактори тривалість спуско-підйомних операцій може розглядатись як нечітка величина з певною функцією належності. Тому задача розробки способу нечіткого оптимального керування процесом поглиблення свердловини може бути сформульована у термінах нечіткої математики. «Розмитість» $t_{\text{сп}}$ може суттєво впливати на величину вартості метра поглиблення яка розглядалась як критерій оптимальності. Вказані шляхи розв'язання для визначення значень нечітких параметрів.

Перелік посилань

1. Пат. 2244117 РФ. Способ управления работой в скважине и система бурения скважины / ЭЛДРЕД Уолтер Д. // Бюл. - 2004. - № 1. - С. 96.
2. Влацкая И.В., Литвинов М.А. Разработка нечеткой модели и алгоритма оптимизации выбора управляющих воздействий для технологического процесса бурения скважин // Компьютерная интеграция производства и ИПИ(CALS) технологии / Сборник статей всероссийской научно-практической конференции. - Оренбург: ИПК ОГУ, 2005. - 158 – 162 с.
3. Тєлишева Т.О. Разработка адаптивного метода управления технологическим процессом углубления нефтяных скважин: Автореферат до дис. канд. техн. наук. – Івано-Франківськ: Івано-Франківського нац. техн. ун-т нафти і газу, 1988. - 18 с.
4. Пат. 1231946 СССР. Способ регулирования процесса бурения / Самсоненко В. И, Бойченко В. А // Бюл. - 1995. - № 3. - С. 34.
5. Горбійчук М. І. Оптимізація процесу буріння глибоких свердловин : монографія / М. І. Горбійчук, Г. Н. Семенцов. – Івано- Франківськ : Факел, 2006. – 493 с.

УДК 004.738

ПЕТРЕНКО А.І.

АЛГОРИТМ ДИНАМІЧНОЇ РЕКОНФІГУРАЦІЇ МАРШРУТІВ В ОПТИЧНИХ МЕРЕЖАХ GMPLS

У даній роботі розглядається алгоритм для вирішення задачі реконфігурації маршрутів у оптичних мережах GMPLS зі спектральним ущільненням каналів (WDM). Запропонований алгоритм може відновити кілька збоїв одночасно, заощадити великий обсяг зарезервованої пропускної здатності, підвищити продуктивність використання доступної пропускної здатності та збалансувати загальну завантаженість трафіку мережі. Алгоритм базується на технології GMPLS (Generalized Multiprotocol Label Switching), що забезпечує більш швидкі та продуктивні методи відновлення, ніж аналогічні їм методики традиційних IP-мереж.

This study proposes an algorithm of dynamic reconfiguration in GMPLS optical networks with Wavelength Division Multiplexing (WDM). The proposed algorithm can recover multiple failures

concurrently, save lots of reserved bandwidth, improve bandwidth managing productivity and balance the overall workload of network traffic. The algorithm is based on technology GMPLS (Generalized Multiprotocol Label Switching), that provides faster and more productive methods of recovery, than traditional methods of IP-based networks.

Ключові слова: General Multiprotocol Label Switching (GMPLS), Wavelength division multiplexing (WDM), Dynamic reconfiguration.

Вступ

Мережа інтернет росте швидкими темпами. Щорічно зростає об`єм інтернет трафіку, кількість приватних та громадських IP-мереж. Область застосування обчислювальної техніки швидко розширюється, охоплюючи все більше і більше областей. У зв'язку з розширенням областей застосування засобів обчислювальної техніки, на певному етапі їхнього розвитку виникла об'єктивна потреба підвищити ефективність комп`ютерних мереж та швидкість передачі інформації.

Останні розробки в області MPLS [1] відкривають нові можливості для подолання деяких обмежень традиційних IP-мереж. MPLS комутатори використовують простий алгоритм заміни міток замість стандартного способу пересилання пакетів. Успіх MPLS дає поштовх для створення універсальної технології комутації з використанням міток. На даний момент IP-трафік в магістральних мережах передається з використанням ATM або SDH мереж. На кожному з рівнів вирішення питань маршрутизації, управління та розподілу мережевих ресурсів виконується незалежно один від одного. Тому основною вимогою при розробці технології стала уніфікація функцій управління. Нова технологія отримала назву Generalized MPLS (GMPLS) [2]. Вона розширює і уніфікує функції маршрутизації і сигналізації MPLS для будь-яких транспортних технологій канального і фізичного рівнів.

Оптоволоконні мережі забезпечують величезну пропускну здатність та можливість швидкої реконфігурації за розумною ціною. Оптичні мережі нового покоління дозволяють передавати інформацію на мультигігабітних та терабітних швидкостях. Таке підвищення швидкодії передбачає новий спосіб мультиплексування та перемикання. Таким чином, використання мереж WDM стає дуже важливим.

В WDM [3] мережах використовується оптичне волокно, що передає декілька хвиль через одне волокно. Деякі компоненти, такі як оптичні мультиплексори і демультиплексори дозволяють об'єднати багато хвиль в одному оптичному волокні, збільшуючи ємність і швидкість передачі даних. Проте, відмова волокна може перервати декілька транспортних потоків, що призведе до втрати важливих даних. Таким чином, методи діагностики та усунення несправностей стають дуже важливим питанням у подібних мережах.

У даній роботі було запропоновано модифікований алгоритм динамічної реконфігурації, що дозволяє гнучко відновлювати мережу в разі відмови та швидко перерозподіляти втрачену пропускну здатність.

Опис алгоритму

Запропонований алгоритм динамічної реконфігурації [4] може відновити кілька збоїв одночасно, заощадити великий обсяг зарезервованої пропускної здатності, підвищити продуктивність використання доступної пропускної здатності та збалансувати загальну завантаженість трафіку мережі. Алгоритм базується на мережі GMPLS (Generalized Multiprotocol Label Switching), що забезпечує більш швидкі та продуктивні методи відновлення, ніж аналогічні їм методики традиційних IP-мереж. Даний алгоритм уникає грубого "розсіювання" трафіку при використанні протоколів нижніх рівнів. Отже, у даному розділі розглянуто евристичний алгоритм, що дозволяє швидко відновлювати мережу після відмови, використовуючи сигнали управління GMPLS в оптичних мережах з спектральним ущільненням каналів (WDM). Запропонований алгоритм інтегрує традиційні методи відновлення мереж з утилізацією порушеної пропускної здатності з використанням схеми Largest Load - First Fit (LLFF) для досягнення швидкого відновлення.

В мережі GMPLS відмова однієї з фізичних ланок може призвести до порушення багатьох LSP, що проходили через дану ланку. Більше того, кожен LSP може займати декілька довжин хвиль на кожній ланці, що ускладнює процес відновлення. Запропонований алгоритм фокусується на даній проблемі, щоб призначити підходящі довжини хвиль для зірваних шляхів LSP та встановити найоптимальніший шлях для швидкого відновлення. Довільна мережа представляється графом G = (V, E), де E представляє сукупність ребер - ланок, а V - кінцева точка, вузол, що може бути пристроєм LER або LSR.

Ремаршрутизація: ER ⊂ E представляє собою шлях ремаршрутизації при відмові фізичної ланки від початкового до кінцевого вузлів ремаршрутизації. Подібний шлях може бути визначений, як

$$E_{R_i} = V_1, V_2, \dots, V_N,$$

де індекс і визначає номер шляху ремаршрутизації, а VN позначає вузол шляху ремаршрутизації під номером N.

Алгоритм реконфігурації об`єднує декілька традиційних способів відновлення – відновлення по шляху, відновлення на півшляху та відновлення по ланці для досягнення максимальної продуктивності. Виконання алгоритму починається відразу після виявлення відмови. Загалом, алгоритм може бути описаний наступними кроками:

1. В разі відмови перший вузол нижче від місця відмови (ближче до вузла-джерела) призначається активним вузлом ремаршрутизації, тобто першим вузлом відновленого LSP.

2. Виконується пошук всіх допустимих шляхів відновлення, що починаються у даному вузлі. На знайдені маршрути накладаються деякі обмеження – кожен з найдених шляхів має виводити до будь-якого з вузлів даного LSP вище від місця обриву не пересікаючи порушені сегменти мережі.

3. Обчислюється вага кожного зі знайдених маршрутів. В даному випадку вага шляху визначається як час, що витрачається кожним із доступних шляхів при обслуговуванні запиту. Функція часових витрат для кожного шляху ремаршрутизації визначається за формулою

$$Cost(E_{R_i}) = \sum_{j=1}^{N-1} T_{E_j} + N * T_V,$$

де T_{E_j} - затримка, спричинена поширенням сигналу у окремій ланці шляху E_{R_i}, а T_V - час, витрачений на обробку інформації в одному вузлі мережі.

4. Виконується огляд допустимих маршрутів. Знайдений шлях призначається в якості активного LSP. Якщо було знайдено декілька маршрутів, обирається шлях з найменшою вагою. Якщо жодного шляху не було знайдено – активним вузлом ремаршрутизації призначається перший вузол нижче від поточного активного вузла. Відбувається перехід до пункту 2. Якщо активний вузол є першим вузлом даного LSP, вважається, що допустимого шляху відновлення не було знайдено. Виконання алгоритму припиняється.

5. Перевіряється можливість утилізації пропускної здатності, що була зарезервована пошкодженими LSP – чи має знайдений маршрут достатню кількість вільних довжин хвиль для розміщення нового LSP. Якщо утилізація можлива – виконується перерозподіл трафіку пошкоджених LSP.

6. Виконується перевірка – чи всі пошкоджені LSP було ремаршрутизовано. В разі ствердної відповіді виконання алгоритму припиняється, в іншому випадку відбувається перехід до пункту 3.

7. Вищеописаний алгоритм виконується для кожного із пошкоджених LSP. У даному алгоритмі для перерозподілу трафіку використовується метод "Найбільше навантаження - перший підходящий" (Largest Load - First Fit). Звідси випливає, що трафік LSP з найбільшим навантаженням буде перерозподілено в першу чергу. Розподіл широкої пропускної здатності є складною задачею, тому навантажені LSP обробляються раніше, ніж LSP з невеликим обсягом трафіку. Якщо обсяг резервної пропускної здатності обраного шляху достатній для перерозподілу трафіку, даний шлях обирається в якості основного. В іншому випадку серед

доступних маршрутів буде обрано наступний шлях. Блок-схема вищеописаного алгоритму зображена на рис. 1.

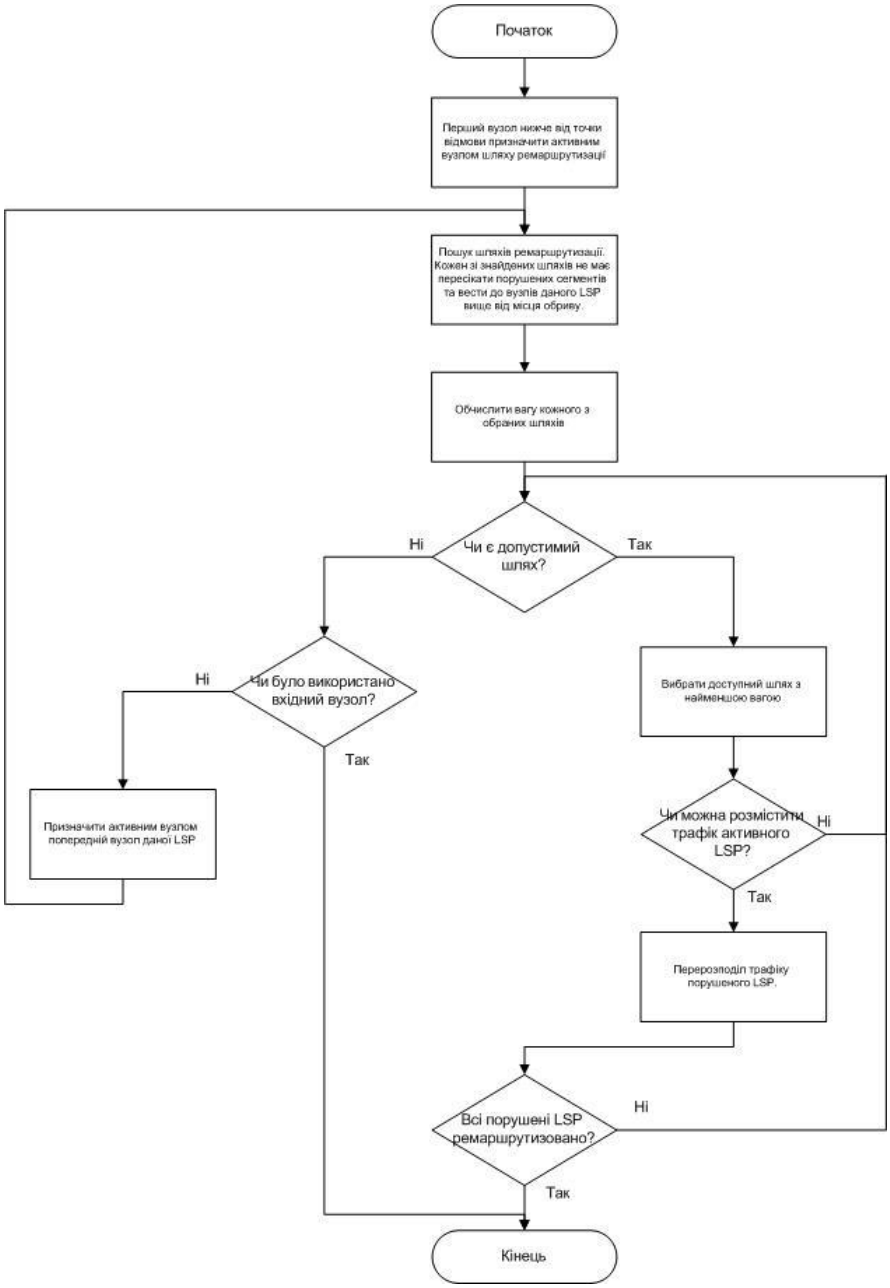


Рис. 1. Блок-схема алгоритму

Модифікації алгоритму

Розглянутий у даному розділі алгоритм є досить ефективним рішенням для розв`язання задач відновлення та ремаршрутизації у оптичних GMPLS-мережах. Представлений алгоритм дозволяє гнучко реконфігурувати мережу, досягаючи максимально швидкого відновлення та оптимального перерозподілу трафіку пошкоджених ланок. Проте, очевидний недолік даного алгоритму полягає у способі вибору шляху відновлення. Згідно з вищезазначеним алгоритмом, для вибору шляху проводиться обчислення ваги кожного зі знайдених шляхів ремаршрутизації, після чого обирається шлях з мінімальною вагою (пункти 3-4 алгоритму). Усі допустимі маршрути починаються з одного «активного» вузла ремаршрутизації, але можуть виводити до різних вузлів LSP вище від точки пошкодження. Таким чином, вибір найкоротшого шляху

ремаршрутизації не гарантує встановлення мінімального допустимого маршруту від вузла відновлення до вузла призначення.

Одним із способів вирішення даної проблеми є модифікація способу вибору шляху ремаршрутизації. Запропонована модифікація алгоритму полягає у обчисленні ваги кожного з маршрутів відновлення як суми ваг двох відрізків – власне шляху ремаршрутизації, та сегменту мережі від першого вузла активного LSP до вузла призначення. Таким чином, рішення про вибір маршруту приймається на основі інформації про повний шлях від вузла відновлення до призначення, а не його частини, як це представлено у оригінальному алгоритмі. Загалом, з урахуванням вищезазначених зауважень, пункти 3-4 оригінального алгоритму, що відповідають за вибір шляху ремаршрутизації, можна модифікувати наступним чином:

3. Обчислюється вага кожного із знайдених шляхів ремаршрутизації. В даному випадку вага шляху визначається як час, що витрачається при обслуговуванні запиту. Функція часових витрат для кожного шляху ремаршрутизації визначається за формулою

$$Cost\left(E_{R1_i}\right) = \sum_{j=1}^{N-1} T_{E_j} + N * T_V, \quad (1)$$

де T_{E_j} - затримка, спричинена поширенням сигналу у окремій ланці шляху E_{R1_i}, а T_V - час, витрачений на обробку інформації в одному вузлі мережі.

4. Для кожного зі знайдених у пункті 3 маршрутів виконується обчислення повного шляху відновленого LSP. Функція часових витрат для повного шляху обчислюється за формулою

$$Cost\left(E_{R_i}\right) = Cost\left(E_{R1_i}\right) + Cost\left(E_{R2_i}\right), \quad (2)$$

де $Cost\left(E_{R1_i}\right)$ - вага шляху ремаршрутизації, визначеного за формулою (1), а $Cost\left(E_{R2_i}\right)$ - вага сегменту шляху від останнього вузла шляху R1 до вузла призначення. Значення $Cost\left(E_{R2_i}\right)$ визначається аналогічно до ваги шляху R1 за формулою

$$Cost\left(E_{R2_i}\right) = \sum_{J=1}^{N-1} T_{E_j} + N * T_V, \quad (3)$$

5. Виконується огляд допустимих маршрутів. Знайдений шлях призначається в якості активного LSP. Якщо було знайдено декілька маршрутів, обирається шлях з найменшою вагою повного відновленого шляху, попередньо визначеного у пункті 4. Якщо жодного шляху не було знайдено – активним вузлом ремаршрутизації призначається перший вузол нижче від поточного активного вузла. Відбувається перехід до пункту 2. Якщо активний вузол є першим вузлом даного LSP, вважається, що допустимого шляху відновлення не було знайдено. Виконання алгоритму припиняється.

Подібна модифікація вимагає дещо більших обчислювальних витрат, ніж оригінальний алгоритм. Проте, вона оптимізує пошук шляхів, дозволяючи гарантовано отримати найкоротший шлях до вузла призначення. Модифікований алгоритм наведено на рис. 2.

Можна запропонувати іншу модифікацію даного алгоритму, що також стосується способів вибору кращого шляху ремаршрутизації. Нижченаведена модифікація призначена для оптимізації вибору шляху відновлення при високому степені завантаження мережі (більш, ніж 50 %).

При подібному ступені завантаження мережі основна проблема полягає в тому, що не всі фізичні зв`язки здатні забезпечити необхідну пропускну здатність. Таким чином, прийняття рішення про вибір маршруту на основі його ваги не завжди дає кращий результат. Запропонована модифікація полягає у обчисленні ваги шляху ремаршрутизації як кількості фізичних зв`язків у складі маршруту. Даний метод заснований на наступній логіці – чим більше фізичних зв`язків у складі маршруту, тим більша ймовірність того, що хоча б один із них не буде здатен надати необхідної пропускної здатності. Таким чином, основним параметром для прийняття рішення є довжина маршруту, а не його вага.

З урахуванням запропонованих зауважень, пункт 3 оригінального алгоритму можна модифікувати наступним чином:

обчислюється довжина кожного із знайдених шляхів ремаршрутизації. В даному випадку довжина шляху визначається як кількість фізичних ланок, через які проходить даний маршрут.

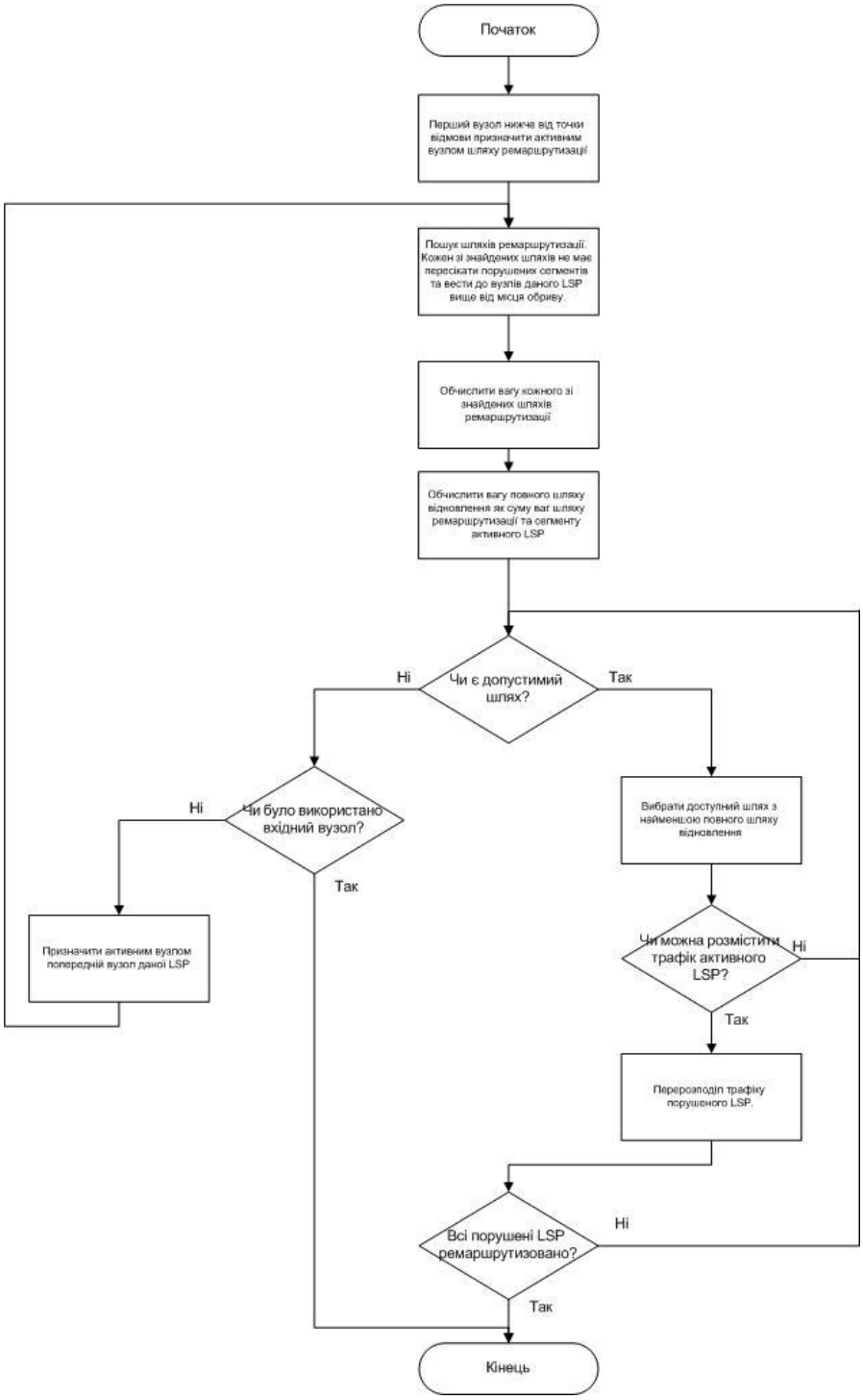


Рис. 2. Блок-схема модифікованого алгоритму

Довжина шляху ремаршрутизації може бути визначена за формулою

$$Cost\left(E_{R_i}\right) = \sum_{j=1}^{N} E_j, \quad (4)$$

де E_j - окрема ланка шляху. Подальше рішення приймається на основі обчисленої у даному пункті довжини.

Загалом, можна зауважити, що запропонований спосіб вибору шляхів є ефективним при високому ступені завантаженості системи.

Висновок

У даній роботі було запропоновано модифікований алгоритм динамічної реконфігурації оптичних мереж GMPLS. Розглянутий евристичний алгоритм дозволяє швидко відновлювати мережу після відмови, використовуючи сигнали управління GMPLS в оптичних мережах з спектральним ущільненням каналів (WDM). Запропонований алгоритм інтегрує традиційні методи відновлення мереж з утилізацією порушеної пропускної здатності з використанням схеми Largest Load - First Fit (LLFF) для досягнення швидкого відновлення. Результати досліджень показали, що алгоритм динамічної реконфігурації показує більшу швидкість та ймовірність відновлення, ніж традиційні методи відновлення.

Перелік посилань

1. Network Configuration Example GMPLS Feature Guide// *Juniper Networks, Inc.* - 2014.
2. Generalized Multiprotocol Label Switching. Definition and Overview// *The International Engineering Consortium*. – 2013
3. George N. Rouskas. Routing and Wavelength Assignment in Optical WDM Networks // *Encyclopedia of Telecommunications*. - 2003.
4. I-Feng Hwang I-Shyan Hwang Chia-Chun Chien. Dynamic Restoration with Just Enough Utilization Algorithm in IP with GMPLS over WDM Networks// *Department of Computer Science and Engineering, Yuan-Ze University*. – 2007.

УДК 004.056.5

Полєсков А.М.,
Волокита А.М.

СПОСІБ ТЕСТУВАННЯ СТІЙКОСТІ ПАРОЛІВ З ОЦІНКОЮ ЕНТРОПІЇ ЇХ СКЛАДОВИХ ЧАСТИН

В даній роботі показані існуючі системи перевірки стійкості паролів на прикладі алгоритмів Microsoft та Google, розглянуті основні недоліки кожного з алгоритмів, проведено їх порівняльний аналіз, запропоновано модифікований спосіб тестування стійкості паролів, який дає релевантну оцінку.

This article is about systems of password strength on example of Microsoft and Google algorithms, main disadvantages of each one. A comparative analysis was conducted. Solutions that override these shortcomings were proposed.

Ключові слова: пароль, перевірка, стійкість.
Key words: password, checking, strength.

Вступ

Паролі завжди були слабким місцем в безпеці комп’ютерних систем. Людський фактор зводив нанівець всю працю програмістів. Хакери знайшли безліч способів для зламу персональних сторінок. Найпопулярніший з них - це брутфорс, який генерує безліч варіантів комбінацій паролів допоки не підбере правильний. Метод може здаватися не ефективним, але завдяки людському фактору, він працює дуже добре. По статистиці дворічної давності,4,7% користувачів вибирають в якості пароля слово "password", 8,5% - "password" або "123456", 10 найпопулярніших паролів покривають 14% всієї призначеної для користувача бази (40% - топ-100 , 79% - топ-500, 91% - топ-1000) [1].

На шляху проблеми простих паролів стають системи перевірки їх стійкості. Вони допомагають користувачам зрозуміти, що їх пароль занадто простий, і його легко можна зламати. Але проблема у тому, що популярні системи перевірки стійкості паролів дуже прості, і не дають релевантного результату щодо стійкості, протидії брутфорсу та ентропії[2].

В даній роботі показані існуючі системи перевірки стійкості паролів на прикладі алгоритмів Microsoft та Google, розглянуті основні недоліки кожного з алгоритмів, проведений їх порівняльний аналіз, запропонований модифікований спосіб тестування стійкості паролів.

Сучасний алгоритм Microsoft.

Перевірка реалізована у вигляді скрипту passwdcheck.js[3].

Аналіз коду показав, що у скрипті Microsoft знаходиться два аналізатори стійкості паролів – актуальний, досить простий, та минулий, більш глибокий.

Розглянемо актуальний аналізатор. Рівень стійкості пароля дається в діапазоні [0; 4] в залежності від «бітової стійкості»***bits.***

- *bits* ≥ 128 — **4**;
- 128 < *bits* ≥ 64 — **3**;
- 64 < *bits* ≥ 56 — **2**;
- *bits* < 56 — **1**;
- пустий пароль — **0**.

Для оцінки «бітової стійкості» використовується формула:

bits = log(charset)*(length/log(2)),

де:

- *bits* — бітова стійкость;
- *log* — натуральный логарифм;
- *length* — довжина пароля;
- *charset* — сумарний розмір множин для кожного з типів нижче, якщо вони присутні в строці:
  - малі англійські літери [*abcdefghijklmnopqrstuvwxyz*];
  - великі англійські літери *[ABCDEFGHIJKLMNOPQRSTUVWXYZ]*;
  - спеціальні символи *[~`!@#$%^&*()-_+="]*;
  - цифри *[1234567890]*;
  - решта символів.

Алгоритм досить простий, тому він не враховує велику кількість аспектів стійкості паролів, як то словарні атаки та повторення символів.

Попередній алгоритм Microsoft

Попередній алгоритм в коді скрипту закоментований.

Для аналізу стійкості використовуються наступні показники:

- Довжина пароля.
- Кількість типів символів, що використовуються: великі латинські, малі латинські, цифри, спец-символи.
- Відсутність слова в словнику з урахуванням подібності символів.
- Близькість слова до словарного з певною ймовірністю.

Стійкість пароля ділиться на 5 класів [4].

1. Найкращий (Best):

- Довжина пароля — не менше 14.
- Кількість різних типів символів, що використовуються — не менше 3.
- В словнику відсутнє близьке з вірогідністю 0,6 слово з урахування подібності символів.

2. Стійкий (Strong):

- Довжина пароля — не менше 8.

- Кількість різних типів символів, що використовуються — не менше 3.
- В словнику відсутнє близьке з вірогідністю 0,6 слово з урахування подібності символів.

3. Средний (Medium):

- Довжина пароля — не менше 8.
- Кількість різних типів символів, що використовуються — не менше 2.
- В словнику відсутнє слово з урахування подібності символів.

4. Слабкий (Weak):

- пароль складається з мінімум одного довільного символу.

Таблиця подібності символів

- '3' - 'e'
- 'x' - 'k'
- '5' - 's'
- '$' - 's'
- '6' - 'g'
- '7' - 't'
- '8' - 'b'
- '|' - 'l'
- '9' - 'g'
- '+' - 't'
- '@' - 'a'
- '0' - 'o'
- '1' - 'l'
- '2' - 'z'
- '!' - 'i'

Алгоритм Google

На відміну від Microsoft, який реалізував перевірку за допомогою JavaScript (доступного для аналізу), Google перевіряє за допомогою Ajax-запиту до власного сервісу. Код запиту не знаходиться в публічному доступі, тому аналізувати його можна лише тестуванням різних паролів та порівнянням отриманих оцінок.

Посилання на запит:

http://www.google.com/accounts/RatePassword?Passwd=PsWD

Passwd – змінна паролю.

Повертає число 1-4, відповідне стійкості пароля.

Порівняння алгоритмів.

Порівнюючи алгоритми від флагманів IT-індустрії можна зробити висновок, що їх робота базується лише на алфавіті та регістрі символів. Обидва алгоритми Microsoft дуже поверхові і перевіряють лише кількість різних символів, зовсім не приділяючи уваги ентропії.

Щодо алгоритму Google, то крім самого пароля в цей сервіс (при зміні пароля на gmail) передається прізвище, ім'я і дата народження. Дані про IP можуть братися з браузера. Для простої перевірки на стійкість – це надлишкові персональні дані.

Запропонований спосіб перевірки стійкості

Ефективний алгоритм перевірки паролів на стійкість має об’єднати сильні сторони кожного з існуючих методів. При цьому найважливішим моментом є вірне обчислення ентропії пароля.

Мірою стійкості паролів традиційно є ентропія - міра невизначеності, яка вимірюється в бітах. Ентропія в 1 біт відповідає невизначеності вибору двох паролів, в 2 біта - з 4 паролів, в 3 біта - з 8 паролів і т.д.Ентропія в N біт відповідає невизначеності вибору з 2^N паролів [5].У разі випадкових паролів (наприклад, згенерованих за допомогою генератора випадкових

чисел)ентропія дорівнює логарифму по основі два кількості можливих паролів для заданих параметрів.

Ентропія розраховується як число варіантів поділу надвоє безлічі можливих паролів. Нижче наведено найпростіший алгоритм оцінки надійності пароля:

- n: довжина пароля;
- C: потужність пароля: розмір символьного простору (26 для пароля, що містить тільки літери нижнього регістру, 62 - для пароля, що містить великі та маленькі літери регістрів, а також цифри).

Ентропія = $n * \log_2 C$.

Якщо пароль згенеровано негенератором випадкових чисел, а людиною, то вирахувати його ентропію набагато складніше. Найпоширенішим підходом до підрахунку ентропії в цьомувипадку є підхід, який запропоновано американським інститутом NIST2[6]:

- ентропія першого символу паролю становить 4 біта;
- ентропія наступних семи символів паролю становить 2 біта на символ;
- ентропія символів з 9-го по 20-й становить 1,5 біта на символ;
- всі наступні символи мають ентропію 1 біт на символ;
- якщо пароль містить символи верхнього регістру і неалфавітні символи, то його ентропія збільшується на 6 біт.

Але стандартна перевірка на ентропію не дає переваг у 2016 році через те, що існуючі обчислювальні потужності дозволяють дуже швидко робити велику кількість переборів.

Коректно працюючий спосіб розрахунку стійкості паролю повиненобчислювати ступінь ентропії паролю, як суму ентропій його складових частин. Будь-які частини пароля, що знаходяться між ідентифікованими комбінаціями, вважаються послідовностями, що можуть бути підібрані перебором, тамають власну ентропію. Тоді простір загальних можливостей представлення паролю довжиною L або менше дорівнює наступному виразу (рис. 1).

$$\lg\left(\sum_{i=2}^{L}\sum_{j=1}^{\min t,i-1}\binom{i-1}{j-1}sd^j\right)$$

Рис.1

Запропонована програма базується на основі того, що ентропія пароля є сумою ентропій його частин. Не беручи до уваги «ентропію конфігурації», тобто ентропію числа і розташування частин пароля, спосіб не буде обчислювати загальну ентропію, що не надає структурі пароля ніякої цінності.Спосіб передбачає, що зломщикові структура пароля вже відома (наприклад, «прізвище-підібрана комбінація-цифри»), і розраховує тільки кількість спроб вгадати елементи пароля. Таким чином, при обчисленні ентропії враховується структура паролю [7].

Висновки

В даній роботі показані варіанти збільшення складності паролів, розглянуті основні недоліки існуючих систем перевірки стійкості, проведений порівняльний аналіз цих систем.

Описано спосіб, який за рахунок урахування структури паролю, дозволяє точніше вимірювати стійкість паролю, обчислюючи ентропію не для всього паролю, а окремих його частин. В майбутньому планується проведення дослідження варіантів обчислень загальної ентропії складених парольних структур на основі запропонованого способу та представлення його у вигляді веб-утиліти.

Перелік посилань

1. ITManagerDaily [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу:http://www.itmanagerdaily.com/bad-news-for-password-security(дата звернення 10.04.2016) – Назва з екрана.

2. Бєлєнко А. Пароли: стойкость, политика назначения и аудит / А. Бєленко - Защита информации. Инсайд. 2009. №1. С. 61-64.

3. Bonneau J. Guessing human-chosen secrets / Bonneau J. - Technical Report UCAM-CL-TR-819. 2012. 161 p.
4. Гуфан К.Ю. Оценка стойкости парольных фраз к методам подбора / К.Ю. Гуфан, В.А. Новосядлий, Д.А. Едель - Открытое образование. 2011. №2. С. 127-130.
5. Марков А.С. Методы оценки несоответствия средств защиты информации / А.С. Марков, В.Л. Цирлов, А.В. Барабанов, М.: Радио и связь, 2012, 192 с.
6. Brunett M. Perfect Password: Selection, Protection, Authentication. Syngress Publishin/ Brunett M. - 2006. 194 p.
7. Wheeler D. zxcvbn: realistic password strength estimation / Wheeler D. - Dropbox Tech Blog, 2012.

УДК 004.4'2

ПРОХУРЕНКОВ. В.
КУЛАКОВ Ю.О.

WEBRTC ЯК МЕТОД ВИРІШЕННЯ ПРОБЛЕМ ІНТЕРНЕТ-КОНВЕРГЕНЦІЇ КОМУНІКАЦІНИХ ПОСЛУГ

Зв'язок завжди був важливою частиною життя людини і засоби, за допомогою яких люди спілкувалися, просунулися дуже різко за останні кілька років, що пов'язано з розвитком технологій. До того ж, зі швидким прогресом в області технологій і з еволюцією електронних пристроїв, таких як комп'ютер, мобільні телефони, планшети та ін, традиційні методи комунікації поставлені під сумнів і є велика схильність до їх поліпшення.Технологія WebRTC,безумовно, є частиною цього поліпшення.

Таким чином, цей електронний документ описує концепціюWebRTC (Web Real-Time Communication), який є наступним покоління веб-зв'язку.Основною метою даної роботи є огляд майбутньої служби зв'язку, тобто WebRTC і вивчення його особливостей у наданні більш гнучкого способу спілкування, який дозволяє веб-браузерам забезпечувати комунцкацію в режимі реального часу.

Communication has always been an important part of human life and methods which people used for communication, moved very sharply in recent years due to technological developments. In addition, with the rapid advances in technology and the evolution of electronic devices such as computers, mobile phones, tablets and other traditional communication methods questioned and there is a great tendency to their improvement. WebRTC is part of this improvement.

Therefore, this document describes the concept of electronic WebRTC (Web Real-Time Communication), which is the next generation of Web communications. The main goal of this work is to review the future of communication services, i.e. WebRTC and study its features to provide a more flexible method of communication that allows web browsers to provide communication in real time.

Ключові слова: WebRTC, зв'язок в реальному часі, комунікація, браузер, протокол, веб-сервер.

Вступ

В даний час поряд з традиційними телекомунікаціямиактивно застосовуються інтернет-комунікації, і спостерігається їх конвергенція.Принцип конвергенції всіх служб зв'язку реалізується в мережах зв'язку NGN. Базовою NGN основою IP є опорні мережі, які можуть забезпечити інтеграціюпослуг передачі всіх типів трафіку: даних, голосу і мультимедіа. У мережі NGN,яка повинна забезпечити передачу всіх типів трафіку, основними рівнямиє транспортний і сервісний. Якщо в еволюції транспортних мереж спостерігається формування конвергентного

мережевого рівня P-OTN, то в еволюціїпослуг зв'язку (сервісних послуг), спостерігається формування конвергентних веб-комунікаційних послуг в режимі реального часу.

Інтернет-комунікації - це способи спілкування людей через мережу Інтернет.Найбільш перспективними напрямками Інтернет-комунікацій є Web-комунікації в режимі реального часу. Веб-комунікації в режиміреального часу - це способи спілкування через IP-мережі за допомогою веб-браузерабез установки плагінів і розширень. Веб-комунікації в режимі реальногочасу - це сервіси, побудовані по архітектурі "клієнт-клієнт".

До інтернет-комунікаційнихсервісів відносяться: системи обміну повідомленнями, онлайн-відео і VoIP. Службиобміну повідомленнями діляться на служби обміну повідомленнями в режимі офф-лайн (електронна пошта,SMS розсилка) і служби миттєвих повідомлень (IRC, веб чати, IM, PTT ітак далі) в режимі он-лайн.

Чати, веб чати, голосові і відео чати в веб інтерфейсі, IM, VoIP, - цесервіси, які забезпечують інтернет-комунікації в режимі онлайн черезскладові мережі з пакетною комутацією. Слід зазначити, що системи обмінуповідомленнями мають свої комунікаційні мережі, і в основному побудовані на архітектурі "клієнт-сервер". Інтернет-комунікаційні сервіси вимагають абоустановки клієнтських додатків на призначені для користувача пристрої, або установкиплагінів і розширень в веб-браузери.Сервіси, які забезпечують інтернет-комунікації в режимі онлайн,є додатками, в яких канали передачі голосу, відео, даних, текстуі файлів, як правило, не інтегровані, за винятком деяких IM, наприклад Skype.

Деякі з систем зв'язку працюють на приватних (закритих) протоколах.Як правило, зазначені програми не можуть одночасно працювати вдекількох комунікаційних мережах, тобто системи зв'язку не можутьвзаємодіяти один з одним, в результаті чого необхідно встановлюватиокремі додатки для кожного сервісу. Але слід зазначити, що деякісистеми зв'язку відкривають специфікації своїх протоколів і намагаютьсянадати можливість спілкування між різними системами.

На підставі вищевикладеного матеріалу основними проблемамиінтернет-конвергенції комунікаційних послуг є:

- інтеграція різних видів зв'язку в одному сервісі;
- забезпечення синхронізації різних видів зв'язку між собою івзаємодії їх з усіма існуючими мережами загального користування (PSTN, SIP, LTE, IMS);
- застосування відкритих протоколів зв'язку в комунікаційних послугахреального часу;
- застосування веб-браузерів (інтегрованих клієнтів) в якості єдинихзасобів спілкування (інтерфейсів) різних термінальних пристроїв.

Методи вирішення проблем інтернет-конвергенції комунікаційних послуг

Проблему конвергенції комунікаційних послуг реального часу, тобтоінтеграцію каналів передачі голосу, відео, даних і доступ до них за допомогоюєдиного мережевого додатку (веб-браузера) можна вирішити на основі технологійWebRTC, WebSocket протоколів і додатків SIPML5, webrtc2sip.По суті, браузери, які підтримують WebRTC, WebSocket і SIPML5 можуть бутиєдиним засобом (інтерфейсом) для всіх призначених для користувача пристроїв (ПК, смартфонів, IP-телефонів, мобільних телефонів і т.д.), які забезпечують комунікації в реальному часі[1].В даний час більшість веб чатів, як способів комунікацій черезІнтернет, засновані на мережевий клієнт-серверній архітектурі. Роль клієнтськоїчастини веб чатів виконує браузер.Але на зміну клієнт-серверній архітектурі, приходять однорангові абопирингові архітектури мережі (P2P), які можуть вирішитипроблему інтеграції каналів передачі голосу, відео і даних в одному пристрої- Веб-браузері.

В основному для створення P2P відеочатів застосовуються технології знизькою якістю передачі мультимедійних даних. Крім того, для виведенняголосу і відео потоку з мікрофона і відеокамери в p2p відеочатахнеобхідна установка плагіна для веб-браузера.

Одним із сучасних рішень в сфері веб-комунікацій є WebRTC технологія, а також протокол WebSocket, передбачений в специфікації HTML5, за допомогою яких вирішуються проблеми

створення P2P відеочату без застосування плагінів. Крім того,WebRTC забезпечує кращу якість передачі звуку і відео ніж Flash [2].

WebRTC - це відкрита технологія, призначена для створення пірінговихмереж зв'язку, яка дозволяє пересилати текстові та мультимедійні данібезпосередньо між браузерами без допомоги сервера [4]. Сигнальний сервер використовується тільки для установкир2р з'єднання між двома браузерами або WebRTC клієнтами. Програмний код клієнтської ісерверної частини чату реалізується на JavaScript. Клієнтська програмавзаємодіє з браузерами через API WebRTC [3].

Технологія WebRTC реалізується трьома інтерфейсами JavaScript API:

- RTCPeerConnection - для створення з'єднання "клієнт-клієнт" або Peer-toPeerміж браузерами (реалізує аудіо / відео дзвінки);
- Медіа потік (GetUserMedia) - для захоплення мікрофона, відео камери іпередачі мультимедіа;
- RTCDataChannel - для передачі даних.

WebSocket - забезпечує повністю асинхронний ісиметричний обмін повідомленнями між браузером і сервером через один TCP-сокет [5]. У деяких випадках в P2P чатах для асинхронного обміну повідомленнямиміж браузером і сервером може бути застосований Ajax (XMLHTTPRequest).

Додаток SIPML5 для VoIP або SIP клієнта для браузера, написані наHTML / CSS / JavaScript, і шлюз webrtc2sip відкрили шлях до створення SIP-софтфона в веб-інтерфейсі або SIP клієнта в веб-браузері. Тепер з будь-якого веб-браузера (який підтримує WebRTC), за допомогою SIPML5 можна здійснювати дзвінки або відеодзвінки в мережі SIP, LTE, IMS через шлюз webrtc2sipабо сервер Asterisk [6].

Висновок

WebRTC обіцяє перенести можливості комунікації на якісно новий рівень. Технологія з відкритим вихідним кодом проекту дозволяє сумісним веб-браузерам спілкування в режимі реального часу за допомогою простого JavaScript API. WebRTC надає можливості розробки додатків для спілкування в реальному часі для будь-якого веб-розробника. Важливою річчю є те, що WebRTC технологія «запакована» в браузер, тобто вмонтована в браузер.

WebRTC встановлює бачення майбутньої комунікації, яка в якійсь мірі вже існує в сьогоденні.

Таким чином, ця стаття зосереджена на огляді технологій веб-комункацій, а також на дослідженні WebRTCяк методу вирішення проблем інтернет-конвергенції комунікаційних послуг.

Перелік посилань

1.Вікіпедія [Електронний ресурс] –http://en.wikipedia.org/wiki/WebRTC.
2.Основи WebRTC (інструкція) [Електронний ресурс] – http://www.html5rocks.com/en/tutorials/webrtc/basics/.
3.WebRTC - Огляд [Електронний ресурс] –http://www.pkeconsulting.com/pkewebrtc.pdf.
4.WebRTC, питання та відповіді [Електронний ресурс] – http://www.webtorials.com/content/2013/05/eleven-answers-webrtc-explained.html.
5.WebRTC - новий метод веб комунікації [Електронний ресурс] – http://webrtcbook.com/presentations/WebRTCIEEE04-02-13.pdf.
6.Офіційний веб-сайт WebRTC [Електронний ресурс] –http://www.webrtc.org/.

УДК 004.93

РАДЕР Р.І.,
КУЛАКОВ Ю.А.,

Використання альфа-процедури для зменшення параметрів що відслідковуються для вирішення задачі масштабування

В статті розглянуто умови підтримання оптимальної архітектури в кластері серверів.Надано огляд переваг алгоритму альфа-процедура та приведений спосіб використання алгоритму для вирішення задачі масштабування. Проведено експерименти з побудови моделі на веб-додатку та базі даних.

Conditionstosupporttheoptimalclusterarchitectureareprovidedinthearticle.Theoverviewoftheadvantagesofthealpha-procedurealgorithmforusinginthetaskofscalingareintroduced. Experimentsoftrainingmodelsforwebapplicationanddatabaseareconducted.

Вступ

На сьогоднішній день наявний тренд з переходу IT інфраструктури малого та середнього бізнесу до хмарних постачальників послуг. Якщо поглянути на опитування Microsoft, у 2011 році 44% організацій були у стані обговорення переходу до хмари замість обслуговування власного парку серверів, опробовували 12% та реально використовували -- 15% [1]. У 2015 році, за даними CloudSecurityAlliance, 15% обговорюють перехід, 41% переходить і 33% реально використовують [2].

Основною перевагою хмарних технологій є можливість перерозподілу ресурсів між учасниками пулу в залежності від завантаження кожного з них. Таким чином, масштабування, в тому числі й автоматичне, є однією з ключових задач у таких системах.

Буде розглянуто спосіб побудови моделі додатку, що масштабується для автоматичного масштабування відповідно до моделі. Буде показаний спосіб зменшення кількості характеристик системи що відслідковується для масштабування, таким чином зменшуючи навантаження на систему керування.

Умови підтримання оптимальної архітектури

Обчислювальні системи що розраховані на обслуговування великої кількості користувачів та обробки великої кількості запитів на одиницю часу потребують гнучкого регулювання використання ресурсів, тому що навантаження не є постійним та залежить від різних факторів: від дня тижня або зміни дня і ночі. Для еластичної зміни споживаних ресурсів архітектура системи має прагнути бути лінійно масштабованою, тобто продуктивність роботи системи має бути якнайбільш лінійно-залежною від кількості ресурсів, виділених для додатку, без необхідності зміни вихідного коду додатку.

Крім періодичних змін можуть відбуватися абсолютно незалежні події, наприклад посилання на ваш ресурс може стати вірусним та залучити масований трафік. Таким чином, навіть якщо ваша архітектура дозволяє масштабувати кожен з компонентів системи без зміни коду лише додаючи ресурси, все ще необхідно щоб адміністратор або автоматизована система керування додавала та вилучала ресурси.

Таким чином, необхідні умови для підтримання оптимальної структури: лінійна масштабованість кожного з компонентів архітектури; система керування що виділяє оптимальну кількість ресурсів компонентам в залежності від поточного навантаження.

Досягти необмеженої лінійності масштабування неможливо, тому верхня межа можливостей такої архітектури буде обмежена продуктивністю роботи компоненту з мінімальною потужністю масштабування.

Використання Альфа-Процедури для вирішення задачі масштабування кластеру

α -процедура є алгоритмом розпізнавання образів та класифікації [3,4,5].

Найбільш важливими перевагами є те, що він

1. Непараметричний;
2. Може значно зменшити простір ознак;
3. Так як він працює тільки в 2D і 3D просторі, легко візуалізувати процес навчання і розпізнавання.

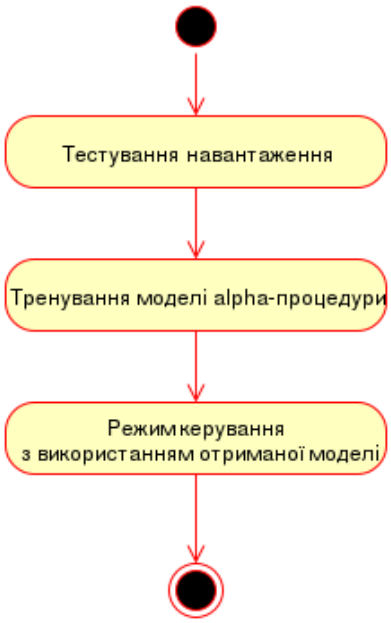


Рис. 1 Отримання оптимальної архітектури за допомогою тренованої моделі

Алгоритм є цікавим з декількох точок зору. По-перше, це є метод класифікації та розпізнавання образів, який будує гіперплощину у просторі ознак (де ознаками в даному випадку є різні динамічні характеристики віртуальних машин, такі як завантаження процесору, пам'яті, кількість запитів та інше) що розділяє класи. В нашому випадку класами є два випадки: час відповіді RTT перевищує порогове значення, або ні.

$$class = \begin{cases} A, & RTT \geq t_{threshold}; \\ B, & otherwise. \end{cases} \quad (1)$$

Використання алгоритму дозволяє редукувати простір ознак, тим самим не тільки визначаючи крайові пороги метрик, а й викидує ті метрики, що не впливають на такі динамічні характеристики системи, як час відповіді (RTT). Це дозволяє зменшити навантаження на систему вже в робочому режимі, зменшуючи кількість параметрів які необхідно відстежувати.

Алгоритм вимагає введення вектору ознак $P_k = (x_1, x_2, ..., x_n)$, які класифіковані «вчителем» на два класи, приймемо що вони називаються A і B. Оскільки метод непараметричний, ніяких інших даних не має бути надано, тобто вхідні дані можуть бути записані як множина векторів ознак із класом (2).

$$X = \{p_i\}, i \in \overline{1..k}; \; X = \{(x_1^1, x_2^1, ..., x_n^1, C^1), (x_1^2, x_2^2, ..., x_n^2, C^2), ..., (x_1^k, x_2^k, ..., x_n^k, C^k)\} \quad (2)$$

Метод заснований на покроковому вибору найбільш «потужної» ознаки. Якщо відобразити одну ознаку на числовій вісі, та позначити об'єкти тренувальних даних на ній, ми можемо обрати ту ознаку, що розділяє дані найкращим чином. Потужність є мірою роздільності даних $F(p_k) \in \overline{0..1}$.

Результатом роботи альфа-процедури є фактично гіперплощина, класифікатор, що дозволяє визначити приналежність нової точки до класу A чи B. Оригінальний метод альфа-процедури повертає гіперплощину як набір пар ознак та кутів проекції на нову вісь.

Модифікований алгоритм, що використовується в поточному дослідженні, будує класифікатор, на кожному етапі якого є не точка розділу, але потенціальна функція [6], функція що є позитивною в точках змінної де об'єкт має бути класифікований як клас А, та від'ємною в випадку де об'єкт має бути класифікований класом В.

Результати дослідження

В процесі дослідження було проведено тестування навантаження а також тренування моделей для веб-додатку що працює по протоколу HTTP, а також бази даних на двох видах постійних носіїв – твердотілий накопичувач та жорсткий диск.

Експеримент із веб-додатком

Було проведено тестування навантаження для веб-додатку. Отримана потенціальна функція зображена на рис.2. Обрана лише одна ознака x0, що є користувацьким процесорним часом за секунду часу. Таким чином, заданої умови (відповідь за <100ms) на обладнанні DigitalOcean (машина із 512 МБ пам'яті та 1 процесором) можна досягнути якщо додаток використовує менше 10 одиниць процесорного часу за секунду.

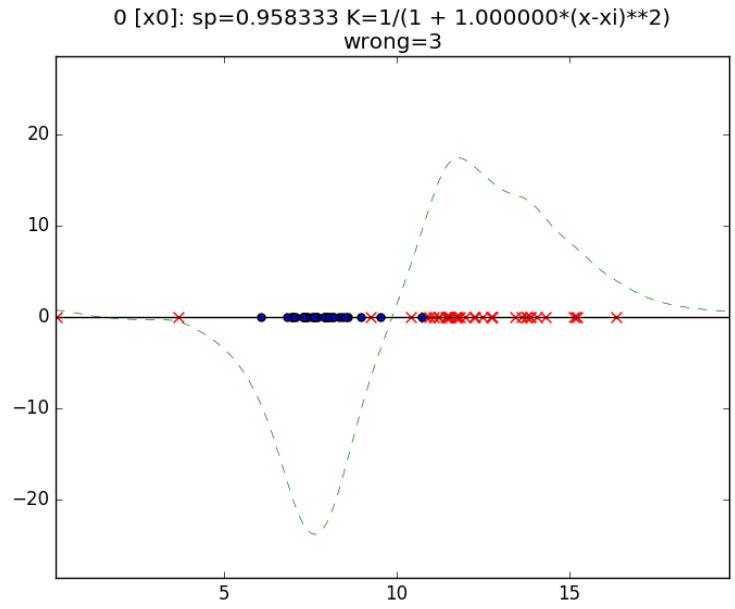


Рис. 2 Отримана потенціальна функція для веб-додатку

Експеримент із базою даних

Було проведено тестування навантаження для бази даних. Отримана потенціальна функція зображена на рис.3 та 4. Обрані алгоритмом ознакиx0 та x2, що є користувацьким процесорним часом за секунду часу та кількістю операцій вводу-виводу. Таким чином, заданої умови (відповідь за <100ms) на комп'ютері (із 512 МБ пам'яті та 1 процесором, та жорстким диском) можна досягнути якщо додаток використовує менше 30 одиниць процесорного часу за секунду та менше 30 операцій вводу-виводу за секунду.

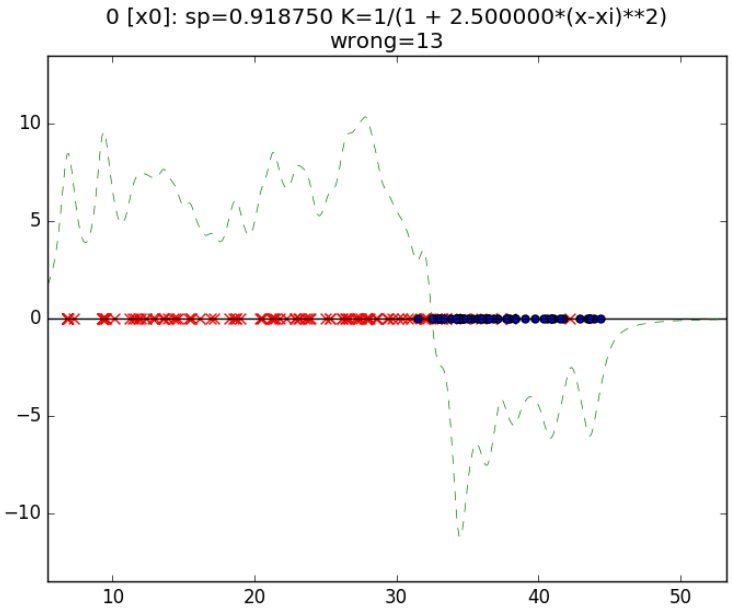


Рис. 3 Отримана потенціальна функція для бази даних, крок 1

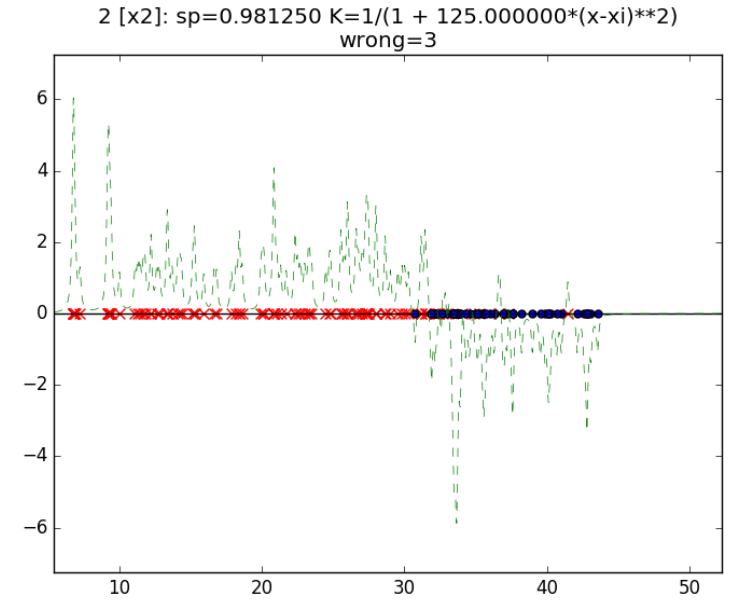


Рис. 4 Отримана потенціальна функція для бази даних, крок 2

Висновки

В даній роботі було описано спосіб використання модифікованого алгоритму альфа-процедура для редукування кількості ознак що необхідно відслідковувати системою контролю, а також отримання моделі (узагальненого портрету) додатку що працює на відповідному обладнанні, що може бути використано для вирішення задачі масштабування.

Було представлено спосіб зменшення кількості ознак що відслідковуються за допомогою тренування моделі альфа-процедури перед безпосередньо керуванням.

Показані результати експериментального тренування моделей, що суттєво зменшило кількість необхідних до відслідковування характеристик системи.

Перелік посилань
1.Microsoft, IT Pro QUESTION HOUR. IT ProCloudSurvey. – 2011 р, Режим доступу: https://technet.microsoft.com/en-gb/gg710912.aspx
2.CSA - CloudSecurityAlliance. CloudAdoptionPractices&PrioritiesSurveyReport. – 2015 р, Режим доступу: https://downloads.cloudsecurityalliance.org/initiatives/surveys/capp/Cloud_Adoption_Practices_Priorities_Survey_Final.pdf
3.Васильев В.И., Ланге Т.И., Баранов А.К. ИНТЕРПРЕТАЦИЯ РАЗМЫТЫХ ПОНЯТИЙ. - КДС 99 // VIII Международнаяконференция 1999 р.
4.Васильев В.И. Принцип редукции в задачах обнаруженияzакономерностей // Кибернетика и системныйанализ, №5, 2004г., с. 69-81.
5.Васильев В.И.Распознающиесистемы.- Наук. Думка, 1969р.
6.Радер, Р.І. MethodofPotentialFunctionasFeatureChoiceCriterioninAlphaProcedure // StatistischeWoche 2015 (Nachwuchsworkshop - семінар молодих вчених), Гамбург, Німеччина, 2015р.

УДК 004.051, 004.056

РІПНЕВСЬКИЙ О.О.
ВОЛОКИТА А.М.

ВИКОРИСТАННЯ ГІБРИДУ CPU/GPU У КРИПТОГРАФІЧНИХ ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЕННЯХ

В даному докладі розглянуто кластерні системи CPU/GPU,їхпереваги і недоліки, їхвикористання у системах криптографічногозлому.Демонструється один з зломщиків,щовикористовуєтаку структуру.

This article describes CPU/GPU cluster systems,their advantages and disadvantages,usability in cryptography crack systems.Example of such system demonstrated further.

Ключові слова:CPU,GPU,паралельні обчислення,хмарні обчислення
Keywords:CPU,GPU,parallelcomputing,cloudcomputing

Вступ

Розвиток систем з різноманітними архітектурами основного процесорного елементу, що раніше рухав індустрію у шлях, відомий нам зараз, потрохи починає відставити в зв’язку з технічними або практичними недоліками таких систем. Доцільним є використання найкращих рис різних систем за для найвищої продуктивності.

У час, коли завдання починають вимагати все більших потужнотей для адекватної роботи, використання застарілих та неефективних засобів не ефективно. Важливо також пам’ятати,що даних стає більше, томунеобхідно використовувати систему, що може масштабуватися в залежності від поставленої задачі, наприклад роботи з BigData. За останні час, хмарні технології та кластери стали дуже популярні, через необхідність в обробці значних об’ємів даних і їхню можливість до масштабування, разом з тим кластери мають мати високу надійність, доступність, і мати можливість швидко адаптуватися розмірами системи до задач.

Кластери

Актуальним є використання гібридних кластерів, тобто тих що складаються з декількох CPU,які працюють разом з декількома GPU. GPU дозволяють виконувати паралельні обчислювання, тому що більшість операцій підтримуються на рівні ядер. Це дозволяє отримувати

максимальну обчислювальну потужність, через відсутність обробників команд, та більш практичне використання комп'ютернихресурсів. Сьогодні використання таких технологій, як OpenGL або CUDA, дозволяє програмам працювати значно швидше, ніж на мультиядерних,чи навіть мультипроцесорних, системах.

Хмарні технології-це наступник традиційних кластерів та дата центрів. Основа роботи хмарних технологій полягає в тому, що непотрібно придбати дорогу мультипроцесорну систему, та надавати технічну підтримку і супроводження. Кінцевому споживачу потрібно платити за робочий час конкретноїмашини, яка може знаходитись в іншому кінці земної кулі. При цьому (згідно досвіду роботи автора з Amazon)при завантаженні значних об'ємів даних ціна виявиться значно менше ніж купівля апаратного забезпечення. Також на основі хмарних технології реалізуються різноманітні послуги, такі як зберігання даних в хмарному просторі і моніторинг різноманітних систем.Через свою маркетингову особливість, а саме оплату лише за використаний час,хмарні технології є дуже багато обіцяючою віхою комп'ютерного розвитку [1].

Ці технології є дуже практичними у розв'язанні складних обчислювальних задач, що можуть бути поділені на багато незалежних частин. Конкретно ця особливість дозволяє подолати парадигму SPMD(одна і та сама програма, що має працювати з декількома одиницями даних).

Це робить шифрування\дешифрування та відновлення паролів ділом простим і не тривалим.З іншої сторони гібридні обчислювання, що виконанні у хмарних середовищах, можуть бути використанні у криптографії і криптоаналізі.

Відновлення

У багатьох ситуаціях (наприклад, відновлення даних, або тестування на проникнення) необхідно відновити відкритий текст пароля, що шифрується за допомогою криптографічного одностороннього хешу. Однією з визначаючих характеристик криптографічного хешу, на відміну від не криптографічної хеш-функції (наприклад, CRC-32) є те, що він призначений для демонстрації сильного лавинного ефекту (зміна в одному біті на вході потягне за собоюзміну половину вихідних бітів)[2].

Деякі алгоритми хешування, такі як MD5, демонструють слабкі колізії: можна створити два повідомлення, які будуть мати однаковий хеш. Це, як правило, більш проста проблема, ніж так звана атака "знаходження прообразу" , де вхідне значення обчислюється на основі хешів до даного параметру. MDCrackє прикладом програми, призначеної для відновлення хешу паролів за допомогою перебору, що здатна використовувати багатоядерний паралелізм в багатопроцесорних системах, але не може працювати на більш ніж одній системи [3].

Пароль можна розглядати як *n* символів (з можливим повторенням) з "набору символів" *c*., тому існуєn^c можливих паролів. В середньому потрібна буде лише їхня половина[3]. Збільшення або *n* або *c* підніме число комбінацій в геометричній прогресії, як показано нижче:

Табл. 1 Залежність кількості комбінацій від n та c

c	*n*	$(N^c)/2$
26	6	154,457,888
26	7	4,015,905,088
26	8	9,885,304,832
52	7	104,413,532,288
52	8	514,035,851,264
62	8	109,170,052,792,448

Попередньо обраховані атаки, такі як веселкова таблиця, можуть здійснюватися проти деяких алгоритмів хешування. Багато додатків, наприклад,Unix-подібні операційні системи, використовують*хеш з сілью*(англ. salt): згенеровані випадковим чином значення, які не потрібно ховати, в поєднанні з паролем під час хешування, що збільшує кількість необхідного простору і часу, потрібного для генерації таблиці. Добре розроблений алгоритм *засолу*(англ. salting) може зробити перебір єдиним ефективним засобом відновлення паролю.

Швидкість перебору в 40 мільйонів хешів в секунду – цілком можливий результат для MD5 на помірно швидкій багатоядерній системі з використанням MDCrack, проте використання перебору на важкому паролі займе дуже багато часу. Для звичайного пароля з 8 символів, вибраних з символів A-Za-Z0-9, ми маємо $c = 62$ і $n = 8$. Це займе в середньому 2,729,251 секунд, або трохи більше місяця, щоб відновити один хеш. Деякі системи, такі як MD5crypt або GPG шифрування файлів ОС Linux / FreeBSD, виконують кілька ітерацій хеш-функції (зазвичай> 1000), щоб подолати можливість перебору - потенційно збільшуючи час відновлення для гіпотетичного паролю до 83 років [3].

Цю проблему можна розподілити шляхом розбиття простору пошуку між декількома системами. У теорії, лінійне масштабування може бути реалізоване через повну відсутність залежностей між блоками простору пошуку: 31 еквівалентний комп'ютер зможуть відновити MD5-хеш за один день, і шляхом додавання додаткових систем (або використання прискорення GPU), час відновлення потенційно може бути зменшено до декількох годин [4].

Приклад архітектури зломщика

У [5] використовується стандартний дизайн «майстер-пристрій»: центральний головний сервер, який відповідає за координацію спільних зусиль відновлення, а також один або кілька обчислювальних вузлів, які виконують фактичне відновлення. TCP-сокети використовуються для обміну даними між обчислювальними вузлами і майстром.

У табл.2 показано значний відрив GPU систем від CPU при виконанні однотипних завдань, таких як перебір хешів з веселкових таблиць.Головний сервер (написаний на C ++) виконує дві функції: він забезпечує користувацький інтерфейс, з якого процес відновлення запущений, і поділяє простір пошуку для кожного обчислювального вузла обробки.

Табл. 2 Порівняння швидкодії CPU та GPU

Система	Швидкість (мільйон хешів/сек)
P4 2.0 GHz	3.24
Core 2 Duo 2.2 GHz	16.19
GeForce 8600M GT	24.42
2x QuadXeon 2.0 GHz	55.33
Quadro FX 4600	102.26
GeForce GTX 260*	250

Під час ініціалізації, головний сервер створює робочу одиницю, в якому підтримує сокет-сервер для зв'язку з обчислювальними вузлами. Тим часом, щоразу, коли під’єднуєтьсяобчислювальний вузол, то створюється окремий потік для його обслуговування.

Майстер відстежує роботу кожного обчислювального вузла. Новий обчислювальний вузол,після закінчення опрацювання, може приєднатися до головного сервера в будь-який час.Коли нове завдання отримано, обчислювальний вузол обробляє його і породжує потік відновлення для кожного обчислювального пристрою (ядра процесора або CUDA GPU) в системі. Кожному потоку відновлення призначається рівна частка роботи в поточному випуску.

Висновок

В цій роботі було показановикористання веселкових таблиць на гібридній системі CPU/GPU.Ця система довела свою корисність для паралельних обчислень, так як гібридні обчислення є дуже важливими для криптографії та криптоаналізі.

Ці системи можуть бути швидко масштабованими і гарно використовують надані ресурси,тому їхній подальший розвиток є дуже важливим.

Перелік посилань

1.Karbowski, A. andNiewiadomska-Szynkiewicz, E. Parallelanddistributedcomputing.WarsawUniversityofTechnology Press,2009. – 458 c.
2.Kunzman, D. M. andKalé, L. V. . Programminghetero-geneousclusterswithacceleratorsusingobject-basedprogramming,ScientificProgrammingVolume 19 (2011), Issue 1, 2011, c.47-62
3.Oechslin, P. "Making a FasterCryptanalyticTime-MemoryTrade-Off", AdvancesinCryptology - CRYPTO 2003,2003,c.617-630
4.ScottHauckandAndréDeHon.ReconfigurableComputing, 1st Edition,2007,c. 944
5.AndrewZonenberg .DistributedHashCracker, RensselaerPolytechnicInstitute Journal,2009,c.395-399

УДК 004.27

РУДНИЦЬКИЙ М. В.
КЛИМЕНКО І.А.

МЕТОД АДАПТИВНОГО ПЛАНУВАННЯ В ДИНАМІЧНО РЕКОНФІГУРОВАНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

В статті запропоновано метод повторного використання апаратних ресурсів, що забезпечує підвищення продуктивності динамічно реконфігурованихобчислювальних систем за рахунок видалення часових затримок в процесі реконфігурації.Використання методу дозволяє мінімізувати накладні видатки реконфігурації,забезпечуючи значне прискорення в алгоритмах, що містять велику кількість однотипних задач. Виконано програмне моделювання запропонованого методу дляобчислювальних алгоритмів, поданих в ярусно-паралельній формі.

The hardware resources reuse method, providing improved performance and removing time delays during reconfiguration of dynamically reconfigured computer systems was proposed. The method usage allowsminimizing reconfiguration overheads, providing significant acceleration in the algorithms, containing a large number of similar tasks. Software simulation of proposed method was completed for presented in multilevel structure algorithms.

Ключові слова: реконфігурація, програмовані логічні інтегральні схеми (ПЛІС), реконфігуровані обчислювальні системи (РОС), метод повторного використання (МПВ), FPGA, ReconfigurableComputer.

Вступ

Основна особливість ПЛІС – можливість багаторазового перепрограмування. Ця властивістьє ключовою для створення пристроїв, архітектура яких має властивість реконфігурації з метою збільшення продуктивності. Такі системи належать до класу РОС, основна концепція їх розробки – забезпечення адаптивності архітектури до класів обчислюваних задач [1].

Але, разом із надзвичайною привабливістю, РОС мають ряд проблем, до яких належать великі накладні видатки на фізичне здійснення реконфігурації – непродуктивні витрати пам'яті та каналів зв'язку,часові та енергозатрати, тощо. Вони спроможні звести нанівець позитивний ефект як від адаптації архітектури, так і від апаратного прискорення. Найбільш критичними є витрати на реалізацію алгоритмів керуванняреконфігурації.

Вирішення проблеми зменшення накладних витрат реконфігурації зазвичай спрямовані на розробки спеціальних алгоритмів планування і розміщення задач. Найбільш розповсюдженими методами є попередня вибірка конфігурацій [2], повторне використання обчислювальних ресурсів [3], поєднання конфігурацій з метою зменшення реконфігураційних витрат [2]. Однак, більшість оглянутих рішень базуються на програмному або експериментальному моделюванні абстраговано від фізичної сторони реалізації, або розглядають доволі загальну архітектурну модель. Всі рішення покладаються на рівень програмної або програмно-апаратної надбудови операційної системи та мають високу складність реалізації алгоритмів управління, що негативно впливає на продуктивність обчислювальної системи.

Роботи, які пропонують архітектурні та технічні вдосконалення фізичного рівня РОС [3, 4], абстрагуються від засобів управління реконфігурацією. Ці апаратні способи прискорення реконфігурації зазвичай базуються на методах кешування й віртуальних моделях пам'яті та орієнтовані на збільшення пропускної здатності інтерфейсів.

Таким чином, всі відомі методи зменшення накладних видатків реконфігурації вбудовані в алгоритми планування й розподілу задач і зазвичай реалізовані на абстрактному рівні операційної системи засобами центрального процесора. Це знижує ефективність реконфігурації за рахунок додаткових витрат ресурсів на реалізацію алгоритмів управління та завантажує операційну систему.

Проблема зменшення накладних витрат реконфігурації в контексті досягнення високої продуктивності в РОС не може бути ефективно вирішена відомими методами і засобами. Це обумовлює високу актуальність і доцільність виконаних досліджень.

Формалізація методу адаптивного планування

Нехай виконувана програма у вихідному стані подана у вигляді ациклічного графу $G_M = (V, E)$, де V – множина вершин, в яких розміщуються обчислювальні задачі, а E – множина ребер, що визначають відношення між задачами. Тоді для подання графу програми в ярусно-паралельній формі (ЯПФ) зручно використати спосіб відображення задач на структуру реконфігурованої системи за рівнями, коли кожний рівень підграфу послідовно відображується на доступну обчислювальну структуру [5].

Визначимо сумарний час виконання задачі T_{Sum}, як суму двох складових, які впливають на загальну продуктивність обчислень:

$$T_{Sum} = T_{Count} + T_{Rconf},$$

де T_{Count} – час виконання задачі апаратними засобами, T_{Rconf} – час реконфігурації апаратної задачі на ПЛІС.

Складова T_{Count} визначає позитивний ефект від традиційних способів зменшення часу виконання задач (поглиблення паралелізму та апаратне прискорення). Складова T_{Rconf} є

непродуктивною складовою, яка визначає час, витрачений саме на процес реконфігурації апаратури. Саме ця складова обумовлює втрату продуктивності під час обчислень.

Час реконфігурації складається наступних узагальнених компонент: часу виконання програми реконфігураційного процесу, що забезпечується алгоритмами планування та розподілу задач ($T_{control}$); часу передавання данихконфігурації, що завантажуються із системної бібліотеки конфігураційних файлів (БК), із зовнішньої пам'яті конфігурацій(ЗПК) або із внутрішньої пам'яті ПЛІС (T_{comm}) та часу прошивання мікросхеми (T_{mapp}):

$$T_{reconf} = T_{control} + T_{comm} + T_{mapp}$$

Отже, для зменшення часу реконфігураціїпостає задача зменшення комунікаційних витрат. Для цього ефективним методом є повторне використання ресурсів апаратних задач, вже колись розміщених на поверхніреконфігурованої області (РО) ПЛІС.

Таким чином, основною задачею є виконання наступного положення:

$$T_{reconf} = \min(T_{control}) + \min(T_{comm}) + T_{mapp},$$

для рішення якої запропонований наступний метод повторного використання ресурсів апаратних задач.

Кожна задача є деякою функцією та характеризується параметром «Тип операції», який відповідає типу виконуваної функції. Функція, яку виконує задача, заздалегідь синтезована в цифрову схему і, як апаратна задача, збережена в центральній БК. Таким чином, параметр тип операції $I_j \mid j = \overline{1, I_{\max}}$, де $I_{\max}$ – кількість реалізованих операцій.

Під час реконфігурації апаратні задачі у вигляді функціональних блоків розміщуються на поверхні ПЛІС. Технологічні обмеження її площини та розміри конфігурацій обмежують кількість розміщуваних функціональних блоків (ФБ) значенням $F_{\max}$. Тоді кожний блок визначається номером $F_s \mid s = \overline{1, F_{\max}}$.

Конфігурація задач може відбуватися за різними послідовностями дій в залежності від місця знаходження конфігураційних даних і тривати відповідний до цього час:

I. Апаратна задача знаходиться у заздалегідь сконфігурованому вигляді на поверхні реконфігурованої області. Загальний час виконання задачі $T_{sum\,j}$ визначається як:

$$T^{\mathrm{I}}_{sum\,j} = T_{count\,j}, \qquad (1)$$

де $T_{count\,j}$ – час виконання апаратної задачі.

II. Конфігураційні дані апаратної задачі знаходиться в ЗПК. Загальний час виконання $T_{sum\,j}$ визначається як:

$$\begin{aligned} T^{\mathrm{II}}_{sum\,j} &= T_{reconf\,j} + T_{count\,j} = \\ &= T_{comm\,j} + T_{count\,j} \end{aligned}, \qquad (2)$$

де $T_{count\,j}$ – час передачі конфігураційних даних та завантажування конфігурації на ПЛІС, що визначається пропускною здатністю інтерфейсів вводу/виводу ПЛІС та швидкодією ЗПК.

III. Конфігураційні дані апаратної задачі знаходиться в БК. Загальний час виконання $T_{sum\,j}$ визначається як:

$$T_{sum\,j}^{\text{III}} = T_{reconf\,j} + T_{count\,j} = \\ = (T_{comm\_net\,j} + T_{comm\,j}) + T_{count\,j}, \qquad (3)$$

де T_{comm_net} – час пошуку та передавання конфігураційних даних із БК до ПЛІС мережевими засобами зв'язку.

Запропонований метод повторного використання апаратних ресурсів функціональних блоків (МПВ) забезпечує наступний час виконання послідовності задач:

$$T_B^{\text{МПВ}} = \sum_{j=1}^{K} (P_j T_j + \Delta T_j),$$

де складова P_jT_j відповідає сумарному часу виконання всіх екземплярів апаратної задачі I_j на поверхні РО, а складова ΔT_j відповідає часу її одноразової конфігурації.

Алгоритм роботи планувальника

Загальний алгоритм реалізації методу складається із наступних процесів, які ініціюються готовністю до виконання чергової обчислювальної задачі відповідно графу алгоритму та згідно реалізованого розкладу:

1. Пошук конфігурації відповідної апаратної задачі.Залежно від місця її розташування виконується розгалуження:
2. Запуск задачі на виконання, якщо апаратна задача вже розміщена на РО у вигляді ФБ.
3. Завантаження конфігураційних даних із ЗПК. Запуск задачі на виконання.
4. Завантаження конфігураційних даних із БК. Запуск задачі на виконання.
5. Якщо конфігурація шуканої апаратної задачі знайдена на поверхні РО, але вона використовується іншою обчислювальною задачею, приймається рішення про необхідність завантаження дубліката апаратної задачі на поверхню РО. Процесварто почати за виконання наступних умов:

- час виконання обчислювальної задачі в знайденому ФБ більший часу завантаження дубліката апаратної задачі із БК;
- апаратна задача часто повторюється.

Якщо приймається рішення про недоцільність завантаження дубліката апаратної задачі, система переходить в стан очікування звільнення необхідного ФБ.

Також запропонований метод вирішує задачу управління розкладом розміщення та підтримки конфігурацій апаратних задач.

Критерієм тривалості підтримки обирається частота викликів даної задачі під час виконання алгоритму обчислення, тобто чим частіше викликається задача, тим довше її апаратна реалізація буде підтримуватись на площині РО. Для забезпечення необхідної тривалості вводиться бонус підтримки.

Кожна апаратна задача, яка перший раз розміщується на поверхні ПЛІС, отримує найбільший бонус. Надалі, за кожним зверненням в пошуку конфігурації необхідної апаратної задачі бонус підтримки задач, які не знадобилися, декрементується, а бонус задачі, щоповторно використалась,інкрементується. Таким чином, більш використовувані задачі динамічно витісняють меншактуальні.

Апаратні задачі, що більше не мають бонусів, видаляються з поверхні РО. Зважаючи на те, що час повторного завантаження конфігурації із БК є критичним з точки зору збільшення накладних видатків реконфігурації, варто зберегти її конфігурацію в додатковій швидкодіючій ЗПК.

Програмні засоби реалізації методу

Симулятор РОС включає наступні блоки, що реалізовані як програмні модулі, які представляють функції відповідних апаратних блоків: *HardwareSystem* – головний модуль, що емулює функціонал контролера реконфігурації та включає підлеглі модулі: *FPGA* – реконфігурована область; *memory* – локальна пам'ять, що включає функціонал ЗПК; *bonuses* (підтримка бонусів). Модуль *HardwareSystem* виконує функції пошуку конфігурації задачі – *findConfiguration()* та завантаження задачі в ПЛІС – *load()*. Функція *findConfiguration()* повертає одне з трьох результатів: *TSK_FPGA*, *TSK_MEM*, *TSK_LIB*, значення яких встановлюються залежно від того, де було знайдено конфігурацію задачі.

РО представлена масивом об'єктів, що відповідають завантаженим в даний момент ФБ апаратних задач. Завантажений в ПЛІС функціональний блок представляється набором властивостей, що зображені в табл. 1.

Табл. 1 Слово стану ФБв ПЛІС

F_s	I_j	$T_{count\,j}$	$A_{S\,j}$	B_s	*Count*
Ідентифікатор функціонального блоку	Тип виконуваної операції	Скорегований час виконання	Просторова інформація	Бонус підтримки	Таймери та лічильники

ЗПК представлена множиною об'єктів, що відповідають апаратним задачам,що були витіснені з ПЛІС. В цій пам'яті апаратна задача представляється набором властивостей, що показані в табл. 2.

Табл. 2 Слово стану апаратної задачі в ЗПК

I_j	$T_{count\,j}$	A_j	B_s	C_j
Тип виконуваної операції (ідентифікатор задачі)	Скорегований час виконання (*correctTime*)	Адресна інформація	Ймовірність використання	Таймери

Модуль *lib*моделює БК та представлений масивом даних (*data*),що завантажується з файлу (*LIBRARY_FILE*). Модуль містить набір функцій, що забезпечують доступ до даних. Обчислювальні задачі у вихідному стані розміщені в центральній БКі представлені об'єктами з властивостями з табл. 3.

Табл. 3 Слово стану апаратної задачі в БК

I_j	$T_{\exp}$	$B_{\exp}$	D_j
Унікальний ідентифікатор задачі (тип операції*hwN*)	Очікуваний час виконання (*workTime*)	Розмір бітового потоку (*bytestreamWords*)	кількість вхідних даних (*dataCount*)

Моделювання роботи методу

Модуль (*simulator*) моделює функціонал РОС. Для оцінки часових залежностей під час моделювання методу використані наступні константи: час звернення до пам'яті –

MEMORY_ACCESS_TIME, час завантаження на мікросхему – *LOAD_LAST_TIME*, час завантаження одного слова даних – *LOAD_DATUM_TIME*, максимальний час для випадкової складової доступу в мережі – *NETWORK_MAX_RANDOM_TIME* та функцію моделювання – *simulate()*, результатом якої є діаграма Ганта в текстовому вигляді.

Як вже було сказано вище, розрахунок часу виконання T_j апаратної задачі відбувається за формулами (1), (2) та (3), де $T_{count\,j}$ – час виконання апаратної задачі I_j на поверхні реконфігурованої області, включаючи процеси вводу вихідних даних та виводу результатів (*LOAD_DATUM_TIME*), завантаження задачі (*LOAD_LAST_TIME*) і її виконання (*workTime* або *correctTime*); $T_{comm\,j}$ – час пошуку й передавання конфігураційних даних із ЗПК, що залежить від часу звернення до пам'яті (*MEMORY_ACCESS_TIME*) та розміру конфігураційного файлу (*bytestreamWords*); T_{comm_net} – час пошуку й передавання конфігураційних даних із БК мережевими засобами зв'язку, залежить від мережевої складової (*NETWORK_MAX_RANDOM_TIME*),розміру файлу конфігурації ічасу звернення до пам'яті (*MEMORY_ACCESS_TIME*).

Дослідження проводились для серії програм, поданих графами алгоритмів в ЯПФ. Досліджувались арифметичні програми з великою кількістю однотипних задач.Як видно з рис. 1 та рис. 2, найбільш ефективним з точки зору часу реконфігурації є повторне використання ресурсів ФБ, які заздалегідь сконфігуровані на поверхні ПЛІС.

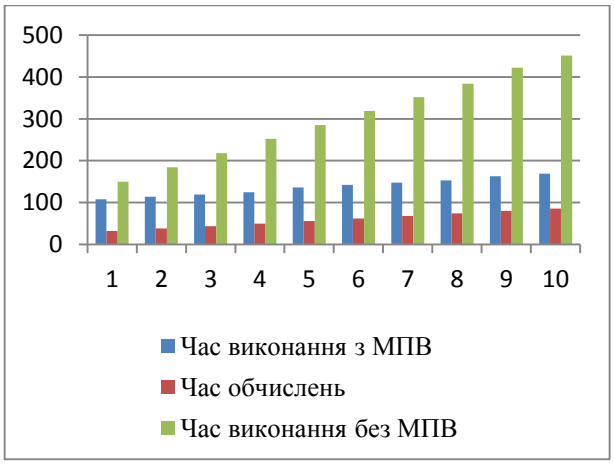


Рис. 1. Залежність часу виконання задач від кількості повторів однотипних задач

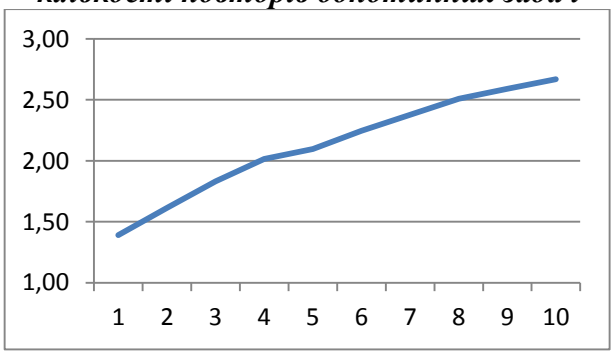


Рис. 2. Залежність коефіцієнту прискоренняві д кількості повторів однотипних задач

Висновки

На підставі проведених досліджень можна зробити наступні висновки щодо ефективності та практичної значимості запропонованих засобів прискорення реконфігурації:

- спосіб зберігання і підтримки конфігурацій апаратних задач за рахунок використання поверхні ПЛІС та засобів управління розкладом підтримки конфігурацій, забезпечує зменшення часу реконфігурації;
- метод прискорення реконфігурації за рахунок реалізації МПВ функціональних блоків на підставі запропонованого способу зберігання задач, дозволяє видалити практично всі непродуктивні витрати часу;
- організація багаторівневої пам'яті дозволяє реалізувати спосіб зберігання конфігурацій, а розроблені апаратні засоби управління розкладом розміщення та підтримки конфігурацій функціональних блоків дозволяють зменшити обчислювальну складність алгоритмів управління;
- емулятор РОС та програмна модель методу прискорення реконфігурації дозволяють в режимі реального часу моделювати процес управління реконфігурованими ресурсами, та є зручними інструментами для дослідження часових характеристик РОС.

Перелік посилань

1.Каляев И. А. Высокопроизводительные реконфигурируемые вычислительные системы на основе плис Virtex-6 и Virtex-7 / И. А. Каляев, А. И. Дордопуло, И. И. Левин, Е. А. Семерников // Параллельные вычислительные технологии (ПаВТ'2012) (Новосибирск, 26 – 30 марта 2012 г.). – Челябинск : Издательский центр ЮУрГУ, 2012. – с. 449 - 458.

2.El-Araby E. Exploiting Partial Runtime Reconfiguration for High-Performance Reconfigurable Computing / E. El-Araby, I. Gonzalez, T. El-Ghazawi // ACM Transactions on Reconfigurable TechnologyandSystems(TRETS).–USA,NY,NewYork,ACM,2009.–Vol.1,Issue4,Article№ 21.

3.Al-Wattar A. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems / A. Al-Wattar, S. Areibi, F. Saffih // Proceeding in 26th International Parallel andDistributedProcessingSymposiumWorkshops&PhDForum(IPDPSW).–IEEE,2012.–P401–406.

4.Liu S. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems / S. Liu, R.N. Pittman, A. Forin, J.-L. Gaudiot // Transactions on Embedded Computing Systems (TECS). – USA, NY, New York, ACM, 2013. – Volume 12, Issue 3, Article № 72. – 21 p.

5.Кулаков, Ю. О. Організація багаторівневої пам'яті в реконфігурованих обчислювальних системах: зб. наук. пр. / Ю. О. Кулаков, І. А. Клименко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+,2014. – № 61. – С. 18–26.

УДК 004.771

САВИНА К.О.
КУЛАКОВ Ю.О.

СПОСОБ ПОСТРОЕНИЯ ВИРТУАЛЬНЫХ СЕТЕЙ В СИСТЕМАХ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

В настоящей статье рассмотреныинструменты виртуализации VMware,Citrix и Microsoft и их основные характеристики. Представлено сравнение данных инструментов с дальнейшим определением лучшего из них, а также приведен анализ преимуществ и недостатков на основании таблицы оценки по основным критериям.

The subject of the article is the virtualization tools from VMware, Citrix and Microsoft corporations and their main characteristics. A comparison of these instruments with further determination the best of them is provided, as well as an analysis of advantages and disadvantages based on the evaluation table on the main criteria.

Введение

Суть концепции облачных технологий заключается в предоставлении конечным пользователям удаленного динамического доступа к услугам, вычислительным ресурсам и приложениям (включая операционные системы и инфраструктуру) через интернет. Развитие сферы хостинга было обусловлено возникшей потребностью в программном обеспечении и цифровых услугах, которыми можно было бы управлять изнутри, но которые были бы при этом более экономичными и эффективными за счет экономии на масштабе.

Большинство сервис-провайдеров предлагают облачные вычисления в форме VPS-хостинга, виртуального хостинга, и ПО-как-услуга(SaaS). Облачные услуги долгое время предоставлялись в форме SaaS, например, MicrosoftHostedExchange и SharePoint.

Основные характеристики

Важнейшими характеристиками вычислительных центров являются [1]:

- *Изоляция*. Архитектура должна обеспечивать эффективную изолированность между различными сетями заказчика. Это включает поддержку их частных адресных пространств, которые потенциально могут быть перекрывающимися, и изолированный трафик. Распределение ресурсов должно управляется таким образом, что клиенты не могут повлиять на использование ресурсов друг друга неконтролируемым образом.
- *Прозрачность*. Инфраструктура и аппаратные средства, лежащие в основе центра обработки данных, должны быть прозрачными для пользователя. Каждый клиент должен иметь логический вид своей собственной сети, независимо отфактической реализации. Это упрощает администрирование для клиента и повышает безопасность.
- *Географическая независимость*. Виртуальные машины (VM)и сети клиентов должны быть " независимы от расположения", т.е., можгутбыть физически распределены в любом месте центра обработки данных. Это может значительно улучшить использование и упростить выделение ресурсов.
- *Простота управления политикой*. Каждый клиент может иметь свои собственные политические требования и требования безопасности. Архитектура должна позволить клиентам настроить индивидуальные параметры политики на лету, и обеспечить соблюдение таких параметров в сети. [2]
- *Масштабируемость*. Количество клиентов, которые могут поддерживаться, должно быть ограничено только ресурсами, имеющимися в центре обработки данных, а не искусственными представлениями конструкции.
- *Бюджетность*. Решение должно в основном опираться на имеющиеся в наличии устройства, чтобы снизить новые материальные вложения для поставщиков облачных услуг.

Инструменты виртуализации

Для поддержки облачных вычислительных сред используется технология виртуализации.

Наиболее известными инструментами виртуализации являются CitrixXenServer, VMwarevSphereи Hyper-V компании Microsoft. Каждый из них имеет свои преимущества и недостатки.

Веб-сервис https://www.whatmatrix.com предлагает полную сравнительную таблицу характеристик с начислением очков каждому из инструментов в зависимости от степени поддержки того или иного сервиса. [3] Основываясь на данном сравнении, можно дать указанным инструментам виртуализации такую оценку:

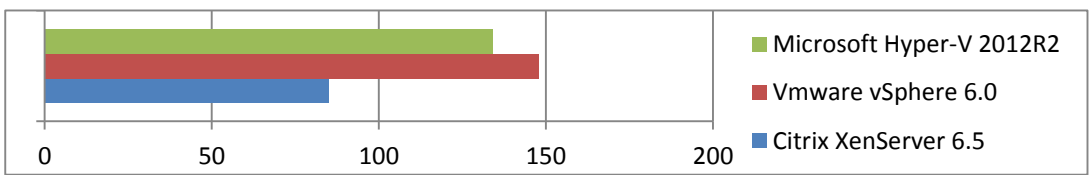


Рис. 1 – Оценка инструментов виртуализации согласно степени поддержки различных сервисов

Более подробно с категориями оценивания и количеством баллов по каждой из них можно ознакомиться в табл. 1.

Таблица 1. Оценка инструментов виртуализации по категориям

Category	XenServer / 6.5	vSphere / 6.0	HyperV / 2012R2
Total	85	**148**	134
General	**7**	5	4
Management	16	**38**	37
VM Mobility and HA	14	**23**	21
Hypervisor	22	**39**	29
Network and Storage	26	**43**	**43**

Согласно системе оценивания данного сервиса, лидером является компания VMware, чейvSphere получил первенство практически во всех категориях.

Преимущества и недостатки

Основными преимуществами vSphere стали наибольшая поддержка поставщиков и наибольшее количество функций, возможность управления с помощью браузера, шаблоны виртуальных машин с возможностью авторазвертывания, увеличение доступности виртуальных машин (достигается с помощью автоматического перезапуска в случае ошибки), улучшенная система мониторинга приложений, поддержка bootableUSB, репликация данных и многое другое.

Все же, и у этого инструмента есть свои слабые места: высокая первоначальная стоимость, отсутствие преднастроенного портала самообслуживания, отсутствие встроенной поддержки хранилищ SAN (есть возможность ее включения с помощью плагина VMwareSAN 6.0). [4]

Основным же недостатком инструмента vSphere стала аппаратная несовместимость: несмотря на то, что совместимости vSphere продолжает расширяться, другие средства виртуализации имеют более широкую аппаратную поддержку. Средства виртуализации корпорации Майкрософт имеют самый большой набор поддерживаемых устройств. [5]

Являясь составным компонентом системы WindowsServer 2008 R2, гипервизор Hyper-V R2 поддерживает те же самые драйверы. [6] Несмотря на то, что VMware расширяет номенклатуру своих изделий и также добавила поддержку съемных модулей памяти, количество устройств, поддерживаемых системами Windows все-таки больше.

Заключение

В данной статье рассмотрены инструменты виртуализации ведущих производителей, таких как VMware,Microsoft и Citrix. Рассмотрены ключевые характеристики оценки систем облачных вычислений, а также проведено сравнение указанных инструментов VMwarevSphere, MicrosoftHyper-V и CitrixXenServer с приведением таблицы оценок каждого из них по основным категориям оценивания.

В результате анализа преимуществ и недостатков инструментов виртуализации можно сделать вывод, что в данный момент лидером является vSphere от компании VMware, имея лишь незначительные недостатки по сравнению с конкурентами. В случае, когда необходима наиболее полная аппаратная совместимость, лучшим выходом станет использование Hyper-V от Microsoft.

Списоклитературы:
1.Fang Hao, T.V. Lakshman, Sarit Mukherjee, Haoyu Song. Secure Cloud Computing with a Virtualized Network Infrastructure [Электронныйресурс]. – Режимдоступа: http://pages.cs.wisc.edu/~akella/CS838/F12/838-CloudPapers/SEC2.pdf
2.Шайхутдинов А.М. Критерии сравнения облачных технологий [Электронный ресурс]. – Режим доступа: http://sci-article.ru/stat.php?i=1425487858
3.Productevaluationreport[Электронный ресурс]. – Режим доступа: https://www.whatmatrix.com/comparison/Virtualization#
4.Software Insider. Virtualization tools comparison [Электронныйресурс]. – Режимдоступа:http://virtualization.softwareinsider.com/compare/7-9-40/Citrix-XenServer-Free-Edition-vs-Microsoft-Hyper-V-Server-2008-R2-SP1-Standard-vs-VMware-vSphere-Enterprise-Edition
5.Виртуализация сети в Hyper-V. Концепция [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/company/microsoft/blog/173159/
6.Сравнение производительности IaaS-экосистем. Часть 5: MicrosoftAzureVirtualMachines технологий [Электронный ресурс]. – Режим доступа: http://cloudzone.ru/community/blogs/iaas/303.html

УДК 004.724

СІНЯКОВ М.В.
КУЦ В.Ю.

СИТУАЦІЙНА МАРШРУТИЗАЦІЯ В ДИНАМІЧНИХ МОБІЛЬНИХ МЕРЕЖАХ

У даній статті розглянуті динамічні мобільні мережі на прикладі бездротових mesh-мереж, фактори їх оптимізації, наведено приклад протоколу маршрутизації B.A.T.M.A.N., що був розроблений для динамічних мереж. Також розглянута оптимізація цього протоколу та наведені результати цієї оптимізації.

The subject of the article is dynamic mobile networks using mesh networks as an example, factors for their optimization, B.A.T.M.A.N protocol developed for dynamic networks. The optimized routing algorithm is proposed for the protocol.

Динамічні мобільні мережі

Мобільні мережі є мережами, що працюють у відкритому просторі, тому для дистанційної комунікації генеруються шляхи з великою кількістю хопів. Одним з найбільш ефективних властивостей такої мережі є кооперативне взаємодія з сусідніми вузлами. Ці мережі, як правило, знають про стан сусідніх вузлів, що забезпечує ефективну комунікацію в режимі реального часу[1]. Вони дозволять зберігання й аналіз історії зв’язку для виконання адаптивної маршрутизації. На такі мережі сильно впливають перешкоди та обмеження, що мають враховуватися для побудови ефективного маршруту. Основними вимогами до генерації маршрутів у мобільних мережах є відстань, енергозатратність, завантаженість мережі, втрачені пакети та затримка комунікації.

Час зв'язку, затримка також залежить від відстані, яку проходить пакет. Але практично завжди найкоротший маршрут не є оптимальним, необхідно розглядати й інші властивості мережі. Якщо вузли працюють від резервних батарей, їхній запас потужності стає вирішальним для генерування маршруту, тому необхідно генерувати енергоефективні маршрути. Ще одним критичним фактором є завантаженість конкретного вузла, найбільш завантажений вузол ніколи не є ефективним. Інші параметри оптимізації маршруту включають аналіз швидкість передачі даних, аналіз затримки на зв'язок і аналіз втрат пакетів.

Ще одне міркування в ідентифікації маршруту є сценарій зв'язку. В такому випадку виконується ситуаційна маршрутизація. Ситуаційна маршрутизація – це інтелектуальний підхід маршрутизації, який включає в себе ефективне формування маршруту на основі вимог сценарію.Також є підходи маршрутизації, залежні від розташування, енергетичної потужності та інші.

Протокол B.A.T.M.A.N.

B.A.T.M.A.N. – протокол маршрутизації, що розробляється як заміна OLSR[2].Основною особливістю B.A.T.M.A.N. є децентралізація відомостей про кращий маршрут в мережі – жоден вузол не володіє всіма даними. З використанням цієї техніки відпадає необхідність у поширенні інформації про зміни в мережі на всі вузли. Кожен вузол зберігає інформацію тільки про «напрямок», з якого надходять дані, і так само їх поширює. Таким чином, вузли передають один одному пакети по динамічно створюваним маршрутами.B.A.T.M.A.N. не намагається визначити весь маршрут, а тільки перший крок пакета в потрібному напрямку[4]. Дані пересилаються сусідові в цьому напрямку, який використовує той самий механізм. Процес повторюється, поки дані не досягнуть мети.

Оптимізація протоколу B.A.T.M.A.N.

Ця робота описує оптимізацію протоколу B.A.T.M.A.N. для динамічних mesh-мереж. Суть проблеми полягає в тому, щоб оптимізувати протокол маршрутизації, так щоб він міг адаптуватися і швидко реагувати на швидкі і динамічні топологічні зміни. Для оптимізації будуть використовуватися підходи ситуаційної маршрутизації на основі оцінки якості зв’язку для досягнення кращої пропускної здатності.

З урахуванням рухливості мобільних вузлів, швидкі зміни топології в динамічній мережі неминучі.Таким чином, ідеальний підхід полягає в урахуванні поточної ситуації мережі при прийнятті рішень про маршрутизації. B.A.T.M.A.N. визначає найкращий зв'язок такий, що має найбільшу кількість успішно доставлених контрольних повідомлень за короткий проміжок часу (вікно)[3]. Це не завжди вірно, бо може бути обраний зв'язок, який на початку вікна був активним, а потім перестав працювати. Наприклад, за вікно було передано 10 контрольних повідомлень. Впродовж цього вікна по зв’язку 1 були передані 5 повідомлень на початку вікна [1111100000], по зв’язку 2 були передані 4 контрольні повідомлення, але наприкінці вікна [0000001111].

Вага зв’язку оцінюється формулою (1):

$$\sum_{i=1}^{n} a_i \,, \qquad (1)$$

де n – кількість очікуваних контрольних повідомлень впродовж оцінюваного вікна, a_i= 1 для повідомлення, що надійшло, 0 – для повідомлення, що не надійшло. Тобто оцінка першого зв’язку буде рівною 1+1+1+1+1+1+0+0+0+0 = 6, а другого – 0+0+0+0+0+0+1+1+1+1 = 4, тобто V.A.T.M.A.N. вважатиме перший зв'язок кращим, бо впродовж вікна він передав більше контрольних повідомлень, але насправді другий зв'язок буде кращим вибором, бо через нього контрольні повідомлення находили щойно, і має бути обраний другий зв'язок.

В цій роботі пропонується надати більший пріоритет останнім контрольним повідомленням, і у наведеному прикладі обрати другий зв'язок. Найпростіший спосіб пріоритизації – надати вагу кожному контрольному повідомленню тим більшу, чим щодавніше воно прийшло, наприклад рівною номеру контрольного повідомлення за рахунком упродовж конкретного вікна, для якого виконується оцінка каналів, як показано у формулі (2).

$$\sum_{i=1}^{n} a_i * i \qquad (2)$$

Як видно з формули, кожне наступне повідомлення вноситиме більше значення у вагу зв’язка ніж попереднє, наявність першого вносить 1, останнього – n. У прикладі, що розглядається, вікно складається з 10 контрольних повідомлень, тому останнє повідомлення матиме вагу 10. Тому повідомлення, що прийшли нещодавно, а отже показують поточний стан зв’язку, матимуть більший пріоритет за ті, що були отримані на початку.

У цьому випадку перший зв'язок, що розглядається в прикладі, матиме вагу 1+2+3+4+5+0+0+0+0+0 = 15, тоді як зв'язок 2 матиме вагу 7+8+9+10=34. Ця вага дає більш точне уявлення про якість зв’язку у цій ситуації.

Щодо зв’язків, повідомлення в яких приходили приблизно всередині вікна, запропонована оптимізація враховуватиме їх так само коректно, надаючи їм перевагу над зв’язкам, контрольні повідомлення в яких проходили лише на початку вікна, проте зв’язки з контрольними повідомленнями, що отримані наприкінці вікна, матимуть більший пріоритет.

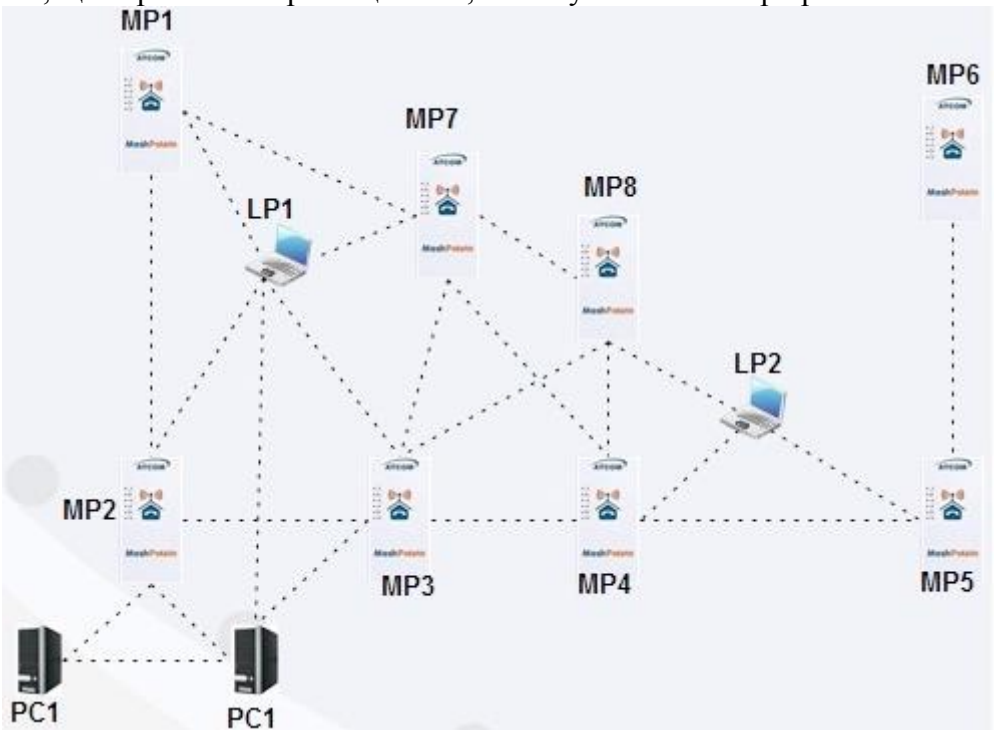


Рис. 1. Схема mesh-мережі

Спосіб, що було запропоновано, був протестований на тестовому стенді, схема якого зазначена на рис. 1. Ціль експерименту – порівняти оригінальний протокол B.A.T.M.A.N. та запропоновану модифікацію. Для генерації та моніторингу трафіку використовувалася утиліта Iperfз різними налаштуваннями.

Табл. 1. Кількість втрачених пакетів

	1M-41K	100M IK	150M-1K	150M-11K	1M-31K	1M-11K	Середнє
BATMAN							
LP1	0,17	0,20	0,12	0,18	0,22	0,14	0,173
LP2	0,77	1,93	1,39	3,3	3,47	0,82	1,946
BATMAN mod							
LP1	0,02	0,03	0,03	0,018	0,042	0,040	0,029
LP2	0,14	0,03	0,05	0,11	0,04	0,15	0,085

Як видно за табл.1, кількість втрачених пакетів на пристроях з безпровідним підключенням зменшилася у декілька разів, пристрій з провідним підключенням майже не показав зміну кількості втрачених пакетів.

Табл. 2. Пропускна здатність

	1M-41K	100M-1K	150M-1K	150M-11K	1M-31K	1M-11K	Середнє
BATMAN							
LP1	0,08	0,08	0,02	0,09	0,08	0,08	0,074
LP2	0,08	0,08	0,08	0,006	0,07	0,08	0,066
PC2	0,04	0,08	0,09	0,09	0,09	0,09	0,081
BATMAN mod							
LP1	0,08	0,08	0,09	0,09	0,08	0,09	0,084
LP2	0,08	0,08	0,09	0,08	0,09	0,08	0,083
PC2	0,12	0,09	0,09	0,08	0,083	0,08	0,090

Середня пропускна здатність збільшилась. Отже оптимізація працює і може бути використана в реальних мережах.

Висновок

У цій роботі розглянуто динамічні мобільні мережі та запропоновано оптимізацію для протоколу B.A.T.M.A.N., яка показала кращі результати за оригінальний алгоритм при проведенні тесту.

Перелик посилань

1. Mesh networking – Wikipedia, https://en.wikipedia.org/wiki/Mesh_networking
2. B.A.T.M.A.N. – Wikipedia, https://en.wikipedia.org/wiki/B.A.T.M.A.N.
3. D. Johnson, N. Ntlatlapa, C. Aichele, “Simple pragmatic approach to mesh routing using BATMAN,” //2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa, 2008
4. A. Munaretto, H. Badis, K. Al Agha, and G. Pujolle, “QoS-enhanced OLSR protocol for mobile ad hoc networks” //Proceedings of ANWIRE 1st International Workshop, Glasgow Scotland, 2003, pp. 171–183.

УДК 004.383

СЕРГІЄНКО А.М.
СЕРГІЄНКО А.А.

БІБЛІОТЕКА МОДУЛІВ ДЛЯ ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР’Є

Розглянуто набір модулів для *r*-точкового швидкого перетворення Фур’є (ШПФ) за алгоритмом Віноградa. Модулі розроблені методом відображення просторового графу синхронних потоків даних у апаратуру, який забезпечує мінімізовані апаратні витрати за рахунок зменшення пропускної здатності у *r* разів. Модулі призначені для побудови швидкісних конвеєрних процесорів ШПФ на базі ПЛІС.

A set of soft IP cores for the Winograd *r*-point fast Fourier transform (FFT) is considered. The cores are designed by the method of spatial SDF mapping into hardware, which provides the

minimized hardware volume at the cost of slow-down of the algorithm by *r* times. The cores are intended for the high-speed pipelined FFT processors, which are implemented in FPGA.

Ключові слова: алгоритм Віноградa, швидке перетворення Фур'є, ПЛІС.

Вступ

Конвеєрні процесори швидкого перетворення Фур'є (ШПФ) за основою, яка не дорівнює 2^k, k = 1,2, використовуються у швидкісних модемах, для обробки радіосигналів та ін. Велика множина таких процесорів реалізується в програмованих логічних інтегральних схемах (ПЛІС). Недоліками існуючих конвеєрних процесорів ШПФ є необхідність узгодження потоків даних між їх ступенями та високі апаратні витрати [1,2].

У роботі розглядається проектування конвеєрних модулів для процесорів ШПФ за взаємно-простими основами, що конфігуруються в ПЛІС.

Модулі для малоточкового ШПФ

Конвеєрний процесор ШПФ зі взаємно-простими основами має менші апаратні витрати і придатний для обробки послідовностей з іншою довжиною, ніж 2^k. Причому обчислення ШПФ для коротких послідовностей ефективно виконувати за алгоритмом Віноградa [2,3]. При цьому виникає проблема узгодження потоків даних між модулями малоточкового ШПФ, оскільки група даних, яка видається одним модулем, не співпадає за числом даних та їх порядком з групою даних, яка примається іншим модулем [1]. Для узгодження двох модулів з паралельним виводом n даних та паралельним вводом m даних між ними слід встановити $n \cdot m$ блоків буферної пам'яті.

Найпростіше ця проблема вирішується тоді, коли за один такт блок видає чи сприймає лише одне дане. У цьому випадку період L-точкового ШПФ має дорівнювати L тактів, а не один, як у процесорів, описаних в [1,2]. Тоді для організації покрокового введення та виведення даних з таких блоків вони повинні, по-перше, мати на входах і виходах відповідні буферні схеми, по-друге, виконувати обчислення лише в кожному L-му такті.

При використанні методики побудови просторового графу синхронних потоків даних (ГСПД) такі етапи синтезу конвеєрного процесора, як вибір ресурсів, складання розкладу, призначення операцій на ресурси і одержання структури виконуються за один крок, що дає змогу краще оптимізувати шукану конвеєрну структуру [4]. При цьому мінімізується як кількість суматорів, так і складність мультиплексорів на їх входах. Саме за цією методикою була розроблена бібліотека модулів для малоточкового ШПФ. Кожен модуль виконує L-точкове ШПФ за алгоритмом Віноградa з періодом L тактів.

Приклад синтезу модуля ШПФ

Розглянемо приклад розробки модуля для r = 3-точкового ШПФ, який виконує алгоритм Віноградa, описаний в роботі [4]:

$t = x_1 + x_2$; $p = x_2 - x_1$; $m_0 = x_0 + t$;

$m_1 = (\cos(2\pi/3) - 1)\cdot t$; $m_2 = j\cdot\sin(2\pi/3)\cdot p$;

$s = m_0 + m_1$;

$X_0 = m_0$; $X_1 = s + m_2$; $X_2 = s - m_2$;

де $\{x_0, x_1, x_2\}$, $\{X_0, X_1, X_2\}$ – комплексні дані та результати, $j = \sqrt{-1}$, причому $m_1 = -1{,}5\cdot t$.

Для мінімізації числа блоків множення та збільшення тактової частоти множення на коефіцієнт виконується через зсуви та додавання множеного. Так,

$$k = \sin 2\pi/3 = 0{,}866025 = 1.0010\ 0010\ 0101\ 01.$$

Тоді множення $k\cdot p$ виконується як

$$k\cdot p = p - (p + p2^{-4})2^{-3} + ((p2^{-2} - p)2^{-2} - p)2^{-10}.$$

Просторовий ГСПД для цього алгоритму з періодом обчислень L = 3 з урахуванням множення на коефіцієнт показано на рис.1. Тут вершина додавання показана як коло з хрестиком, невелике коло означає затримку на такт, великі крапки – вершини вводу і виводу, дуга, навантажена

значенням “>>*k*” означає передачу операнду зі зсувом вправо на *k* розрядів. Координати вершини (*s, t*) означають номер *s* процесорного елементу (ПЕ), в якому виконується даний оператор і номер такту *t* виконання. На рис. 1 координати *s* є умовними – вони позначають назву ПЕ, в який відображається віповідна операторна вершина.

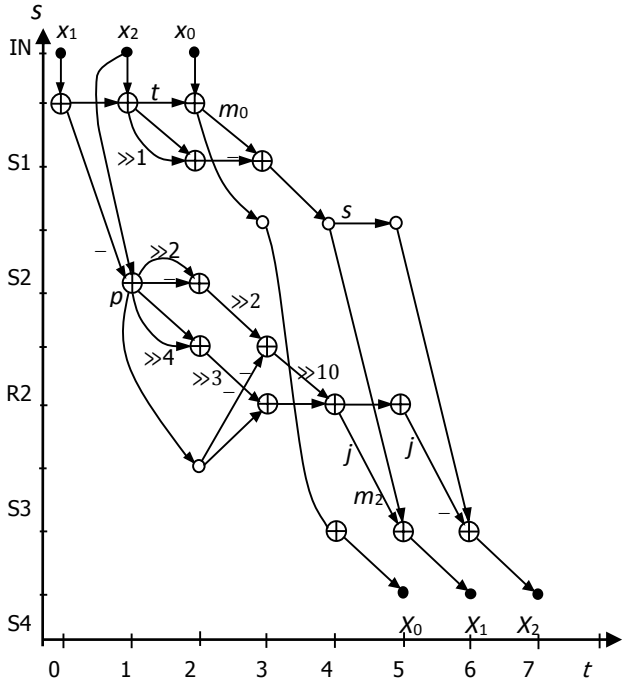


Рис. 1. Просторовий ГСПД для 3-точкового ШПФ

ГСПД на рис.1. відображається в структуру модуля, яку показано на рис.2. Насправді, структури модулів ШПФ не будуються. Замість цього, просторовий ГСПД описується мовою VHDL за методикою, представленою у роботі [4]. Тоді такий опис приєднується до проекту процесора ШПФ і компілюється загальновживаним компілятором-синтезатором для ПЛІС.

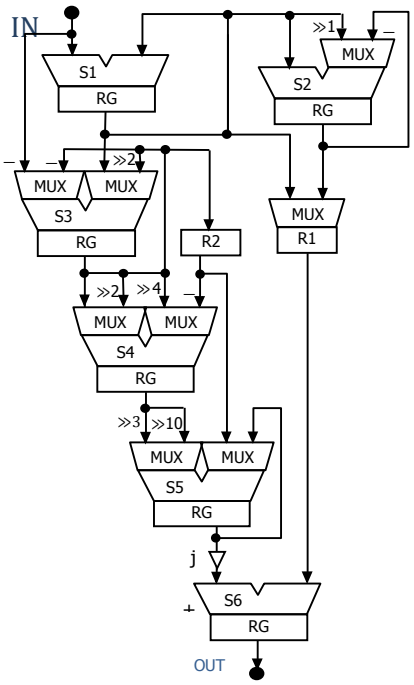


Рис. 2.Структура модуля ШПФ

Результати реалізації модулів

Модулі малоточкового ШПФ описані мовою VHDL і мають мінімальну кількість регістрів та суматорів. Причому вхідні та вихідні дані слідують у потрібному порядку.

У таблиці показані результати синтезу модулів для ПЛІС серії Xilinx Kintex-7 для 16-розрядних вхідних даних. Аналіз таблиці показує, що використання спеціалізованих блоків множення на константу дає змогу збільшити тактову частоту до 1,6 разів у порівнянні зі структурою з застосуванням інтегрованих блоків множення DSP48. Тактова частота модулів дорівнює частоті дискретизації сигналу. Вона є доволі високою і зменшується зі збільшенням складності модуля. Це роз'яснюється тим, що через складність розміщення та трасування на затримку у лініях зв'язку припадає 70-80% від загальної затримки у критичному шляху.

Розроблені модулі ШПФ, які описані мовою VHDL, зібрані у бібліотеку. Додатково інтерфейс кожного модуля описано мовою XML у форматі стандарту IP-XACT.

Результати конфігурування модулів у ПЛІС

Кількість точок	Апаратні витрати, ЛТ + DSP48	Тактова частота, МГц	
3	245	640	
3	201 + 2	433	
4	215	548	
5	945	435	
8	1187	424	
15 = 3·5	2131	346	
16	3616	368	
64 = 8·8	1985 + 4	338	
128 = 8·16	5277 + 4	324	

Такий опис вміщує метадані про модулі, що спрощують їх інтеграцію у систему у багатьох САПР ПЛІС, які підтримують стандарт IP-XACT. Це такі метадані, як імена портів вводу-виводу, їх розрядність, код ідентифікації модуля, опис алгоритму, який реалізовано у модулі, ім'я файлу опису модуля, графічне зображення модуля, описане мовою SVG тощо. Наприклад, параметр nb, який задає розрядність даних, описується як:

```
<spirit:parameters>
 <spirit:parameter maximum="32" minimum="8"
     type="int">
    <spirit:name>nb</spirit:name>
 <spirit:displayName>nb</spirit:displayName>
    <spirit:value spirit:id=
          "uuid_cd7f1ae6_bfb2">16</spirit:value>
    <spirit:vendorExtensions>
       <kactus2:vector>
            <kactus2:left>15</kactus2:left>
            <kactus2:right>0</kactus2:right>
       </kactus2:vector>
    </spirit:vendorExtensions>
 </spirit:parameter>
</spirit:parameters>
```

Тоді структурну схему конвеєрного процесора ШПФ можна зібрати у графічному редакторі САПР, наприклад, в системі Kactus2, скомпілювати у VHDL-опис на структурному рівні та

сконфігурувати у ПЛІС будь-якої підходящої серії та виробника. Слід додати, що Kactus2 – це проект з відкритим кодом, розроблений в університеті м. Тампере і оснований на стандарті IP-XACT. В табл. вказані результати синтезу таких процесорів ШПФ для кількості точок 64 і 128.

Висновки

Реалізація *r*-точкових модулів в ПЛІС забезпечує проектування високошвидкісних конвеєрних процесорів ШПФ з оптимізованим об'ємом апаратних витрат. Показано, що модуль ШПФ, який виконує *r*-точкове перстворення за *r* тактів має високу тактову частоту і невеликий обсяг апаратних витрат за рахунок конвеєрних обчислень, властивості 6-вхідних ЛТ, а також застосування спеціалізованих блоків множення. Розроблені модулі ШПФ використані для створення конвеєрних процесорів ШПФ з довжиною перетворення 64, 128, і 256.

Виконується розробка WEB-застосування, яке забезпечує автоматичний синтез конвеєрних процесорів ШПФ на основі розроблених модулів з використанням стандарту IP-XACT. Деякі розроблені модулі ШПФ знаходяться у відкритому доступі для випробування та використання [5].

Перелік посилань

1. Lofgren J., Nilsson P. On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems // Proc. Norchip. –2011. –P. 1–4.
2. Quershi F., Garrido M., Gustafsson O. Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on Winograd Fourier transform algorithm // Electronic Letters. –V.49. –N.5. –2013. –P.348 – 349.
3. Нусбаумер Г. Быстрое преобразование Фурье и алгоритмы вычисления сверток. –М.: Радио и связь. –1985. –248 с.
4. Сергиенко А. М., Симоненко В. П. Отображение периодических алгоритмов в программируемые логические интегральные схемы // Электрон. моделирование. – 2007. – Т. 29, № 2. –С. 49–61.
5. Sergiyenko A., Usenkov O. Pipelined FFT/IFFT 128 points processor. – Режим доступу: http://opencores.org/project,pipelined_fft_128

УДК 004.383

СЕРГІЄНКО П.А.
ЛЕПЕХА В.Л.

ЯДРО 16-РОЗРЯДНОГО RISC-ПРОЦЕСОРА

Розглянута розробка шістнадцятирозрядного ядра RISC-процесора, реалізованого в ПЛІС, яке має високу швидкодію та невеликі апаратні витрати. Система команд ядра адаптована для виконання алгоритмів компресії та керування.

A soft IP core of the 16-bit RISC microprocessor for the FPGA implementation is considered, which distinguished by low hardware volume, and high speed. The core instruction set is adapted to implement the lossless compression, and control algorithms.

Ключові слова: ПЛІС, RISC, LISA, хеш-функція.

Вступ

Програмовані логічні інтегральні схеми (ПЛІС) дають змогу реалізувати процесори зі спеціалізованою системою команд, завдяки чому мінімізуються апаратні витрати, енергоспоживання і зростає продуктивність. У [1] показана ефективність реалізації 8-розрядного мікропроцесора у ПЛІС, у якого мінімізована система команд. Тепер розробка процесорних ядер зі спеціалізованою системою команд підтримується засобами САПР, такими як Synopsys Processor Designer, який використовує методику LISA [2].

Рівень розвитку ПЛІС дає змогу реалізувати в них великі багатопроцесорні системи [1,2]. Але нові серії ПЛІС мають ряд особливостей, що впливають на вибір архітектури модулів, які в них конфігуруються. У нових поколіннях ПЛІС, якщо число транзисторів збільшується учетверо, то об'єм ресурсів для трасування сигналів збільшується лише удвічі. Затримки у лініях зв'язку стали в 1–3 рази більшими за затримки у логічних таблицях (ЛТ) та інших логічних елементах. Як результат, конфігуровані модулі мають бути компактними, щоби їх внутрішні сигнали далеко не поширювались.

Вимозі компактності не відповідають ядра мікропроцесорів з поширеною архітектурою, такі як Nios, Microblaze, Mico32, OpenRISC, MIPSFPGA [3]. Для досягнення прийнятної тактової частоти ці процесори мають п'ять та більше ступенів конвеєра команд. Для виконання кількох незалежних програмних потоків у такому ядрі реалізуються система переривання та зміни контексту, віртуальна пам'ять, кеш-ОЗП, швидкісний інтерфейс до зовнішньої динамічної пам'яті.

Множина архітектур таких процесорів обмежена 8- та 32-розрядними процесорами. Крім того, більшість процесорів орієнтована на виконання кількох обчислювальних процесів, включаючи роботу операційної системи. На виконання цих процесів потрібні суттєві часові та апаратні витрати [3].

Метою роботи є побудова процесорного ядра яке займає мало місця в ПЛІС і в той же час є швидкодіючим та має зручну систему команд для виконання логічних операцій над послідовностями символів. Для цього був розроблений 16-розрядний процесор RISC-ST2, який є процесорним елементом такої системи.

Особливості архітектури процесорного ядра

Архітектура процесорного ядра RISC-ST2 орієнтована на виконання однієї задачі, яка має багато логічних операторів та обробку потоків даних з полями змінної довжини. Це, наприклад, пакування даних за алгоритмом LZW. При необхідності виконання кількох паралельних задач кожна з них може виконуватись на окремому ядрі. Кілька ядер можуть бути об'єднані у систему через свої регістри вводу-виводу та систему переривань.

За основу було взяте ядро RISC-ST, яке описане в [4]. При формуванні структури приймались до уваги рекомендації проектування процесорів для ПЛІС, які приведені у [1]. Довжина команди була збільшена до 18 розрядів, що дало змогу оптимізувати систему команд та краще задіяти об'єм вбудованої пам'яті ПЛІС. До ядра додано команди обробки окремих бітів слів, виділення бітових полів заданої довжини, злиття полів, зсуву слів, підрахунку числа нульових старших розрядів. Для реалізації доступу до асоціативної таблиці введено команду обчислення хеш-функції.

На рис. 1 показана структура процесора. Товстими лініями виділені регістри, які належать до ступенів конвеєра. Завдяки тому, що конвеєр команд є трьохступеневим, більшість команд виконується за один такт, а команди переходу та читання пам'яті – за два такти. Причому при виконанні команд переходу та тривалого доступу до пам'яті даних конвеєр призупиняється. Виклик підпрограм виконується також за два такти, під час чого адреса повернення та прапорці умов зберігаються у апаратному стеку. Зарезервовано 16 векторів переривань.

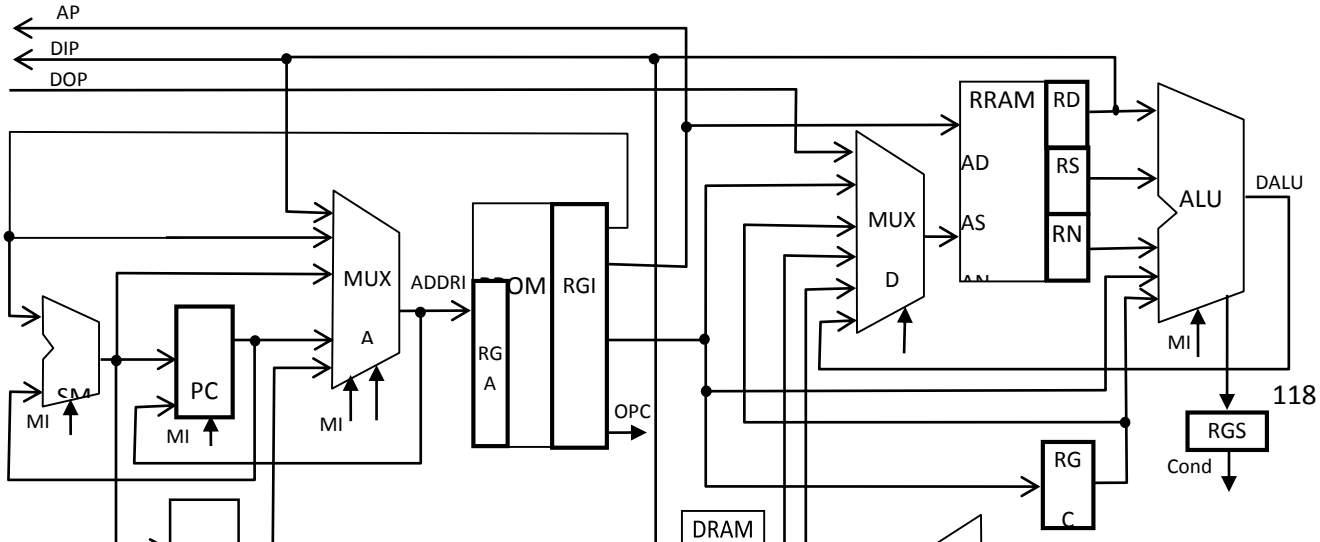


Рис.1. Структура процесорного ядра RISC-ST2

Регістрова пам’ять RRAM процесора має 32 регістри. Причому командно доступними є 16 молодших регістрів. 16 старших регістрів доступні після перемикання банку регістрів чи входу у режим переривання, причому вони мають адреси від 8 до 15. Щоби максимально задіяти можливості регістрової пам’яті, яка реалізована на ЛТ, у процесорі протягом одного такту виконується читання з трьох регістрів та запис у один регістр. При цьому операнд, на місце якого записується результат, зберігається у додатковому регістрі і може бути збережений у регістровій пам’яті за наступною командою.

Використовуються такі види адресації, як регістрова, непряма регістрова, базова, індексна з преінкрементом.

Адресно доступні до 256 периферійних регістрів. Вони використовуються як регістри вводу-виводу спецпроцесорів, які виконують швидкісні обчислення, як наприклад, обчислення елементарних функцій, цифрову обробку сигналів, шифрування.

Пам’ять даних розбита на сторінки по 256 байт і має максимальну ємність 16 мегабайт. Для доступу до пам’яті даних використовується стандартний відкритий інтерфейс Wishbone.

Результати реалізації ядра

Ядро мікропроцесора описане мовою VHDL і не має обмежень для синтезу та конфігурування у ПЛІС будь-якої серії. Модель процесора має вбудований дизасемблер, який спрощує тестування та відлагодження програм. Розроблено програму кросасемблера на мові Java, виходом якої є VHDL-файли блоків пам’яті даних та програм процесора.

У таблиці 1 показані результати синтезу процесора для різних серій ПЛІС та їх порівняння з відомими 16-розрядними процесорами.

Табл.1. Результати конфігурування процесорів у ПЛІС

ПЛІС	Апаратні витрати, ЛТ	Тактова частота, МГц	Прим.
Xilinx Kintex-7	508	237	
Xilinx Artix-7	513	163	
Xilinx Spartan-6	643	144	
Xilinx Spartan-3	861	77	
Altera Max-10	1212	121	
Altera Cyclone-V	495	161	

Altera Stratix-V	546	235	
Xilinx Kintex-7	672	142	1
Xilinx Kintex-7	1387	150	2
Xilinx Spartan-3	3602	74	3

Примітки: 1. Процесор RISC-ST2 з ускладненими командами. 2. Процесор OpenMSP430 [5]. 3. Процесор NanoBlaze [3].

За статистикою, у сучасних ПЛІС на один блок пам'яті припадає, в середньому, 300 ЛТ. Отже, процесор має доволі невеликі апаратні витрати – стільки, скільки припадає на 2 – 3 блоки пам'яті. Ці витрати є значно меншими, ніж у аналога та у семеро разів менше, ніж у поширеного 32-розрядного MicroBlaze [3]. Наприклад, у порівняно дешевій ПЛІС Artix-35T вміщується до 45 процесорних ядер.

Тактова частота процесора з доданими складними командами є нижчою на 67% від частоти ядра з мінімальним набором команд. До таких складних команд належить команда циклу DJNZ, сигнали при виконанні якої проходять критичним шляхом структури ядра. Але такі команди замінюють собою кілька простіших команд і спрощують програмування.

Висновки

Розроблено ядро RISC-процесора зі спеціалізованою системою команд. Завдяки тому, що процесор орієнтований на виконання лише одного обчислювального процесу та обробки переривань, він має невеликі апаратні витрати та високу швидкодію. Він призначений для виконання алгоритмів безвтратної компресії та керування. При зміні системи команд, яка є нескладною, процесор може бути використаний для інших потреб, наприклад, для цифрової обробки сигналів, розпізнавання образів тощо. Сергієнко А.М., Лепеха В.Л. Деякі особливості проектування мікроконтролерів для СНК / А.М. Сергієнко, В.Л. Лепеха // Вісник НТУУ «КПІ», сер. Інформатика, управління та обчислювальна техніка. – Т. 50. – 2009. – с. 70-73.

Перелік посилань

1. Synopsys Processor Designer. Automating the Design and Implementation of Custom Processors and Programmable Accelerators// Synopsys, Inc. – 2010. – 3 P. –Available at www.synopsys.com.
2. Meyer-Baese U. Digital Signal Processing with Field Programmable Gate Arrays. 4-th Ed. – Springer. – 2014. – 930 p.
3. Сергиенко А.М. VHDL для проектирования вычислительных устройств. –К.: –"ДиаСофт". –2003. –210 с.
4. OpenMSP430 / O. Girard // – Rev. 1.15, May, 19, 2015. –129 P. – Availabe at www.opencores.org

УДК 004.4'6

СНІХОВСЬКИЙ В.Л.
ПОРЄВ В.М.

АНАЛІЗ ОБМЕЖЕНЬ ДОСТУПНИХ ОБСЯГІВ ПАМ'ЯТІ У ГІС

В цій статті розглядаються існуючі програмні та апаратні обмеження, що стосуються використання великих обсягівфізичної пам'яті, які слід враховувати у реалізації алгоритмів

геоінформаційних систем. Зібрані дані про популярні апаратні платформи та операційні системи, під керуванням яких може працювати потенціальна ГІС. Також розглянуте питання збереження та організації доступу до великих обсягів даних на зовнішніх носіях.

This article is a review of actual software and hardware limitations on physical memory volume which should be considered in the implementation of GIS algorithms. It contains the data about hardware platforms and operating systems which can run potential GIS.The problem of storing large amounts of data on external storages and providing access to it was addressed.

Вступ

Геоінформаційні системи звичайно працюють з великими об'ємами інформації та відповідно потребують великі об'єми пам'яті для реалізації алгоритмів їх обробки.

Растрові моделі просторових даних засновані на способах квантування простору за допомогою регулярних сіток, кожен елемент яких характеризується певним набором даних. За допомогою такого способу подання даних можлива формалізація просторово-безперервної інформації, властивої більшості природних і значній кількості антропогенних об'єктів [1].

Для представлення тривимірної просторової інформації доцільно використовувати воксельну модель. Воксельна модель - це тривимірний растр. Подібно до того, як пікселі розташовуються на площині двовимірного зображення, вокселі утворюють тривимірні об'єкти в певному обсязі [2]. Воксель- це елемент об'єму.

Наприклад, для опису одного шару поверхні площею 100 км^2 (10км х 10 км) з розмірністю сітки 1 м^2 потрібно 100000000 клітинок. Якщо кожна клітинка 8 В даних, в цілому лише для збереження необхідно виділити

$$8 * 10^8\ B \approx 0.745\ GiB$$

У випадку тривимірного моделювання, за умови врахування підпростору висотою 100 метрів над поверхнею, об'єм даних збільшиться у 100 разів та становитиме

$$0.745 GiB * 100 = 74\ GiB$$

Звісно, просторові алгоритми доцільно застосовувати і до значно більших об'ємів даних. За допомогою засобів ГІС можна моделювати погоду, поширення сторонніх речовин у повітрі, пожарів, тощо на великих територіях. Для обробки таких обсягів даних необхідний об'єм пам'яті може збільшитись у 2-3 рази.

Обмеження програмного забезпечення

Операційні системи, під керуванням яких можуть працювати ГІС, звичайно накладають певні обмеження (інколи штучні) на максимальний об'єм оперативної пам'яті. Це стосується як загального об'єму, так і віртуального адресного простору, що може бути наданий одному процесу.

Обмеження x86-систем.Віртуальний адресний простір таких систем фізично обмежений 4 GiB (через обмеження значення адреси у 32 біта). Це значення є максимальним для одного процесу в системі, що зумовлено архітектурою і не може бути збільшено (існують технології, які дозволяють виділяти процесу декілька віртуальних просторів, проте вони не здобули популярність через присутність на ринку 64-бітних систем). Загальний об'єм пам'яті у системі може бути збільшений до 64 GiB за допомогою технології PAE.PhysicalAddressExtension (PAE) — режим роботи вбудованого контролера пам'яті x86-сумісних процесорів, в якому використовуються 64-бітні елементи таблиць сторінок з 36-бітною адресацією [3].

Обмеження x86-64 систем. Теоретично такі системи можуть використовувати усі переваги 64-бітного адресного простору, але на практиці часто застосовуються певні обмеження. Порівняльну характеристику деяких популярних операційних систем надано у таблиці 1.

Табл. 1. Можливості операційних систем з використання пам'яті [4][5][6].

ОС	**Об'єм віртуального адресного простору 1 процесу**		**Максимальний об'єм доступної пам'яті**	
	x86	x86-64	x86	x86-64
Windows 7 Home Basic	до 3 GiB	8 TiB	4 GiB	8 GiB
Windows 7 Professional	до 3 GiB	8 TiB	4 GiB	192 GiB
Windows Server 2008 Enterprise	до 3 GiB	8 TiB	64 GiB	1 TiB
Windows Server 2008 R2 Foundation	-	8 TiB	-	8 GiB
Windows Server 2008 R2 Standard	-	8 TiB	-	32 GiB
Windows Server 2008 R2 Enterprise	-	8 TiB	-	2 TiB
Windows 10 Home	до 3 GiB	8 TiB	4 GiB	2 TiB
Windows 10 Pro	до 3 GiB	8 TiB	4 GiB	128 GiB
Windows Server 2012 Foundation	-	128 TiB	-	32 GiB
Windows Server 2012 Essentials	-	128 TiB	-	64 GiB
Windows Server 2012 Standard	-	128 TiB	-	4 TiB
FreeBSD	до 4 GiB	128 TiB	64 GiB	4 TiB
RedHatEnterprise Linux	до 4 GiB	128 TiB	64 GiB	12 TiB

Обмеження апаратного забезпечення

Обмеження x86-систем. У таких системах суттєво обмежений віртуальний адресний простір (до 4 GiB), що спричинює значні труднощі в обробці великих обсягів даних. За апаратної підтримки технології PAE можливе збільшення максимального об'єму пам'яті до 64 GiB, але обсяг пам’яті, доступної одному процесу, не буде перевищувати 4 GiB. Також обмеження накладають можливості чипсетів, деякі системи не дозволять встановити більше 4 GiB оперативної пам'яті. У багатьох системах взагалі відсутня можливість встановити додаткову пам'ять.

Використання 32-бітної архітектури є доволі рідким у сучасних системах. Звичайно це мобільні та енергозберігаючіпристрої. Прикладом сучасних процесорів, що базуються на 32-бітному наборі логіки, є лінійка IntelAtom. Сучасні чипи Atom, призначені для використання у серверах, також підтримують 64-бітні інструкції. Незважаючи на це, у таких системах використовується 36-бітна адресація, тому максимальний об'єм оперативної пам'яті обмежений 64 GiB [7].

Обмеження x86-64 систем. Більшість сучасник високопродуктивних систем використовують 64-бітний набір команд. Теоретично максимальний об'єм адресного простору у таких системах становить 16777216 TiB (2^{64}), але на даний момент такий об'єм недоступний. Хоча така адресація дійсно можлива у рамках прийнятого набору інструкцій, виробники центральних процесорів з метою економії використовують скорочений адресний простір (на даний момент більшість x86-64 процесорів використовують 48-бітну адресацію, тобто

контролери пам'яті оперують обсягами до 256 TiB). Збільшення цього значення до максимального теоретичного не призведе до втрати сумісності, але спричинить значні росходи на утримання збільшених таблиць сторінок пам'яті. В архітектурі AMD64 закладена можливість збільшення ліміту фізичної адресації до 2^{52}, на відміну від віртуального адресного простору [8].

Порівняння можливостей сучасних чипсетів за об'ємом пам'яті наведено у таблиці 2 (з урахуванням існуючих материнських плат). Існують також багатопроцесорні системи, що дозволяють встановити більшу кількість модулів пам'яті, які не увійшли до таблиці.

Табл. 2. Характеристики сучасних чипсетів [9].

Чипсет	Intel						AMD	
	Z77	H110	Z170	X79	X99	C612	990FX	SR5690
Максимальний об'єм пам'яті	32 GiB	32 GiB	64 GiB	128 GiB	128 GiB	1.5 TiB	128 GiB	1 TiB

Збереження даних

До задач ГІС також входить збереження величезних обсягів даних, які не обов'язково повинні бути завантажені у пам'ять. Шари просторової інформації часто обробляються окремо, крім того, алгоритми ГІС звичайно можуть бути виконані на деякій порції даних (в обмеженому регіоні). Для забезпечення належної надійності підсистеми збереження даних та збільшення її ємності може виникнути необхідність використання розподілених сховищ.

Існує багато технологій, що допоможуть вирішити цю задачу. Вони відрізняються характеристиками швидкості, надійності та складності організації. Розглянемо деякі рішення.

RAID 0 / RAID 6 збільшує надійність збереження даних за рахунок їх дублювання на різних носіях або додавання контрольної інформації для відновлення. Швидкий, але всі носії працюють в одній машині. RAID 0 відрізняється високою надлишковістю.

NFS. Дані розподілені у мережі. Збільшує кінцеву ємність системи збереження даних за рахунок поєднання вузлів у мережу. Дублювання даних на рівні NFS відсутнє. Мінус – швидкість залежить від якості мережі.

DRBD. Фактично – RAID 1 у мережі. Забезпечує відмовостійкість за рахунок дублювання інформації на різних носіях, що об'єднані мережею.

Ці інструменти є універсальними та досить абстрактними, і можуть бути використані у побудові будь-якої розподіленої системи. Але для досягнення належного рівня ефективності доцільно буде створити свою підсистему збереження даних (адже збереження просторових даних – одна з основних функцій ГІС), що буде зберігати просторові дані у зручних для обробки структурах та задовольняти поставленим вимогам масштабованості та швидкодії.

Висновки

Сучасні апаратні та програмні засоби, доступні на ринку, накладають суттєві обмеження на доступний об’єм ресурсів. Для забезпечення належної швидкодії та відмовостійкості важливо спроектувати та розробити масштабовану систему, що дозволить зберігати дані у розподіленому середовищі та паралельно їх оброблювати на багатьох вузлах. Варто врахувати умови, у яких буде працювати ПО; обирати спеціалізовані серверні продукти для спрощення задачі.

Перелік посилань

1.Капустин Г.А. Растровые модели пространственно распределенных данных. Второй семинар Проблемы ввода и обновления пространственной информации, - Москва, 1997.

2.Порев В. Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002. -432с.: ил.
3.T. Shanley. Pentium Pro and Pentium II System Architecture. Addison-Wesley Professional, 1998. - p. 439.
4.Memory Limits for Windows and 5.6.Windows Server Releases [Електронний ресурс]: Заг. з головної сторінки. Режим доступу: https://msdn.microsoft.com/ en-us/library/windows/desktop/aa366778%28v=vs.85%29.aspx
5.Frequently Asked Questions for FreeBSD 9.X and 10.X [Електронний ресурс]: Заг. з головної сторінки. Режим доступу: https://www.freebsd.org/ doc/faq/hardware.html#idp60519376
6.Red Hat Enterprise Linux technology capabilities and limits[Електронний ресурс]: Заг. з головної сторінки. Режим доступу: https://access.redhat.com/articles/rhel-limits
7.Intel Atom Processor C2750Спецификации [Електронний ресурс]: Заг. з головної сторінки. Режим доступу: http://ark.intel.com/ru/products/77987/Intel-Atom-Processor-C2750-4M-Cache-2_40-GHz
8.AMD64 Programmer's Manual Volume 2: System Programming, order number 24593, revision 3.14, Advanced Micro Devices, September 2007.
9.PickParts. BuildYour PC.[Електронний ресурс]: Заг. З головної сторінки. Режим доступу: http://pcpartpicker.com/parts/motherboard/

УДК 004.021

СОЗІНОВ Є.В.

СПОСІБ РОЗМЕЖУВАННЯ ПРАВ ДОСТУПУ КОРИСТУВАЧІВ ДЛЯ WEB-САЙТУ

У статті розглянута розробка способу розмежування прав доступу користувачів для підвищення ефективності адміністрування та простоти впровадження yweb-сайти. Запропоновано метод розв'язання даної задачі, а також наочний, поетапний приклад його реалізації.

Inthearticlethewayofdevelopmentpermissionsofuserstoincreaseadministrativeefficiencyandease ofimplementationinwebsites. Themethodofsolvingthisproblem, aswellasvisual, step-by-stepexampleofitsimplementation.

Огляд головної ідеї

Спосіб розмежування прав доступу складається з груп та дій.

({Group} + {Actions} або {Group} - {Actions})

Group - це набір імен, які з себе представляють:

1) Права конкретного користувача (наприклад "user1', 'user2' ...). Наприклад використовується для особистих повідомлень, до яких цей користувач має доступ або для допуску редагування тільки його повідомлень на сайті.

2) Групи приватних сторінок (або груп користувачів), до яких необхідно дати права на певні дії. (Наприклад, Адміністратори, Супермодератори та ін.)

3) Додаткові властивості. (Наприклад, прапор - перемикач режимів)

Actions— набір дій, які користувачі з наявними *{Group}* можуть робити. У нашій системі «Підбору товарів» можна використовувати:

A - додавати новий товар

D - видаляти товар

E - редагувати товар

V - бачити товар

C - залишати коментар
B - видаляти коментар

«±»— означає давати користувачеві з такими правами або не давати (пріоритетно) доступ до дії. Наприклад: *Users + VC, Users -C*.Тепер розглянемо приклад прав доступу, для сайту з підбору товарів у магазинах:

1) Об’єкт MainProductPage:
a. *Users+VC,*
b. *Moderator+AEDB,*
c. *Admin+AEDB*
2) Об’єкт Product:
a. User1+ED,
b. Users-C
3) Об’єкт ProductComment:
a. *User2+B*

За замовчуванням на головній сторінці виділяються права 1){a,b,c}. Але, об`єкт 2){a,b} має теж свої права, тому кінцеве розмежування для певного товару буде мати наступний вигляд (на прикладі доступу до певного товару для групи *Users*):[1]

Users+VC-C = Users+V

Спочатку користувачі, за замовчуванням можуть бачити товар та залишати коментар, але для певного товару 2) у якому є свої права, користувачі зможуть тільки бачити цей товар.[3]

Приклад реалізації для розуміння комп`ютером

На прикладі списку декількох прав можна почати з такої БД, яка представлена у вигляді таблиць, що зазначені нижче.

Права доступу для об’єктівMainProductPageТаблиця 1

id	rights_id	group	sign	action
1	100	Users	+	V
2	100	Users	+	C
3	100	Moderator	+	A
4	100	Moderator	+	E
5	100	Moderator	+	D
6	100	Moderator	+	B
7	100	Admin	+	A
8	100	Admin	+	E
9	100	Admin	+	D
10	100	Admin	+	B

Дана таблиця має наступні поля:

- **rights_id**— ідентифікатор списку прав
- **group**— назва групи
- **sign** — знак операції
- **action**— назва дії

Права доступу для об’єктів Product Таблиця 2

id	rights_id	group	sign	action
11	101	User1	+	D
12	101	User1	+	E
13	101	Users	-	C

Права користувачів ***Таблиця 3***

id	rights_id	group
1	10	User1
2	10	Users
3	10	Moderator

Далі потрібно додати права користувачам, виходячи з цього можна з`ясувати чи має користувач право на дію. Для цього необхідно створити таблицю прав, яка має поля:[4]

- rights_id — ідентифікатор списку прав користувачів
- **group** — назва групи в якій знаходиться користувач

Тепер нормалізуємо ці таблиці для їх розуміння комп`ютером. У результаті вийдуть три таблиці, та зникнуть id ключі. Також замінимо знаки операцій (+) на 1 та (-) на 0, тому що так буде легше звертатися до них. Ідентифікатор group_idвказує на назву у таблиці **rights_names.[2](див. Таблиця 4)**

Назви груп **rights_names** ***Таблиця 4***

group_id	name
10	Users
11	Moderator
12	Admin
1001	User1
1002	User2
1003	User3

Ця таблиця слугує лише для інформативності результатів та не впливає на роботу загальної схеми та має два поля - ідентифікатор та ім`я групи.

Права користувачів **rights_groups*****Таблиця 5***

rights_id	group_id
1	1001
1	10
1	11

Ця таблиця повинна бути прив'язана до користувачів і вказує на те, які користувач має права.

Права об`єктів **rights_action*Таблиця 6***

rights_id	group_id	sign	action
100	10	1	product_view
100	10	1	comment_create
100	11	1	product_create
100	11	1	product_edit
100	11	1	product_delete
100	11	1	comment_delete
100	12	1	product_create
100	12	1	product_edit
100	12	1	product_delete
100	12	1	comment_delete
101	1001	1	product_edit
101	1001	1	product_delete
101	10	0	comment_create

Ця таблиця повинна бути прив'язана до об'єктів і каже, з яким правами, які дії користувач може виконувати.

Створивши таку БД, комп'ютеру, який оперує числами, стало набагато простіше працювати з таблицями.Додати права можна записуючи дію у таблицю у вигляді рядка, який має тип ENUM (поле ’action’), що значно спрощує розуміння і розробку проектів.

Висновок

На простому прикладі був розглянутий спосіброзмежування прав доступу користувачів до web-сайту, який забезпечує розмежування будь-якого контенту на сайті, а також є простий для розуміння. Тому даний спосіб легко інтегрувати у сайт будь-якої складності.

Перелік посилань

1. Дронов В. А. 2004. - PHP, MySQL и Dreamweaver MX 2004. РазработкаинтерактивныхWeb-сайтов285-290.
2. СтанславГорнаков2009 - Осваиваемпопулярныесистемыуправления сайтом (CMS) 322-330
3. Portela, IreneMaria 2009 - InformationCommunicationTechnologyLaw, Protectionand Access Rights
4. RonLepofsky 2014 - TheManager'sGuidetoWebApplicationSecurity

УДК 004.056.5

СТАДНЮК Є.В.
ВОЛОКИТА А.М.

МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО ПОШУКУ ДАНИХ НА ПРИКЛАДІ АГЕНТСТВА З НЕРУХОМОСТІ

В даній статті розглянуті методи інтелектуального пошуку даних, представлені алгоритми інтелектуального пошуку, проведено порівняльний аналіз звичайного пошуку даних та інтелектуального. Також показані переваги даного методу та спосіб його реалізації для агентства з нерухомості.

The subject of the article is a methods of intelligent search of data, submitted intelligent search algorithms, comparative analysis of native (keywords) search and data mining. Also shown the advantages of this method and the way to implementation for the real estate agency.

Ключові слова: пошук, дані, алгоритм.
Ключевые слова: поиск, данные, алгоритм.
Keywords:search, data, algorithm.

Вступ

В сучасному світі задача пошуку та пошукових запитів полягає не лише в простому пошуку даних, а й в можливості дати користувачу альтернативну інформацію за його запитом. Такі пошукові гіганти як Google, Yandex, Yahoo вже давно не використовують пошук лише по ключовим словам, а використовують інтелектуальні методи пошуку, щоб дати користувачу більш якісну інформацію.

Методи інтелектуального пошуку у WEB

Довільний інформаційний WEB-ресурс, як правило, надає засоби пошуку інформації. Найбільш розповсюдженим є пошук за ключовими словами, але останнім часом дуже стрімко почав розвиватись інтелектуальний пошук за смисловим змістом інформації. Основними недоліками пошуку за ключовими словами, є наступні:[1]

1) пошук за ключовими словами не враховує числові параметри документа, за якими можна робити фільтри та сортування результатів пошуку;

2) простий пошук за ключовими словами не враховує категорії слів, які близько відносяться до предметної області, по якій проводиться пошук. Такі категорії слів найбільш часто зустрічаються в пошукових запитах.

3) пошук за ключовими словами не враховує семантичні характеристики –відмінки та семантичні зв’язки;

4) пошук за ключовими словами орієнтований на предметну область, а не на користувача.

Існуючі методи інтелектуального пошуку вирішують розглянуті недоліки пошуку за ключовими словами:

1) врахування важливості ключових слів, близьких до предметної області, вирішується за допомогою створення в базі даних спеціальних груп синонімів для таких ключових слів і подальшого врахування їх пошуковою системою;

2) врахування числових параметрів документа вирішується за допомогою надання спеціального шаблону для створення документа. Окремі поля цього шаблону будуть відповідати числовим даним, і ці дані будуть зберігатися у внутрішній базі даних в числовому вигляді, а показуватися користувачеві у вигляді звичайного текстового документа;

3) кожен користувач може створити персонального пошукового інтелектуального агента, і, таким чином, орієнтувати пошук не на предметну область, а на свої особисті дані.[4]

Методи інтелектуального пошуку для виявлення шкідливого програмного забезпечення

Аналіз ситуації щодо шкідливого програмного забезпечення показує динамічне зростання кількості вірусних програм, серед яких опосередковане місце займає клас вірусів — троянські програми, які здатні виконувати на персональному комп'ютері деструктивні або шкідливі дії. Розробники троянських програм розробляють нові способи потрапляння на персональні

комп'ютери, знаходять можливість маскування від сканерів антивірусного програмного забезпечення та постійно вдосконалюють код троянських програм. В свою чергу, розробники антивірусного програмного забезпечення постійно проводять вдосконалення методів пошуку та виявлення троянських програм, оновлюють антивірусні бази, застосовують передові технології виявлення. Найпоширенішими методами виявлення троянських програм в антивірусних засобах є сигнатурний аналіз, метод контрольних сум та евристичний аналіз.[1]

Принцип роботи сигнатурного аналізу визначає границі його функціональності — можливість виявляти лише вже відомі віруси. Недоліком даного методу також є необхідність з'єднання з Інтернет для оновлення антивірусних баз. Відновлення пошкоджених файлів, виявлених за допомогою сигнатурного аналізу, неможливе, оскільки троянська програма часто є окремим об’єктом.

Метод контрольних сум має суттєві недоліки: можлива наявність однієї і тієї ж контрольної суми для різних файлів, відчутне використання системних ресурсів для знаходження контрольної суми файлів.

Евристичний сканер виконує порівняння зразків поведінки шкідливих програм із поведінкою досліджуваного об'єкту та нараховує «бали», якщо поведінка програми схожа на один із зразків. Сучасні евристичні аналізатори побудовані з використанням нейронних мереж. Проте в цьому методі присутній високий відсоток невірного спрацювання, тобто виникає ситуація, коли бали нараховуються не шкідливій програмі. Евристичний аналізатор працює досить повільно, використовує значні ресурси персональних комп’ютерів при виконанні підозрілих програм у режимі віртуальної машини та може помилково позначати файл як троянську програму, в той час як це може бути цілком корисне програмне забезпечення чи його частина.

Враховуючи недосконаліть існуючих методів пошуку троянських програм, необхідним є розроблення та впровадження нових методів шляхом використання технологій та алгоритмів інтелектуальної обробки інформації, які б підвищили достовірність та ефективність антивірусного діагностування персональних комп’ютерів.

Алгоритми інтелектуального пошуку. Жадібний алгоритм

Жадібний алгоритм – це алгоритм, що полягає у прийнятті локально оптимальних рішень на кожному етапі, допускаючи, що кінцеве рішення також виявиться оптимальним.

Завдання, що вирішуються жадібними алгоритмами, характерні дві особливості: по-перше, до них застосуємо «Принцип жодного вибору», а по-друге, вони мають властивість «Оптимальності для підзадач».

«Принцип жодного вибору» застосовується, якщо послідовність локально оптимальних виборів дає глобально оптимальне рішення. У типовому випадку доказ оптимальності слід такою схемою:

1) Доводиться, що жадібний вибір на першому кроці не закриває шляху до оптимального рішення: для всякого рішення є інше, узгоджене з жадібним вибором і не гірше першого.

2) Показується, що підзадача, що виникає після жодного вибору на першому кроці, аналогічна вихідної.

3) Міркування завершується за індукції – методом доведення для всіх натуральний чисел. [1]

Задача має властивість «Оптимальності для підзадач», якщо оптимальне рішення задачі містить в собі оптимальні рішення для всіх її підзадач.

Алгоритми інтелектуального пошуку. Алгоритм сегментації

Алгоритм сегментації – це алгоритм, який ділить дані на групи чи кластерні елементи, які мають схожі властивості. Найчастіше цей алгоритм використовують для виділення потрібної інформації із зображення, проте його можна використовувати і в інших сферах.

Виділяють наступні типи алгоритму сегментації:

1) Порогові алгоритми;

2) Алгоритм нарощування областей;

3) Граничні алгоритми;
4) Алгоритми сегментації на основі кластеризації.[2]

Порогові алгоритми використовують, коли необхідно розпізнавати зображення, області якого значнорозрізняються по яскравості. Для сегментації алгоритмом порогового розподілу необхідно отримати бінарне зображення. Для цього встановлюється деяке порогове обмеження. Після квантування функція зображення відображає елементи зображення з рівнем яскравості більше порогового в значення 1, менше порогового 0. У таких випадках оптимальне значення порога визначається досить легко на основі аналізу гістограм.[3]

Окремою гілкою виділяються алгоритми сегментації, засновані накластеризації. Їх перевага - вони автоматичні і можуть бути використані для будь-якої кількості ознак і класів. Існуючі алгоритми кластеризації, створені для знаходження кластерів, відповідних будь-якої статичної моделі. Ці алгоритми можуть дати збій, якщо параметри моделі були обрані некоректно, по відношенню до класифікованих даних, або якщо модель не охоплює в належній мірі характеристики кластерів. Також багато алгоритми допускають похибки, якщо дані складаються з кластерів різної форми, щільності і розмірів. Алгоритми вимірюють схожість двох кластерів за допомогою динамічної моделі. В процесі кластеризації злиття двох кластерів відбувається тільки, якщо внутрішня зв'язність і схожість двох кластерів тісно пов'язана із зовнішньою зв'язністю кластерів між собою і близькістю елементів всередині кластера.[4]

Інтелектуальний пошук для агентства з нерухомості

У сучасних реаліях пошук житла повинен враховувати низку факторів: ціну на помешкання, місце розташування, інфраструктуру тощо.Тому результати пошуку за допомогою звичайних фільтрів по ціні квартири/будинку чи за районом міста можуть не дати бажаних результатів. Рішенням такої проблеми може стати інтелектуальний пошук оснований на асоціативному пошуку. Тобто результатами пошукових запитів будуть не лише помешкання, знайдені за фільтрами, які вибрав користувач, а й помешкання асоціативно близькі до бажаних за такими критеріями:

1) Однакова кількість кімнат в будинках, які знаходяться поруч з шуканим.
2) Строк оренди помешкання незабаром закінчується.
3) Прогнозована вартість житла через певний період.

Для заданого пошукового запиту можна використовувати пороговий алгоритм сегментації, який би знаходив помешкання виділяючи (селектуючи) його з поміж інших за показником граничної ціни (графік) та який би міг прогнозувати ціну помешкання через певний період спираючись на попередні цінові зміни, що видно з рис.1, де вісь ординат показує ціну помешкання у тисячах доларів. З даного рисунку можна зробити висновок, що помешкання буде коштувати приємливо для користувача лише у квітні, травні та червні.

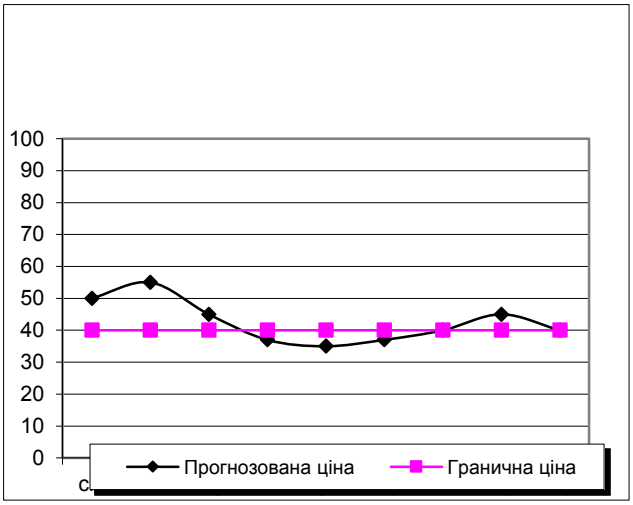


Рис. 1. Графікпрогнозованої ціни помешкання

Висновки

В даній роботі показані алгоритми інтелектуального пошуку даних та їх актуальність у наш час. Представлені недоліки пошуку за ключовими словами та методи їх рішення за допомогою інтелектуальних алгоритмів. Показані основні властивості ключових алгоритмів та їх практична дія на прикладі агентства з нерухомості.

Перелік посилань

1.ХараламбосМарманис, Дмитрий БабенкоАлгоритмі интеллектуальногоИнтернета. Передовые методики сбора, анализа и обработкиданных. 2011. №1. С. 161-264.
2.АлександрГалушкин. Нейронные сети. Основныетеории. 2012. С. 27-40.
3.Э.ХантИскусственныйинтеллект 2010, 292 с.
4.CleverAlgorithms: Nature-InspiredProgrammingRecipes. 2012. 172 p.

УДК 378.14

СЮР Г.В.,
КОВАЛЮК Т.В.

ПРО ОДИН МЕТОД ТА ЗАСОБИ ПЕРЕВІРКИ ЯКОСТІ НАУКОВО-ТЕХНІЧНОЇ ДОКУМЕНТАЦІЇ

У статті розглянуто питання перевірки науково-технічної документації на унікальність з метою підвищення її якості. Описано наявні методи, існуючі програмні рішення проблеми та найбільш доречний алгоритм.

Ключові слова: науково-технічна документація, якість, плагіат, унікальність тексту, шингли, алгоритм шинглів.

In this article is considered verification of scientific and technical documentation for uniqueness in order to improve its quality. Described methods existing, existing software solution and the most appropriate algorithm.

Вступ

У статті розглянуто питання перевірки науково-технічної документації на унікальність з метою підвищення її якості. Описано наявні методи, існуючі програмні рішення проблеми та найбільш доречний алгоритм.

Проблема виявлення плагіату в науково-технічній документації наразі залишається актуальною, адже хоч на сьогоднішній день і існує достатньо велика кількість сервісів і програм, які дозволяють виявити плагіат, вони мають забагато недоліків.

Проблема якості науково-технічної документації

Критерії оцінки [1] змісту – це:

- значимість події або подій, відображених в документах;
- значення інформації, що міститься в документах, її повторення в інших документах;
- цільове призначення, вид і різновид документа.

Зупинимось на такому критерії якості, як повторення викладеної інформації в інших документах. Адже плагіат з появою Інтернету перетворився в серйозну проблему. Потрапивши в Інтернет, знання стає надбанням усіх, дотримуватися авторського права стає все важче і навіть неможливо. Поступово стає складніше ідентифікувати початкового автора.

Стрімкий розвиток мережі Інтернет поряд зі зростаючою комп'ютерною грамотністю сприяє проникненню плагіату в різні сфери людської діяльності: плагіат є гострою проблемою в освіті, промисловості та науковому співтоваристві.

Плагіат є злочином. Це вводить в оману читачів, приносить шкоду автору, і надає незаслужені блага плагіаторові.

Широкий доступ до вітчизняної та зарубіжної літератури, багаторазове збільшення числа професійних видань, публікацій в Інтернеті – все це практично зводить нанівець будь-які редакторські прагнення «перевірити» або «встановити» справжність і оригінальність аргументів і фактів, які використовуються в рукописах, пропонованих до публікації.

Методи виявлення плагіату

Методи характеризуються за типом оцінки подібності.

Глобальна оцінка використовує великі частини тексту або документа для знаходження подібності в цілому, в той час як локальні методи на вході перевіряють обмежений сегмент тексту.

В даний час найбільш поширеним підходом є Дактилоскопія: з ряду документів вибирається набір з декількох підрядків, які і є «відбитками». Розглянутий документ буде порівнюватися з «відбитками» для всіх документів колекції. Знайдені відповідності з іншими документами вказують на загальні сегменти тексту [2].

Перевірка документа дослівним перекриттям тексту представляє собою класичне порівняння рядків.

Як правило, використовують моделі, такі як суфіксне дерево або суфіксний масив, які були адаптовані для виконання цього завдання в контексті комп'ютерного виявлення плагіату. Однак зіставлення підрядка є нежиттєздатним рішенням для перевірки великих колекцій документів (алгоритм відпрацьовує в середньому 2h порівнянь, де h – довжина рядка, в якій ведеться пошук) [3].

Аналіз "безлічі слів" є спрощенням уявлення, що використовується в обробці природної мови і пошуку інформації. У цій моделі текст представлений як невпорядкований набір слів. Документи представлені у вигляді одного або декількох векторів, які використовуються для попарного обчислення подібності [4].

Цитування – комп'ютерний метод виявлення плагіату, призначений для використання в наукових документах, що дозволяє використовувати цитати і довідковий матеріал. Визначає загальні цитати двох наукових робіт.

Шаблон цитат є підпослідовностями, що містять не тільки загальні цитати для двох документів, а й подібний порядок і близькість цитат в тексті, що є основними критеріями для визначення шаблону цитат [5].

Стильометрія або вивчення мовних стилів – це статистичний метод для виявлення авторства анонімних документів і для комп'ютерної перевірки на плагіат.

Будуються стилометричні моделі для різних сегментів тексту, уривків, які стилістично відрізняються від інших. І шляхом порівняння моделей можна виявити плагіат [6].

Огляд існуючих рішень

Advego Plagiatus

Програма здійснює онлайн перевірку з використанням пошукових систем. Програма видає відсоток збігу тексту і виводить знайдені джерела. Програма не перетворює літери, тобто немає перетворення регістру, немає обробки і зміни латинських букв в російських/українських словах на аналогічні літери російського/українського алфавіту для текстів російською/українською мовою. Також відсутня підтримка пошуку по власній базі; через особливості роботи виникають ситуації, коли результати перевірки відрізняються від разу до разу.

Переваги Advego:

– Немає обмежень на кількість тексту, що перевіряється.

– Потужна підтримка.

– Багато функцій і налаштувань.

– Швидке оновлення.

Недоліки Advego:

– Трохи повільніше основного конкурента - програми Etxt Антиплагіат.

– Маленька довжина шингла за замовчуванням.

– Зайва підозрілість.
– Часта поява CAPTCHA.

«ETXT Антиплагіат»

Ця програма відрізняється вражаючим набором можливостей, в тому числі що не мають ніякого відношення до унікальності. Заснована «eTXT Антиплагіат» на двох алгоритмах: «Метод виявлення копій» (шингловий) і «Метод виявлення рерайта» (кореляційний).

Переваги Etxt:

– Немає обмежень на кількість тексту, що перевіряється.
– Багато функцій і налаштувань.
– Швидке оновлення.
– Перевірка унікальності зображення - функція, що дозволяє порівняти два графічних файли.
– Підтримка антикапчерів - в настройках можна ввести адресу будь-якого сервісу і ключ.
– Перевірка текстів в пакетному режимі - для цього достатньо вказати папку, де знаходяться файли.

Недоліки Etxt аналогічні Advego:

– Маленька довжина шингла за замовчуванням.
– Зайва підозрілість.
– Часта поява CAPTCHA.

Content Watch

Цей сервіс заснований на кореляційному алгоритмі.

Переваги Content-watch.ru:

– Відсутність паніки і помилкових спрацьовувань.
– Швидка і точна перевірка.
– Найдешевша реалізація API.

Недоліки Content-watch.ru:

– Без реєстрації можна перевірити до 5 текстів в день розміром не більше 3000 знаків. Зареєстрованим користувачам доступно 20 перевірок по 20 000 знаків.

TEXT.RU

Онлайновий сервіс, заснований на кореляційному алгоритмі.

Переваги Text.ru:

– Можливість додавання завдання в чергу.
– Наявність посилань на результати перевірки.
– Можливість підключення API, а також масової (але платною) онлайн-перевірки всіх сторінок сайту на унікальність.

Недоліки Text.ru:

– Часто бувають технічні збої.
– Часом дуже довга черга і в результаті вкрай низька швидкість перевірки.
– Дуже незручна, та ще й платна перевірка тексту за посиланням.
– Відсутність будь-яких налаштувань і режимів перевірки;
– Розмір тексту для незареєстрованих користувачів - 2000, після реєстрації - 15 000 знаків;
– Немає підсвічування частин тексту для кожного знайденого домену.

Але найголовнішим недоліком вищезазначених сервісів та програм є відсутність перевірки посилань, адже якщо два фрагменти різних текстів мають посилаються на одне й те ж джерело, значить це – не плагіат!

Постановка задачі

Автор ставить задачу розробки програмного засобу перевірки якості науково-технічної документації з урахуванням наявності CAPTCHA, варіювання розміру шингла, необмеженості тексту. Засіб, що розробляється, має перевіряти схожість посилань на використані джерела, визначати унікальність

зображення – функція, що дозволяє порівняти два графічних файли, перевіряти тексти в пакетному режимі – для цього достатньо вказати папку, де знаходяться файли, здійснювати пошук схожого тексту в Інтернеті. Доцільно реалізувати швидку і точну перевірку, виявляти посилання на результати перевірки, підсвічувати частини тексту для кожного знайденого домену.

Алгоритм шинглів

Одними з перших досліджень в області знаходження нечітких дублікатів є роботи U. Member [7] і N. Heintze [8]. У цих роботах для побудови вибірки використовуються послідовності сусідніх букв. Дактилограма файлу або документа включає всі текстові підрядки фіксованої довжини. Чисельне значення дактилограм обчислюється за допомогою алгоритму випадкових поліномів Карпа-Рабіна [9]. В якості запобіжного подібності двох документів використовується відношення числа загальних підрядків до розміру файлу або документа. U. Monter використовував цей підхід для знаходження схожих файлів (утиліта sif), а N. Heintze – для виявлення нечітких дублікатів документів (система Koala).

У 1997 році A. Broder [10] запропонували новий, «синтаксичний» метод оцінки подібності між документами, заснований на представленні документа у вигляді безлічі всіляких послідовностей фіксованої довжини к, що складаються з сусідніх слів. Такі послідовності були названі «шингли». Два документа вважалися схожими, якщо їх множини шинглів істотно перетиналися. Оскільки число шинглів приблизно дорівнює довжині документа в словах, тобто є досить великим, авторами були запропоновані два методи семплювання для отримання репрезентативних підмножин.

Один з них залишав тільки ті шингли, чиї «дактилограми», які обчислюють за алгоритмом Карпа-Рабіна [9], ділилися без залишку на деяке число т. Основний недолік – залежність вибірки від довжини документа і тому документи невеликого розміру (в словах) представлялися або дуже короткими вибірками, або взагалі нс мали таких. Другий метод відбирав тільки фіксоване число s шинглів з найменшими значеннями дактилограм або залишав всі шингли, якщо їх загальна кількість не перевищую s.

Подальшим розвитком концепцій A. Broder являються дослідження D. Fetterly [11].

Ідея кожного ланцюжка обчислюються 84 дактилограми за алгоритмом Карпа-Рабіна [9] за допомогою взаємнооднозначних і незалежних функцій, що використовують випадкові набори («min-wise independent») простих поліномів. В результаті кожен документ представлявся 84 шингли, які мінімізують значення відповідної функції.

Потім 84 шннгла розбиваються на 6 груп по 14 (незалежних) шинглів в кожній. Ці групи називаються «супершинглами». Якщо два документи мають схожість, наприклад, р ~ 0.95 (95%), то 2 відповідних супершингли в них збігаються з імовірністю $p^{14} \sim 0.95^{14} \sim 0.49$ (49%).

Оскільки кожен документ представляється 6 супершнгламн, то ймовірність того, що у двох документів співпадуть не менше 2-х супершинглів, дорівнює: $1 - (1-0.49)^6 * - 6* 0.49 *(1-0.49)^5 \sim 0.90$ (90%).

Для порівняння: якщо два документи мають схожість тільки р - 0.80 (80%), то ймовірність збігу не менше двох супершинглів становить всього 0.026 (2.6%).

Таким чином, для ефективної перевірки збігу не менше 2-х супершинглів (і, отже, підтвердження гіпотези про подібність змісту) кожен документ представляється всілякими попарними поєднаннями з 6 супершинглів, які називаються «мегашингли». Число таких мегашинглів дорівнює 15 (число сполучень з 6 по 2). Два документа подібні за змістом, якщо у них збігається хоча б один з мегашинглів.

Ключова перевага даного алгоритму в порівнянні з дослідженнями A. Broder полягає в тому, що, по-перше, будь-який документ (в тому числі і дуже маленький) завжди представляється вектором фіксованої довжини, і, по-друге, подібність визначається простим порівнянням координат вектора і нс вимагає (як у A. Broder) виконання теоретико-множинних операцій.

Висновки

В даній статті були виявлені критерії аналізу якості науково-технічної документації, більш детально розглянуто проблему плагіату.

Розглянуто методи вирішення даної проблеми. Проаналізовано існуючі програмні рішення виявлення плагіату, їх переваги та недоліки. На основі цього була зроблена постановка задачі та обрано алгоритм шинглів.

Перелік посилань

1. Критерии оценки документов [Электронный ресурс] Режим доступа: http://www.grandars.ru/college/pravovedenie/ocenka-dokumentov.html
2. Brin S., Davis J., Garcia-Molina H. Copy Detection Mechanisms for Digital Documents (англ.) // Vine.. — 2001.
3. Monostori K., Zaslavsky A., Schmidt H. Document Overlap Detection System for Distributed Digital Libraries (англ.) // ACM. — 2000.
4. Leong A., Lau H., Rynson W. H. Check: A Document Plagiarisment Detection System (англ.) // ACM. — 1997.
5. Gipp B., Meuschke N., Beel J. Comparative Evaluation of Text- and Citation-based Plagiarism Detection Approaches using GuttenPlag. (англ.) // ACM. — 2011.
6. Meyer zu Eissen S., Stein B. Intrinsic Plagiarism Detection. (англ.) // Springer. — 2006.
7. Manber U. Finding Similar Files in a Large File System. Winter USENIX Technical Conference, 1994.
8. Heintze N. Scalable document fingerprinting. In Proc. of the 2nd USENIX Workshop on Electronic Commerce, Nov. 1996.
9. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. СПб.: Невский диалект, 2003.
10. Broder A., Glassman S., Manasse M. And Zweig G. Syntactic clustering of the Web. Proc. of the 6th International World Wide Web Conference, April 1997.
11. Fetterly D., Manasse M., Najork M. A Large-Scale Study of the Evolution of Web Pages, WWW2003, May 20-24, 2003, Budapest, Hungary.

УДК 004.27

ТАРАН В. І.,
КЛИМЕНКО І. А.

МЕТОД АДАПТИВНОЇ РЕКОНФІГУРАЦІЇ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ НА БАЗІ ПЛІС

У даній статі запропоновано метод, що полягає у скороченні часу реконфігурації обчислювального алгоритму за рахунок попередньої реконфігурації, яка відбувається на попередньому рівні графа ярусно-паралельної форми алгоритму.

The subject of the article is method that allows to decrease the reconfiguration time by performing reconfiguration before task execution. In this way, reconfiguration is performed on the previous stage of the multilevel structure algorithm graph.

Вступ

На сьогодні актуальними є реконфігуровані обчислювальні системи. Елементною базою для їх побудови є програмовані логічні інтегральні схеми – ПЛІС, які можна конфігурувати для вирішення різних задач. Але є проблема – значні часові витрати на перебудову

обчислювальної структури, що значним чином впливає на ефективність функціонування таких систем. Тому виникає потреба в розробці методів та засобів, що дозволять прискорити процес реконфігурації обчислювальної структури.

Запропонований метод полягає у скороченні часу реконфігурації обчислювального алгоритму за рахунок попередньої реконфігурації, яка полягає у переносі складової часу конфігурації на попередній рівень графа ярусно-паралельної форми алгоритму – ЯПФ. Це дозволить скоротити критичний шлях графа ЯПФ алгоритму.

Спрощена структура ПЛІС

Здебільшого програмовані логічні інтегральні схеми складаються із наступних елементів:

- Конфігуруванні логічні блоки, що реалізують певну логічну функцію
- Програмованих електронних зв'язків міжконфігурованими логічними блоками
- Програмованих блоків вводу/виводу, що забезпечують зовнішній зв'язок із внутрішньою логікою мікросхеми

На рис. 1. зображена спрощена структура мікросхеми ПЛІС. За способом зберігання конфігурації ПЛІС поділяються на:

- SRAMbased

Найбільш розповсюдженій тип ПЛІС. Конфігурація зберігається в статичній пам'яті, типу CMOS. До переваг можна віднести – можливість багаторазової реконфігурації[4]. Через особливість використаної пам'яті, мікросхему необхідно повторно конфігурувати при кожному вмиканні. Це призводить до збільшення апаратної складності, оскільки кінцевий пристрій повинен мати додаткові компоненти, такі як мікросхема FLASH та завантажувач конфігурації.

- Flashbased

В даному типі, конфігурація мікросхеми зберігається у внутрішній FLASH чи EPROM пам'яті. Цей тип мікросхем не має недоліків ПЛІС типу SRAM, але за рахунок використання в них FLASH та EPROM пам'яті є дорожчими у виробництві. Як правило, мікросхема має обмежену кількість циклів перезапису.

- Anti-fusebased

Технологія за якою конфігурація програмованої логічної інтегральної схеми виконується одноразово. Цей процес побудовано на плавленні спеціальних перемичок у потрібному місці на чипі. До переваг даного типу відносять високу швидкодію та надійність.

Часткова реконфігурація

Часткова реконфігурація ПЛІС – це можливість динамічної модифікації логічних блоків мікросхеми, частково завантажуючи в них бітова потоки (бінарні файли), при цьому решта логіки продовжує працювати без перебоїв [5]. Технологія часткової реконфігурації дозволяю розробникам змінювати функціональність "на льоту", виключаючи необхідність повної реконфігурації та перерозподілу зв'язків, що в рази підвищує гнучкість можливостей використання програмованих логічних інтегральних схем. Використання часткової реконфігурації дає змогу розробникам використовувати менше число схем, зменшити споживання електроенергії та підвищити можливості модернізації. Це дозволяє більш ефективно використовувати ресурси мікросхеми, завантажуючи тільки ту функціональність, яка необхідна у певний момент часу [6].

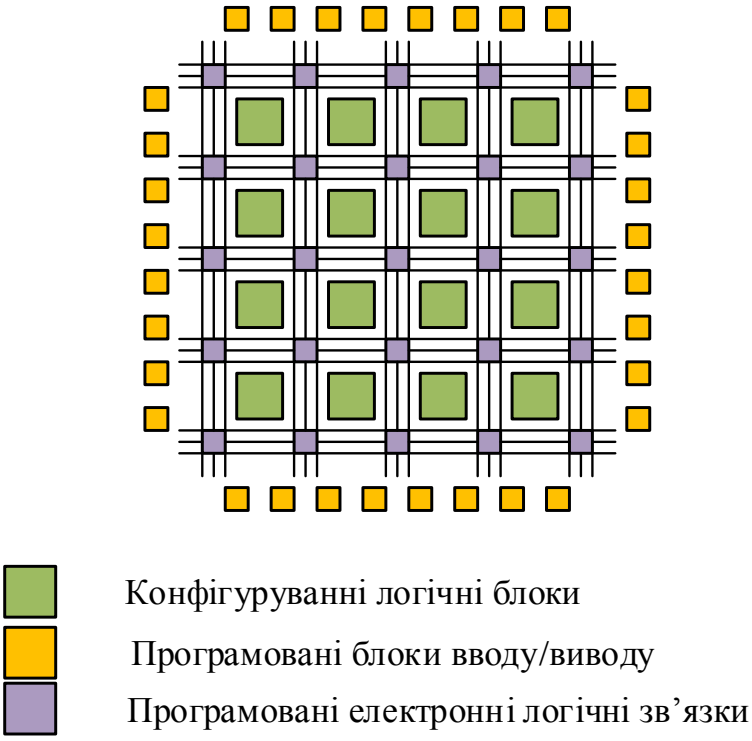


Рис. 1. Спрощена структура програмованої логічної інтегральної схеми

На рис. 2. показано базові принципи часткової реконфігурації програмованої логічної інтегральної схеми. В мікросхему можна завантажити необхідні часткові бітові потоки, для виконання певної функції.

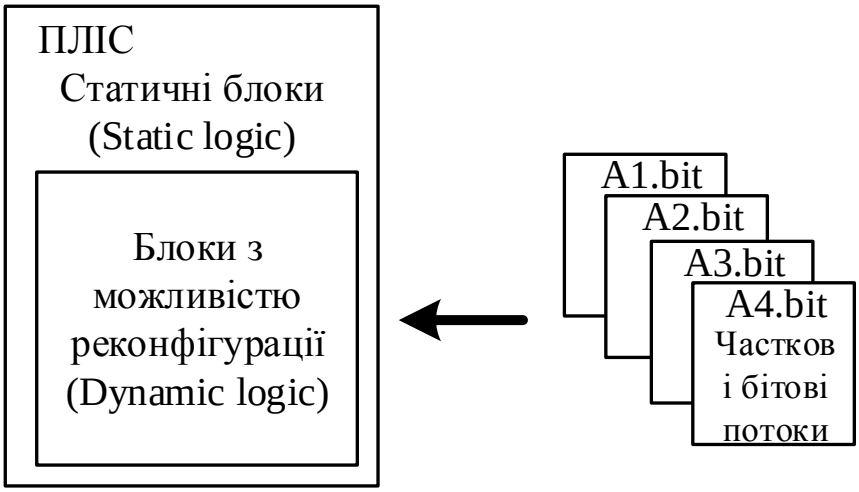


Рис. 2. Часткова реконфігурація ПЛІС

Завантаження конфігурації ПЛІС відбувається через спеціальний порт JTAG за допомогою засобів персонального комп'ютера [3]. Часткові бітові потоки зберігаються у спеціально відведеній області зовнішньої флеш-пам'яті. Часткова реконфігураціяініціюється логікою управління, реалізованою в ПЛІС. Фізичне управління процесом реконфігурації здійснює керувальний контролер у складі реконфігурованої системи.

Також для реконфігурації програмованої логічної інтегральної схеми використовується модуль ICAP (InternalConfiguration Access Port). ICAP — це апаратний засіб ПЛІС, який дає можливість динамічно завантажувати в пам'ять потік бітів без застосування порту JTAG [1].

При цьому для реконфігурації не потрібно використовувати зовнішній пристрій і вся необхідна конфігураційна інформація зберігається на тій же платі, що зменшує час передачі даних.

Програмна модель реалізації запропонованого методу

В рамках дослідження розроблена програмна модель реалізації запропонованого методу, яка дозволяє виконувати адаптивну реконфігурацію обчислювальної системи на базі ПЛІС, за рахунок проведення структурного аналізу графа алгоритму та формування черги реконфігурації логічних блоків ПЛІС. Алгоритм роботи програми може бути представлений наступними кроками: 1. Підготовка даних, на якому створюється граф задачі; 2. Структурний аналіз графа – створення матриці переходів та побудова ярусно-паралельної форми алгоритму; 3. Формування черги часткових бітових потоків для реконфігурації; 4. Реконфігурація блоків ПЛІС.

За результатами моделювання отримані часові діаграми реалізації запропонованого методу (рис. 3.), на яких видно, що використання адаптивної реконфігурації блоків ПЛІС дозволяє суттєво зменшити сумарний час виконання задачі.

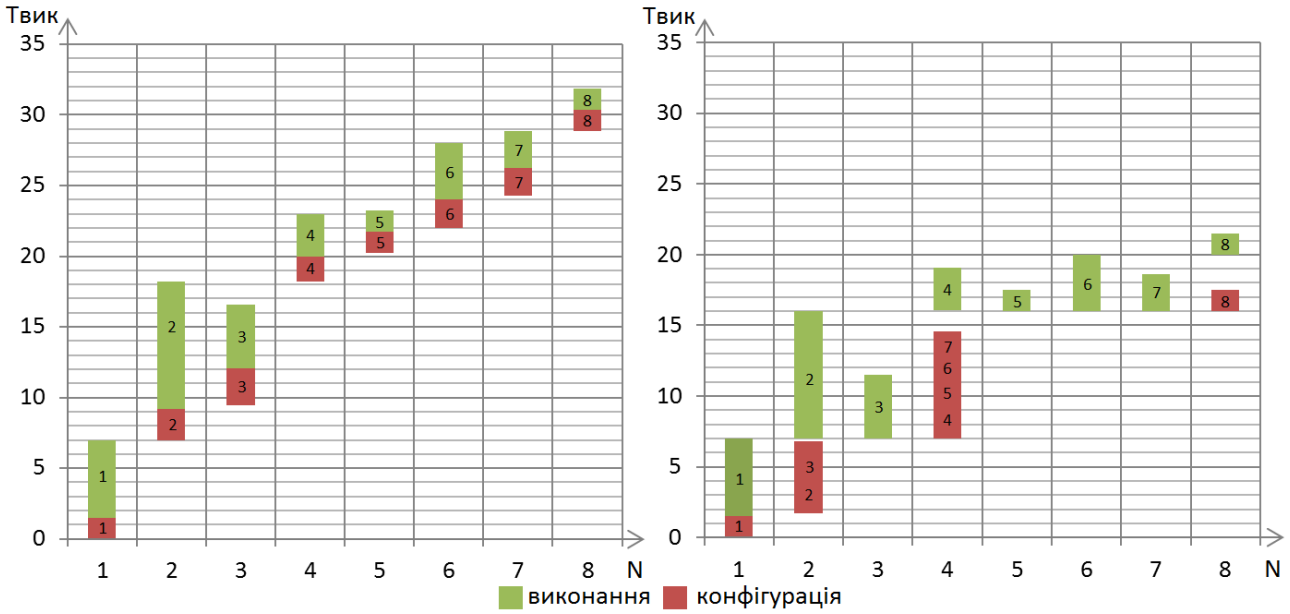


Рис. 3. Порівняльна діаграма часу виконання задачі

Перелік посилань

1) 29. Ming L. Run-time Partial Reconfiguration speed investigation and architectural design space exploration / L. Ming, W. Kuehn, L. Zhonghai, A. Jantsch. // Field Programmable Logic and Applications, 2009. (FPL 2009). (Prague, August 31—September 2, 2009). — IEEE Computer Society, 2009. — P. 498—502.

2) Birla M. Partial run-time reconfiguration of FPGA for computer vision applications / M. Birla, K. N. Vikram. // Parallel and Distributed Processing, 2008 (IPDPS 2008). (Miami, April 14—18, 2008). — IEEE Computer Society, 2008. — P. 1—6.

3) FPGAs [Електронний ресурс] – Режим доступу до ресурсу: http://www.slideshare.net/abhilash128/lec-23.

4) Partial Reconfiguration in the Vivado Design Suite [Електроннийресурс] – Режимдоступудоресурсу: http://www.xilinx.com/products/design-tools/vivado/implementation/partial-reconfiguration.html.

5) Field Programmable Gate Array (FPGA) [Електроннийресурс] – Режимдоступудоресурсу: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm.

УДК 004.056.5

ТКАЧЕНКО І.М.,
МАРКОВСЬКИЙ О.П.

ЕФЕКТИВНЕ ОБЧИСЛЕННЯ КВАДРАТНОГО КОРЕНЯ НА ПОЛЯХ Галуа GF(2^m)

У статті запропоновано спосіб прискореного обчислення кореня на полях Галуа GF (2^m). Теоретично показано, що завдання обчислення коренів на полях Галуа може бути зведена до розв'язання системи лінійних бітових рівнянь. Запропоновано технологію реалізації цієї теоретичної ідеї. Доведено, що обчислювальна складність O*(m)* запропонованого способу істотно менше ніж складність відомих способів, яка становить O*(m^2).*

In article, the method of accelerated calculation of square root on Galois fields GF (2^m) has been proposed. By the theoretical way, it has been shown that computing roots on Galois fields' calculation can be reduced to solving system of linear bits equations. New technology of this theoretical idea was proposed. It has been proved, that calculation complexity O*(m)* of proposed method is much smaller in comparing to known methods, which equals O*(m^2).*

Вступ

Рішення алгебраїчних рівнянь на кінцевих полях і, зокрема, на полях Галуа GF GF(2m), являє собою одну з найбільш важливих завдань обчислювальної алгебри і теорії чисел [1].

Найважливішою складовою частиною технологій вирішення алгебраїчних рівнянь на кінцевих полях є обчислення квадратного кореня. Типовим застосуванням операції добування кореня є стиснення і відновлення точки на еліптичній кривій [3]. Точка з координатами (x, y) на кривій стискається до виду (x, □□□де $\beta \in \{0,1\}$. Для відновлення чисельного значення y по (x,β), необхідно вирішити квадратне рівняння y2 = P(x), тобто обчислити квадратний корінь $\sqrt{P(x)}$. Подібна ситуація виникає і при хешуванні на еліптичних кривих, яке використовується в ряді криптосистем [4,5].

В останні роки з розвитком технологій розподілених обчислень істотно зросли можливості комп'ютерних систем, які потенційно можуть бути використані для порушення захисту. Найбільш простим заходом підвищення криптостійкості систем, заснованих на використанні полів Галуа GF(2m) є збільшення розрядності м використовуваних чисел. Це різко уповільнює продуктивність засобів криптографічного захисту. Тому, в сучасних умовах важливою і актуальною є проблема розробки нових підходів до прискорення програмної та апаратної реалізації операції обчислення квадратного кореня на кінцевих полях. Основним резервом зменшення обчислювальної складності цієї важливої для практичних застосувань операції є врахування особливостей її використання в реальних системах.

Аналіз відомих методів обчислення коренів на кінцевих полях

Практична значущість задачі обчислення квадратного кореня на кінцевих полях, особливо для систем криптографічного захисту інформації на основі еліптичних кривих, стимулює інтенсивні дослідження в області технології вирішення цього завдання. Як вже зазначалося, більше ста років тому були запропоновані два базових методи обчислення квадратного кореня на полях Галуа GF(2^m): Tonelli [1] и Cipolla [7]. Tonelli [1] і Cipolla [7].Потім ці методи були розширені для випадку поля GF(q^m), де *q*-просте і отримали назву відповідно: Tonelli-Shanks [6] і Cipolla-Lehmer [8].У 1977 році в роботі [9] метод onelli-Shanks був розширений на випадок вилучення кореня довільного ступеня. В роботі [10] був розроблений спеціалізований метод вилучення кубічного кореня, що відрізняється підвищеною швидкодією.

Для кожного з елементів поля GF(2m), що породжується нерозкладним поліномом P (x) ступеня m, якому відповідає число p, існує мультиплікативна циклічна група, порядок n якої не перевищує 2 m -1. Наприклад, для поля Галуа, утвореного нерозкладним поліномом P (x) =

x 4 + x 3 + 1 (p = 25 10 = 11 001 2) генеруються циклічні групи. Порядок циклічної групи дорівнює 2 м -1, якщо її генератор не має спільних дільників з 2m-1.

Основна ідея знаходження квадратного кореня $\sqrt{A}$ на полях Галуа GF(2m) методом Tonelli-Shanks по суті полягає в проходженні квадратичної циклічної підгрупи до знаходження її елемента, що передує шуканого. У процедурному плані прохід по квадратичної циклічної підгрупі еквівалентний операції експоненціювання [10].

$$B = A^{2m-1} rem(p). \qquad (1)$$

Таким чином, ідея вилучення квадратного кореня на полях GF(2m) теоретично досить проста, однак її практична реалізація пов'язана зі значними витратами обчислювальних ресурсів, оскільки обчислення (1) передбачає виконання m-1 операцій піднесення до квадрату і редукції.

Операції зведення в квадрат виконуються за правилами поліноміального множення, тобто без урахування переносів. Відомо, що операція поліноміального піднесення до квадрату має важливу властивість: двійкові розряди поліноміального квадрата числа А стоять на парних позиціях рівні нулю, а розряди з непарним номером рівні двійковим розрядам числа А, тобто, якщо A=a0+a1·2+a2·22 +...+am-1·2m-1, то:

$$A \otimes A = A^2 = a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + ... + a_{m-1} \cdot 2^{2m-2}. \qquad (2)$$

Найважливішим, з точки зору, організації обчислень, наслідком цієї властивості є той факт, що обчислення поліноміального квадрата не вимагає для своєї реалізації ніяких обчислювальних операцій, а зводиться тільки до перестановки розрядів вихідного числа.

При оцінці обчислювальної складності операції добування квадратного кореня за методом Tonelli-Shanks слід врахувати те, що на практиці розрядність m елементів поля істотно перевищує розрядність w процесора, так, що при виконанні операцій елементи поля розбиваються на s секцій (s = m/w).

Операція редукції, тобто приведення результату поліноміального множення в рамки поля виконується шляхом обчислення залишку від поліноміального поділу $(2 \cdot m-1)$- розрядного результату поліноміального зведення в квадрат на (m+1)- розрядний код утворює полінома р поля. Реалізація операції поліноміального ділення включає виконання (m-1) циклів, в кожному з яких здійснюється зсув на один розряд (m+1)- розрядного коду р і логічне додавання його з кодом поточного залишку в разі, якщо старший розряд останнього дорівнює одиниці. Для зсуву (m+1)- розрядного коду р на один розряд необхідно виконати (s+1) процесорних операцій зсуву. Так як ця операція виконується в кожному з (m-1)циклів редукції, то сумарна кількість процесорних операцій зсуву становить $(s+1)\cdot(m-1)$. Виходячи з того, що в процесі редукції операція додавання виконується, в середньому, в половині циклів, то середня кількість таких операцій становить (m-1)/2. Беручи до уваги, що для реалізації цієї операції на w- розрядному процесорі треба виконати (s+1) процесорних операцій логічного складання для редукції результату множення складає: $(s+1)\cdot(m-1)/2$. Відповідно, середній час виконання однієї редукції результату піднесення до квадрату становить $1.5\cdot(s+1)\cdot(m-1)\cdot\tau$, де τ-час виконання на на процесорі логічної операції. Враховуючи, що вилучення квадратного кореня вимагає (m-1) операції зведення в квадрат, середнє число NT логічних операцій, потрібних для добування квадратного кореня на поле GF(2m) становить:

$$N_T = 1.5 \cdot (s+1) \cdot (m-1)^2. \qquad (3).$$

Аналогічна оцінка обчислювальної складності O(m2) наведена в [1] і для методу Cipolla-Lehmer. У сучасних умовах збільшення продуктивності розподілених - комп'ютерних систем, які потенційно можуть бути використані для порушення захисту, найбільш простим способом підвищення крипостійкості є збільшення значення розрядності використовуваних чисел. При цьому, як випливає з (3) різко зростає обчислювальна складність реалізації операції отримання квадратного кореня на полях GF(2m).

Метою досліджень є розробка способів прискорення вилучення квадратного кореня на полях Галуа, орієнтованих на апаратну реалізацію та широке розпаралелювання.

Обчислення кореня рішенням системи булевих рівнянь

Обчислення квадратного кореня на полях Галуа GF(2m) може бути зведене до розв'язання системи лінійних бітових рівнянь. Нехай задано значенняA=a0+a1·2+a2·22+...+am-1·2m-1, a0,a1,…,am-1∈{0,1}. Необхідно визначити B=b0+b1·2+b2·22 +...+bm-1·2m-1, b0,b1,…,bm-1∈{0,1} таке, що (B⊗B)rem P = A чи B⊗B = P⊗D + A, де D=d0+d1·2+d2·22 +...+dm-2·2m-2. Оскільки двійкові розряди поліноміального квадрата B⊗B числа B стоять на парних позиціях рівні нулю, а розряди з непарним номером рівні двійковим розрядам числа B, тобто B⊗B = b0+b1·22+b2·24 +...+bm-1·22·m-2, b0 = p0·d0 + a0 оскільки наступний розряд B⊗B рівний нулю, то p0·d1 + p1·d0 + a1 = 0. Міркуючи аналогічним чином, може бути отримана система бітових рівнянь:

$$\begin{cases} b_0 = p_0 \cdot d_0 + a_0 \\ 0 = p_0 \cdot d_1 + p_1 \cdot d_0 + a_1 \\ b_1 = p_0 \cdot d_2 + p_1 \cdot d_1 + p_2 \cdot d_0 + a_2 \\ 0 = p_0 \cdot d_3 + p_1 \cdot d_2 + p_2 \cdot d_1 + p_3 \cdot d_0 + a_3 \\ \dots \\ b_{m/2-1} = p_0 \cdot d_{m-2} + p_1 \cdot d_{m-3} + \dots + p_{m-2} \cdot d_0 + a_{m-2} \\ 0 = p_1 \cdot d_{m-2} + p_2 \cdot d_{m-3} + \dots + p_{m-1} \cdot d_0 + a_{m-1} \\ b_{m/2} = p_2 \cdot d_{m-2} + p_3 \cdot d_{m-3} + \dots + p_m \cdot d_0 \\ \dots \\ 0 = p_{m-1} \cdot d_{m-2} + p_m \cdot d_{m-3} \\ b_{m-1} = p_m \cdot d_{m-2} \end{cases} . \quad (4)$$

ри постійному створюючому поліномі *P(x)* поля Галуа GF(2^m) система (4) може бути приведена до вигляду:

$$\begin{cases} b_0 = \lambda_0(a_0, a_1, ..., a_{m-1}) \\ b_1 = \lambda_1(a_0, a_1, ..., a_{m-1}) \\ \dots \\ b_{m-1} = \lambda_{m-1}(a_0, a_1, ..., a_{m-1}) \end{cases} \quad (5)$$

де $\lambda_0, \lambda_1, \dots, \lambda_{m-1}$ – лінійні булеві функції. Із системи (5) безпосередньо обчислюються значення бітів коду квадратного кореня на поле Галуа.

Наприклад, якщо $m = 4$, $P(x) = x^4 + x^3 + 1$, $p_0 = 1$, $p_1 = 0$, $p_2 = 0$, $p_3 = 1$, $p_4 = 1$, система (5) приймає вигляд:

$$\begin{cases} b_0 = a_0 + a_3 \\ b_1 = a_0 + a_1 \\ b_2 = a_1 + a_3 \\ b_3 = a_1 \end{cases} \quad (6)$$

Наприклад, якщо $A = 15$ ($a_0 = 1$, $a_1 = 1$, $a_2 = 1$, $a_3 = 1$), з системи (6) відповідно: $b_0 = 0$, $b_1 = 0$, $b_2 = 0$ і $b_3 = 1$, тобто шуканий корінь $B = 8$.

При програмної реалізації рішення системи (5) можуть бути заздалегідь заготовлені по и бітових масок Mj1, Mj2, …, Mjs для виділення розрядів коду A, які входять в функцію λj для

кожного із значень b_j, $j \in \{0, .., m\text{-}1\}$. Відповідно, обчислення значення b_j здійснюється у вигляді логічної суми бітів парності побітових добутків фрагментів коду А на відповідні маски:

$$b_j = (a_0, a_1, ..., a_{w-1}) \cdot M_{j1} + (a_w, a_{w+1}, ..., a_{2 \cdot w-2}) \cdot M_{j2} + ..$$

$$. + (a_{m-w-1}, a_{m-w}, ..., a_{m-1}) \cdot M_{js}$$

Очевидно, що загальна кількість N_L логічних операцій необхідних для обчислення значень m бітів b_0, b_1, …, b_{m-1} визначається як

$$N_L = s \cdot m \qquad (7)$$

Порівняння отриманого виразу з оцінкою (3) числа операцій, необхідних для обчислення квадратного кореня на GF(2^m) по известному способу за відомим способом, показує, що запропонований спосіб виконання цієї операції забезпечує зменшення обчислення обчислювальної складності приблизно в $1.5 \cdot m$ разів:

$$\beta = \frac{N_T}{N_L} = \frac{1.5 \cdot (s+1) \cdot (m-1)}{s \cdot m} \approx 1.5 \cdot m$$

Враховуючи, що в алгоритмах захисту інформації значення m складає сотні і тисячі біт, виграш в обчислювальній складності і, відповідно, часу обчислення квадратного кореня на GF(2m) досягається використанням запропонованого способу складає 2-3 порядки. Ще більший виграш як у часі обчислення квадратного кореня на полях Галуа, так і по складності схеми, застосування запропонованого способу забезпечує при апаратній реалізації, оскільки вирази (5) представляють собою гранично просту форму обчислення бітів кореня, кожен з яких може обчислюватися паралельно.

Висновки

Запропоновано спосіб прискореного обчислення квадратного кореня на полях Галуа GF(2m), заснований на зведенні цього завдання до розв'язання системи лінійних бітових рівнянь. Доведено, що запропонований спосіб має обчислювальну складність O(m), істотно меншу, ніж відомі методи - O(m2). Запропонований спосіб орієнтований на апаратну реалізацію та паралельні обчислення бітів кореня. Проведені дослідження показали, що використання запропонованого способу забезпечує виграш у часі на 2-3 порядки.

Перелік посилань

1. Menezes A. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Published.-1993.
2. Tonelli A. Bemerkung über die Auflösung quadratischer Congruenzen // Göttinger Nachrichten.- 1891.- PP.344-346.
3. Boneh D., Franclin M. Identity-based encryption from the Well pairing // SIAM Journal of Computing.- Vol.23.- 2003.- № 3.- PP. 354-368.
4. Galbraith S., Paulus S., Smart T. Arithmetic on superelliptic curves // Mathematics of Computation. - Vol.71.- 2002.- PP. 393-405.
5. Barreto P.S.L.M., Voloch J.F. Efficient Computation of Root in Finite Fields // Designs, Codes and cryptography.- 2006.- № 39.- pp. 275-280.
6. Shanks D. Five number-theoretic algorithms // Proc/ 2-nd Manitoba Conf. Number. Math.- Manitoba.-Canada.-1972.-pp. 51-70.
7. Lehmer D.H. Computer technology applied to the theory of numbers // Studies in Number Theory.- Englewood Cliffs.-NJ^Pretice-Hall.- 1969.- pp.117-151.
8. Cipolla M. Un metodo per la risolutione della congruenza di secondo grado // Rendiconto dell'Accademia Scienze Fisiche e Matematiche.- Napoli.-1903.- Ser.3- Vol.IX.- PP.154-163.

9. Aldeman L.M., Manders K. Miller G. On taking root in finite fields // Proc. 18-th IEEE Symposium on Foundations of Computer Science.-1977.-PP.175-177.
10. Nishihara N., Harasawa R., Sueyoshi Y. A remark on the computation of cube root in finite fields // IACR Cryptology ePrint Archive.-2009.- 457

УДК 004.93

УВАРОВ Н. В.,
МАРКОВСЬКИЙ О.П,

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ КОРЕКЦІЇ ПОМИЛОК В ПОСЛІДОВНИХ ІНТЕРФЕЙСАХ КОМП'ЮТЕРНИХ СИСТЕМ

Метою даної роботи є підвищення ефектив-ності корекції багатократних помилок, що виникають при обміні даними між компонентами комп'ютерних систем, за рахунок розробки таких методів їх виправлення, які дозволяють спростити та прискорити обчислення, пов'язані з корекцією, а також розширити клас помилок, що можуть бути виправлені без повторної передачі.

The subject of this article is to improve the correction of multiple errors arising from the exchange of data between components of computer systems through the development of methods of correction which simplify and speed up the calculations associated with the correction and expand the class of errors that can be corrected without retransmission.

Вступ

Вдосконалення технології передачі даних є одним з найважливіших чинників поглиблення глобального процесу інформаційної інтеграції. Разом з тим, традиційно, передача даних є одним з найбільш критичних, з точки зору надій-ності, процесів обробки інформації в комп'ю-терних системах. Інтенсив-ність виник-нення помилок при передачі даних на декілька порядків вища в порівнянні з обчисленнями на процесорних вузлах комп'ютерних систем. Тому однією з вузлових проблем розвитку комп'ю-терних технологій є забезпечення високої надійності обміну даними між компонентами комп'ю-терних систем.

Метод виправлення помилок на основі степеневих зважених контрольних сум

Важливою задачею підвищення надійності процесів обміну даними між компонентами комп'ютерних систем є дослідження шляхів підвищення ефективності виправлення багатократних помилок передачі даних в лініях з імпульсно-кодовою модуляцією, які відповідають теоретичній моделі двійкового симетричного каналу. Саме такі лінії найчастіше використовуються для передачі даних між компонентами обчислювальних систем. Одним із можливих шляхів вирішення вказаної наукової задачі полягає в використанні корегуючих кодів на основі нелінійних зважених контрольних сум. Технології зважених контрольних сум для виявлення та виправлення помилок передачі даних широко використовуються [1, 2, 3].

Головною перевагою вико-ристання зважених контрольних сум вважається можливість їх адаптації до характеру помилок в лінії обміну даними [4]. Не менші можливості відкриває використання зважених контрольних сум і для побудови нелінійних кодів, я якості вагових коефіцієнтів яких виступають результати нелінійних перетворень над ваговими коефіцієнтами.

Важливим моментом використання нелінійних зважених контрольних сум є можливість побудови таких кодів на двох алгоритмічних основах: в звичайній алгебрі цілих чисел

(арифметичні зважені контрольні суми) та поліноміальній алгебрі кінцевих полів Галуа. Використання першої з двох вказаних алгебраїчних основ більш ефективно при використанні програмних засобів виправлення виникаючих помилок передачі даних, а застосування алгебраїчного базису кінцевих полів Галуа більш ефективне при реалізації виправлення помилок апаратними засобами.

Корекція багатократних помилок на основі арифметичних ступеневих зважених контрольних сум

В математичному сенсі головна складність виправлення двократних помилок полягає в формуванні та розв'язанні системи з двох рівнянь, коренями якої є позиції спотворених бітів. Контрольний код має формуватися таким чином, щоб на приймачеві можна було утво-рити систему рівнянь, що мають єдиний розв'язок.

Модель: передається блок, що складається з m бітів: B={b1,b2,…,bm}, причому ∀j={1,..,m}:bj∈{0,1}. Під час передачі зазнають спотворення два біти. Будемо вважати, що біти в блоці нумеруються від 1 до m. Наприклад, нехай передається блок, що складається з 15-ти бітів: 1 1 0 0 1 1 1 0 1 0 1 0 1 0 1. Якщо розглядати випадок корекції лише двократних помилок і вважати, що помилки мали місце при передачі бітів з номерами x1 і x2, то сутність використання нелінійних зважених контрольних сум полягає в використанні деяких нелінійної функції ϕ(x) такої, що система:

$$\begin{cases} x_1 + x_2 = a \\ \varphi(x_1) + \varphi(x_2) = b \end{cases} \quad (0.1)$$

матиме єдиний розв'язок. Одним із можливих варіантів нелінійної функції ϕ(x) є формування її у вигляді ступеневої функції: ϕ(x) = x2.

Очевидно, що для виправлення помилок кратністю три потрібно використовувати ще одну додаткову функцію ψ(x), таку, щоб система:

$$\begin{cases} x_1 + x_2 + x_3 = a \\ \varphi(x_1) + \varphi(x_2) + \varphi(x_3) = b \\ \psi(x_1) + \psi(x_2) + \psi(x_3) = c \end{cases} \quad (0.2)$$

мала єдиний розв'язок. Одним із можливих варіантів нелінійної функції ψ(x) є формування її у вигляді ступеневої функції: ψ(x) = x3.

Для цього пропонується формувати контрольний код на приймачеві та передавачеві у вигляді трьох компонент C ={S1, S2, S3}. Якщо позначити через B={b1,b2,…,bn}, ∀j∈{1…n}: bj∈{0,1} блок даних, передача якого контролюється, то перша компонента S1 – це кількість одиничних бітів в інформаційному блоці за модулем 4:

$$S_1 = (\sum_{j=1}^{n} b_j) \bmod 4 \quad (0.3)$$

Другу компоненту S2 пропонується формувати як арифметичну суму порядкових номерів всіх одиничних бітів у блоці:

$$S_2 = \sum_{j=1}^{n} b_j \cdot j \quad . \quad (0.4)$$

Третю компоненту S3 контрольного коду пропонується обчислювати як арифметична суму квадратів порядкових номерів всіх одиничних бітів у блоці:

$$S_3 = \sum_{j=1}^{n} b_j \cdot j^2 \quad (0.5)$$

Компоненти контрольного коду CS обчисленого на передавачеві позначаються як CS ={S1S, S2S, S3S}. Контрольний код CS разом з блоком B передається на приймач. Останній по прийнятому блоку B′ згідно формул (0.3-0.5) обчислює контрольний код C R ={S1R, S2R, S3R}.

В порівнянні з відомими методами корекції двократних помилок, зокрема корегуючими кодами Хемінга та БЧХ, запропонований метод потребує більшої кількості контрольних розрядів. Проте у сучасних умовах зростання швидкості передачі даних між компонентами комп'ютерних систем, яка сягає десятки і сотні Мегабіт за секунду, значимість передачі додаткових десяток бітів не є значною.

Корекція багатократних помилок на основі логічних ступеневих зважених контрольних сум

Арифметичні операції, що виконуються на полях Галуа, займають важливе місце в сучасних інформаційних технологіях та лежать в основі у ряді алгоритмів захисту інформації. [5] Базовими операціями на полях Галуа GF (2m) є додавання та множення їх елементів.

Пропонується наступна варіація способу виправлення двократних помилок передачі даних.

Передавач відправляє N-бітовий інформаційний блок $B_S = \{b_1^S, b_2^S, ...b_N^S\}$, $\forall l \in \{1,...,N\} : b_l \in \{0,1\}$. Приймач отримує блок $B_R = \{b_1^R, b_2^R, ...b_N^R\}$. Необхідно визначити, чи є блок BR еквівалентним блоку BS, і, якщо BR містить помилки, провести корекцію. Для цього разом з інформаційним блоком BS передається контрольний код, що складається з трьох компонент:

1) компонента σ_S - сума за модулем два логічних добутків бітів блоку та їх номерів:

$$\sigma_S = \bigoplus_{i=1}^{N} b_i^S \cdot i \qquad (0.6)$$

2) компонента ξ_S, що обчислюється у вигляді:

$$\xi_S = \bigotimes_{i=1}^{N} (b_i^S \cdot i) remP \qquad (0.7),$$

де P - просте число в арифметиці без переносів розрядністю 2n+1, тобто число, яке співвідноситься з простим поліномом P(x) степені 2n+1; ξ - остача від поліноміального ділення на P.

3) компонента ω_S - останні 3 біти арифметичної суми бітів блоку:

$$\omega_S = \sum_{i=1}^{N} b_i^S \bmod 8 \qquad (0.8).$$

Обчислювальна складність O(log2m) розв'язання двох систем бітових рівнянь суттєво менша за складність локалізації спотворених бітів за допомогою циклічних кодів, в тому числі кодом Ріда-Соломона, яка становить O(m). Таким чином, запропонований спосіб корекції двократних помилок на основі зважених контрольних сум дозволяє за рахунок спеціалізації зменшити, в порівнянні з відомими методами, кількість контрольних розрядів та суттєво прискорити виконання процедури корекції помилок.

Висновки

Розроблено спосіб виправлення багатократних помилок передачі даних, що має за основу використання арифметичних степеневих зважених контрольних сум, за рахунок чого локалізація помилкових бітів в блоці зводиться до розв'язання системи яке трансформується в розв'язання квадратного рівняння. Таким чином, метод, на відміну від аналогічних модифікацій коду Хемінга та БЧХ, для розв'язання системи рівнянь не використовує перебір, що забезпечує виграш в часу корекції, пропорційний довжині блоку даних.

Розроблено модифікацію запропонованого способу виправлення багатократних помилок передачі даних яка відрізняється тим, що степеневі зважені контрольні суми обчислюються з використанням операцій додавання за модулем 2 та поліноміального множення на кінцевих

полях Галуа, за рахунок чого зменшується обчислювальна складність процедури локалізації пошкоджених при передачі бітів інформаційного блоку.

Перелік посилань

1. Федоречко О.И. К вопросу о получении неразложимых полиномов на полях Галуа для систем защиты информации / Федоречко О.И., Уваров Н.В., Исаченко Г.В. // Вісник Національ-ного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+. – 2012. – № 55. - С.157-162.
2. Кондратенко Ю.П., Мохор В.В., Сидоренко С.А. Verilog-HDL для модели-ро-ва-ния и синтеза цифровых электронных схем. Николаев.: Из-во НАУКМА. 2000.- 208 с.
3. Plank, J., Xu, L.: Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications. // In: Network Computing and Applications, - 2006. - NCA 2006. Fifth IEEE International Symposium. - pp. 173 –180.
4. Koetter R., Vardy A. Algebraic Soft-Decision Decoding of Reed_Solomon Codes // IEEE Trans. Inform. Theory. 2003. V. 49. . 11. P. 2809-2825.
5. Lin S., Costello D.J. Error Control Coding. Fundamentals and Applications. N.Y.: Prenticce Hall, 1983 – 396 p.

УДК 004.383

Хоменко Т.В.,
Сімоненко А.В.

ТЕОРЕТИЧНІ ОСНОВИ ПИДВИЩЕННЯ ЕФЕКТИВНОСТІ УГОРСЬКОГО АЛГОРИТМУ

В статті розглядається принцип підвищення ефективності Угорського алгоритму пошуку максимального паросполучення для дводольного графа. Покозано, що Угорський алгоритм можна використовувати для проектування просторових планувальників задач в розподілених неоднорідних обчислювальних системах і глобальных GRID систем. Представлені теореми, які дозволяють для розріджених дводольних графів, що відображають претендування заявок на ресурси, зменшити часову складність алгоритмів розподілення завдань на ресурси для неоднорідних систем. Показано, що використання такого підходу зменшує складність Угорського алгоритму з $O(n^3)$ до $O(n^{1,5}\log n)$. З огляду на те, що запропонований алгоритм є адаптивним, а пошук максимального паросполучення виконується в розрідженій матриці з коефіцієнтом заповнення менше 30%, то статична часова складність його менша $O(n^{1,5}\log n)$ та близька до лінійної.

The paper considers the principle of efficiency upgrading of Hungarian algorithm for finding maximum matching of a bipartite graph. It is shown that Hungarian algorithm can be used for spatial task planners design in distributed heterogeneous computer systems and global GRID systems. The theorems presented allow the sparse bipartite graphs displaying resource’s request to reduce the time complexity of algorithms of distributing tasks for resources for heterogeneous systems. It is shown that the use of the approach reduces the time complexity of the Hungarian algorithm from $O(n^3)$ to $O(n^{1,5}\log n)$. Taking into account that the algorithm is adaptive, and the search for maximum matching was performed in a sparse matrix with a volume efficiency of less than 30%, the statistical time complexity is less than $O(n^{1,5}\log n)$ and it is closer to the linear function.

Вступ

Розподілені обчислювальні системи та глобальні GRID-системи є новим поколінням обчислювальних систем, які використовуються, в основному, для наукових обчислень. По мірі розвитку таких систем виникають проблеми ефективного розподілення задач в них [4-6]. Однією з проблем великої розподіленої системи або глобальної GRID-системи, є залучення у обчислювальний процес максимального числа ресурсів, і відповідно, розподілення більшого числа задач по ресурсам. В більшості систем використовується планувальник потокового типу, що визначає або випадковий вибір вузла для задачі, або вибір вузла у відповідності з системою пріорітетів. В цьому випадку вплив призначення заявки на ресурс при послідуючих призначеннях не враховується.

Постанова задачі

Метою даної роботи є опис теоретичних основ для розробки планувальників нового типу. Представлений в роботі підхід дозволяє враховувати унікальні властивості дводольного графа та на основі інформації, одержуваної від системи моніторингу обчислювальних ресурсів, розробити адаптивний алгоритм направленого пошуку (АНП) розподілення задач по ресурсам з меншою часовою складністю, ніж відома.

Теоретичні основи розробки модифікованого алгоритма

В основу найбільш відомих алгоритмів пошуку максимального паросполучення в довільному графі покладені два основних підходи: зведення задачі до пошуку максимального потоку в мережі та пошуку зростаючого шляху від вільних вершин. В основу пошуку зростаючого чергуючого шляху покладена схема Дініца та розроблений на його основі алгоритм Хопкрофта-Карпа [1-3]. Відомі алгоритми, які реалізовують цей підхід, мають поліноміальну часову складність. Однак ці алгоритми та програми, що їх реалізують, мають складну структуру або призначені для часткових випадків і не забезпечують прийнятних часових показників, що обмежує застосування їх в системах диспетчеризації в розподілених обчислювальних системах.

Щоб спростити рішення задачі пошуку максимального паросполучення, пропонується розділити його на декілька етапів, коли власне рішенню передує швидкий аналіз вихідної інформації та вироблення рекомендацій для її подальшого рішення. Додаткові кроки значно меншої часової складності, ніж сам алгоритм, не впливають на часову складність алгоритма в цілому, однак дозволяють зменшити розмірність рішення задачі за рахунок виділення призначень, які обов'язково треба зробити та виділити, і які, безумовно, робити не можна, та за рахунок цього запобігти зайвої перевірки на можливість включення їх в рішення.

Теоретичною основою представленого підходу є наступні теореми та наслідки.

Теорема 1. Якщо в матриці $C_{RJ}[i,j]$ графа $G=(V_R,V_J,E)$, існує рішення A потужністю n = N і існує таке призначення (p,q), що $C_{RJ}[p,q]=1$, $C_{RJ}[p,j]=0$, $j\in\{1,..,N\}\backslash q$ та/або $C_{RJ}[i,q]=0$, $i\in\{1,..,N\}\backslash p$, тоді це призначення (p,q) завжди бере участь в рішенні A, тобто, $(p,q)\in A$.

Тобто, якщо для незваженого дводольного графа існує досконале паросполучення та в графі є вершина, якій інцидентне одне ребро, то ребро, інцидентне цій вершині, та вершини, інцидентні цьому ребру, обов'язково входять в досконале паросполучення.

Визначення 5. Призначення (p,q) за теоремою 1 називається обов'язковим.

Наслідок. Якщо існує рішення A^*, то задачу призначення можна розділити на дві частини: в першій частині беруть участь тільки обов'язкові призначення (p,q), в другій — залишилися призначення із нової матриці зв'язності C^*_{RJ}, яку отримують після видалення рядків і стовбців відповідних обов'язковим призначенням, визначеним за Теоремою 1. Причому, зважаючи на редукцію графа і відповідну корекцію матриці зв'язності, цей наслідок може застосовуватись рекурентно.

Теорема 1*. Будь-яка з вершин в незваженому дводольному графі, має ступінь, який дорівнює одиниці, завжди бере участь в одному із варіантів максимального паросполучення.

Теорема 1* справедлива як для вершин-заявок, так і для вершин-ресурсів. В тому випадку, якщо вершини зі ступінню 1 утворюють віяло, то Теорема 1* справедлива для будь-якої вершини, що входить в віяло, та кожна з них может бути взята в паросполучення, а перевірку інших вершин на можливість отримання збільшуючого шляху виконувати не слід.

Теорема 2. Якщо в матриці $C_{RJ}[i, j]$ існує віяло E_{FA}:

$E_{FA} = \{(R^1, q),...,(R^f, q)\}$, $C_{RJ}[R^k, q] = 1$, k = 1,…,f и $C_{RJ}[R^k, J^k] = 0$, $J^k \neq q$

або

$E_{FA} = \{(p, J^1),...,(p, J^f)\}$, $C_{RJ}[p, J^k] = 1$, k = 1,…,f и $C_{RJ}[R^k, J^k] = 0$, $J^k \neq p$,

тоді будь-яка з вершин із E_{FA} входить в один із варіантів максимального паросполучення, досконале паросполучення не може бути отримане і потужність максимального паросполучення визначається з виразу M < N – f +1.

Наслідок 1. Потужність рішення задачі пошуку максимального паросполучення може бути зменшена на кількість пар вершин, визначених за Теоремами 1 і 1*, та пошук паросполучення повинен вестися з нового суграфа.

Наслідок 1*. Розмірність рішення задачі пошуку максимального паросполучення повинна бути зменшена на кількість вершин, що входять в віяло, а пошук паросполучення повинен вестися з нового суграфа.

Наслідок 2. Суміжні ребра, інцидентні вершинам, які визначені за Теоремою 1, повинні бути видалені з подальшого розгляду, а вихідний граф редукований та перетворений в новий суграф.

Наслідок 3. Суміжні ребра, інцидентні вершинам, які визначені за Теоремою 2, повинні бути видалені з подальшого розгляду, а вихідний граф редукований та перетворений в новий суграф.

Оцінка часової складності алгоритму розподілення

Основою АНП являється Угорський метод (алгоритм), який має часову складність $O(n^3)$, де n є кількістю вершин графа вихідної матриці завдань-ресурсів. Для процедури пошуку максимального паросполучення використовують розроблений алгоритм пошуку максимального паросполучення, замість традиційного методу Карпа-Хопкрофта, який був використаний в Угорському методі. Часова складність алгоритму Карпа-Хопкрофта відома як $O(n^{1/2}m)$ або $O(n^{2,5})$, враховуючи $n \leq m \leq n^2$, де m – кількість дуг графа.

Для оцінки часової складності АНП, виконується аналіз Угорського алгоритму, де замість алгоритму Карпа-Хопкрофта, використовується адаптивний алгоритм мультианалізу (АМА).

Розрахунок часової складності АНП

Часова складність алгоритму АНП, реалізованого на основі запропонованого методу покрокового конструювання, складається із суми оцінок складності окремих чотирьох кроків. Складність процедури підготовки вихідної інформації та аналізу (Крок 1) залежить від кількості ребер у вихідному графі, а оскільки при цьому виконуються звичайні операції по формуванню матриці зв'язності або списків інцидентності, то складність виконання цього кроку дорівнює O(E).

З огляду на те, що в запропонованому алгоритмі, окрім матриці зв'язності, формуються ще й вектори ступеней вершин графа, то часова складність цієї процедури збільшується на O(2n). Тоді загальна часова складність виконання цієї процедури дорівнює O(E+2n). Виконання грубого аналізу дозволяє виділити ізольовані вершини і для цього потрібен одноразовий перегляд векторів , що визначає часову складність виконання цього кроку і дорівнює O(n). Другий крок пошуку обов'язкових призначень має часову складність O(n) у випадку кожного виявлення вершини зі ступенем одиниця і редукції вихідного графа часової складності O(2n), а у випадку єдиного рішення, після n кроків, може бути отримане повне рішення. Тоді часова складність знаходження повного розв'язку дорівнює O(n(n-1)/2+E). Загальна часова складність цієї процедури дорівнює O(n+2n)=O(3n). Часова складність кроку 3 визначається обраним алгоритмом сортування одновимірного масиву та складністю аналізу перетвореної матриці зв'язності та її корекції і дорівнює O(2nlog n + E/2 + E/2).

Загальна часова складність алгоритму направленого пошуку (АНП):

$ХА_{НП} = O(E + 2n + n + 3n + 2n\log n + E)=O(2E + 6n + 2n\log n + E) = O(3E + 2n(3+\log n))$.

З огляду на те, що на загальну оцінку складності алгоритму впливають кроки 1 і 4, а значення E в гіршому випадку дорівнює n^2, то складністю інших кроків можна знехтувати. Тоді часова складність алгоритму АНП дорівнює $O(E+n\log n)$ або $O(m+n\log n)$. Однак аналіз алгоритмів, які виконують пошук максимального паросполучення на основі теореми Бержа, за допомогою яких звичайно ілюструється роботоздатність запропонованих алгоритмів, дозволяє зробити висновок, що для більшості варіантів не вимагається виконання кроку 4 , отже і складність алгоритма отримання рішення можна зменшити для цього виду графів до $O(E)$ при використанні списків інцидентності, а при використанні матриці зв'язності до $O(n^2)$.

Таким чином, маємо часову складність Угорського методу в загальному випадку $O(n^{0,5}\times\max(X,n))$ або $O((n^{2,5}/m)\times\max(X,n))$, де $X=X_{АНП}=O[m+n\log n]$ и $n\le m\le n^2$. Тоді часова складність АНП дорівнює $O((n^{2,5}/m)\times[m+n\log n])=O(n^{2,5}+n^{3,5}\log n/m)=O(n^{2,5}\log n)$, коли $m=n$ або $O(n^{1,5}\log n)$, коли $m=n^2$. Слід зазначити, що часова складність АНП визначається для найгіршого випадку. При цьому матриця зв'язності заповнювалась повністю і $m=n^2$. Розрахункова часова складність АНП дорівнює $O(n^{1,5}\log n)$, що менше ніж $O(n^3)$.

Достовірність розрахунку часової складності перевірялась на програмній моделі. Результати статистичного дослідження представлені на рис. 1. З огляду на те, що АНП є адаптивним алгоритмом, а пошук максимального паросполучення виконується в разрідженій матриці з коефіцієнтом заповнення менше 30%, то статистична часова складність АНП менша $O(n^{1,5}\log n)$ та близька до лінійної.

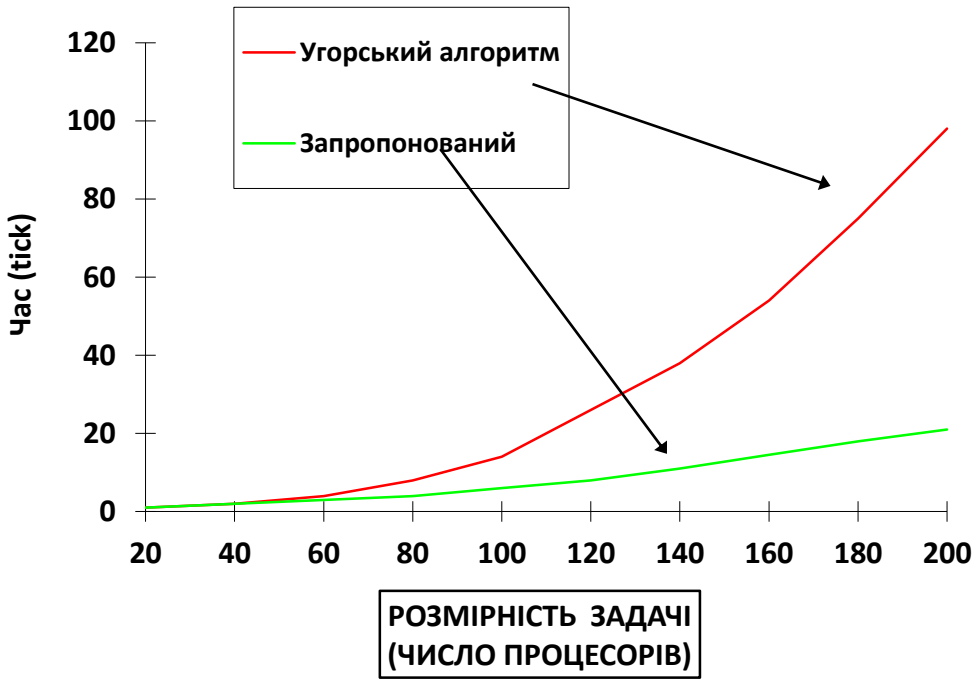


Рис. 1 Порівняння часів розв'язку задачі планування

Висновок

Класичний Угорський алгоритм в кожній ітерації виконує пошук максимального паросполучення для незваженого дводольного графа з використанням алгоритму Карпа-

Хопкрофта та має часову складність $O(n^3)$, яка визначається складністю $O(n^{2.5})$ алгоритмом Карпа-Хопкрофта.

Заміна алгоритма Карпа-Хопкрофта іншим алгоритмом з меншою часовою складністю дозволяє знизити загальну часову складність Угорського алгоритму. Використання відомого принципу, який виключає планування, сприяє в значній мірі підвищенню ефективності системи динамічного планування зі збереженням якості розв'язку в заданих межах.

Запропоновані в роботі процедури підготовки вихідних даних для планування в реальному часі дозволяють використовувати модифікований Угорський алгоритм як базовий для просторового планувальника в розподілених системах обробки інформації.

Перелік посилань

1.А. Кофман. Введение в прикладную комбинаторику.–М.: Наука, 1975, стр. 381-414

2.*Пападимитриу К., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. — М: Мир. — 1985, стр. 223-278

3.*Berge C.* Two Theorems in Graph Theory, Proc. National Acad. of Science USA, 43 (1957), pp. 842-844

4.*Douglas T.* Distributed computing in practice: The Condor experience / T.Douglas, T. Tannenbaum, M. Livny // Concurrency and Computation: Practice and Experience. — 2005. — No. 2. — pp. 323–356.

5.*Симоненко А.В.* Выбор стратегии пространственного планирования в параллельных вычислительных системах // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка, — Київ. — 2001. — **35.** — ст. 104–108.

6.Симоненко А. В., Пих С. В., Слуцкий Н. В., Воробйов В. В. Система пространственного распределения заданий в распределённых вычислительных системах // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — 2012. — **56.** — ст. 170-174

УДК 004.383

ЧЕРЕДНІЧЕНКО С.С.

Спосіб підвищення захищеності MPTCP

TCP / IP зв'язок в даний час обмежується одним шляхом на з'єднання, не зважаючи на те, що часто кілька шляхів існують між адресами. Одночасне використання цих кількох шляхів для TCP / IP сесії дозволить поліпшити використання ресурсів в межах мережі та, таким чином, підвищувати ефективність роботи користувачів за рахунок більш високої пропускної здатності і підвищення стійкості до збоїв у мережі. Багатошляховий TCP забезпечує можливість одночасного використання декількох шляхів між адресатами.

Хоча багатошляховість і збільшує пропускну здатність TCP з'єднання, вона також створює нові способи атак на MPTCP сесію, які не були можливі в стандартному одношляховому TCP.

Ціль роботи — огляд загроз, що створюються вразливостями MPTCP, щоб допомогти забезпечити рівень захищеності протоколу не гірший чим у одношляхового TCP.

TCP/IP connections in current internet restricted by one path per connection, despite aviability of multiple paths between hosts. Simultenious usage of such paths for TCP / IP session enables better utilisation of network resources by clients that leads to increase of throughput and resilience. MPTCP enables simultanious usage of multiple paths between peers.

Despite that multipath solution increase trhougput of TCP connection, it also creates new security threats for MPTCP by design.

Target of this work is to overview of threats that created by MPTCP related vulnerabilities and provide MPTCP with security on par with single path TCP.

Вступ

TCP / IP зв'язок в даний час обмежується одним шляхом на з'єднання, не зважаючи на те, що часто кілька шляхів існують між адресами. Одночасне використання цих кількох шляхів для TCP / IP сесії дозволить поліпшити використання ресурсів в межах мережі та, таким чином, підвищувати ефективність роботи користувачів за рахунок більш високої пропускної здатності і підвищення стійкості до збоїв у мережі. Докладніше з перевагами використання декількох шляхів для TCP можна ознайомитись в [1]. Багатошляховий TCP забезпечує можливість одночасного використання декількох шляхів між адресатами.

Хоча багатошляховість і збільшує пропускну здатність TCP з'єднання, вона також створює нові способи атак на MPTCP сесію, які не були можливі в стандартному одношляховому TCP.

Ціль роботи (зазначена в [2]) — огляд загроз, що створюються вразливостями MPTCP, щоб допомогти дизайнерам забезпечити рівень захищеності протоколу не гірший чим у одношляхового TCP.

Класифікація зловмисників і опис атак.

Класифікація зловмисників

Класифікація зловмисників за їхнім місцем знаходження:

- Зловмисник поза шляхом з'єднання;
- Зловмисник, що перебуває на шляху з'єднання певний проміжок часу;
- Зловмисник, що перебуває на шляху з'єднання впродовж всього часу з'єднання;

Класифікація зловмисників за поведінкою:

- Підслуховувач. Зловмисник прослуховує пакети, щоб пізніше здійснити атаку, але не в змозі змінювати, затримувати чи знищувати пакети.
- Активний атакуючий. Зловмисник в змозі змінювати, затримувати чи знищувати пакети.

Перелік і опис атак

Атака ADD_ADDR - Злом MPTCP сесії, що перетікає в MitM атаку. У цій атаці, зловмисник використовує опцію ADD_ADDR для злому поточного сеансу MPTCP, що дає можливість виконати MitM атаку на MPTCP сесію.

Хост А і Б мають MPTCP сесію, хост С — зловмисник.

Хост С надсилає ADD_ADDR пакет хосту Б з джерелом хоста А. Отримавши SYN+MP_JOIN пакет у відповідь від хоста Б, хост С зманює у пакета джерело на свій адрес і надсилає пакет хосту А. У відповідь хост А надсилає SYN/ACK+MP_JOIN пакет. Хост С знову змінює джерело пакета на свою адресу і надсилає це пакет хосту Б. Хост Б відповідає ACK+MP_JOIN пакетом. Хост С знову змінює джерело пакету на власну адресу і надсилає цей пакет хосту А. Хост А підтверджує валідність пакету і відкриває новий підшлях у з'єднанні. Останнім кроком хост С надсилає TCP RST пакети хосту А і Б з джерелами Б і А відповідно для розриву прямого підшляху між хостами А і Б. Хост С виконав повний взлом MPTCP сесії.

DoS атака за допомогою MP_JOIN — MPTCP DoS атака, що унеможливлює створення нових підшляхів.

За існуючими специфікаціями SYN+MP_JOIN при ініціалізації нового підшляху вводить хост (реципієнт) в стан очікування останнього (3-го) ACK. Кількість таких напіввідкритих з'єднань обмежена. Якщо зловмисник зможе підробити пакет SYN+MP_JOIN, то він зможе виснажити ціль атаки, створивши максимально можливу кількість напіввідкритих з'єднань, тим самим зробивши неможливим для цілі відкривати нові підшляхи у поточній MPTCP сесії.

Посилена атака SYN Flooding — зловмисник використовує SYN+MP_JOIN для ініціалізації SYN flooding атаки.

SYN flooding атаки використовують SYN повідомлення для того, щоб вичерпати ресурси сервера і запобігти створенню нових TCP з'єднань. Найрозповсюдженіша протидія такій атаці - це використання SYN cookies, які дозволяють обробку вихідного повідомлення SYN без входження сервера в стан очікування.

В MPTCP сесії SYN пакети обробляються, без входження хоста (реципієнта) в режим очікування, це досягається за допомогою вищезазначених SYN cookies. Але SYN+MP_JOIN створює новий вектор атаки, так, як вводить хост (реципієнт) в стан очікування. Зловмисник тепер може відкрити звичайну MPTCP сесію, відправляючи SYN і згодом посилати стільки повідомлень SYN + MP_JOIN, скільки максимально підтримується сервером з різними комбінаціями вихідної адреси і порту джерела, споживаючи ресурси сервера без особливого навантаження для себе.

Підслуховування ініціалізації з'єднання — зловмисник присутній під час ініціалізації з'єднання (під час обміну ключами), за допомогою чого може здійснити атаку на MPTCP сесію в будь-який момент життєвого циклу цієї сесії.

У цьому випадку, зловмисник присутній уздовж шляху, впродовж ініціалізації з'єднання і, отже, має можливість дізнатися ключі, які використовуються в MPTCP сесії. Це дозволяє зловмиснику залишити MPTCP сесію і все ще мати можливість зламати цю MPTCP сесію в майбутньому.

Атака SYN/JOIN — зловмисник не перехоплює SYN/JOIN пакет і змінює адресу, яка додається до MPTCP сесії.

Зловмисник присутній на шляху, під час обміну SYN / JOIN. Це дозволяє зловмиснику додати будь-яку нову адресу до MPTCP сесії просто замінюючи адресу джерела SYN / JOIN пакету на бажану.

Вразливість MPTCP до атаки цього роду зумовлена необхідністю розгортання MPTCP на мережі, що вже існує, а це означає підтримку NAT (можливість коректного функціонування через пристрої з NAT), а роботу NAT, на жаль, неможливо відрізнити від SYN/JOIN атаки. Єдиним виходом з ситуації є захист самого вмісту повідомлення, щоб зменшити простір для атаки зловмиснику.

Спосіб поліпшення захищеності для MPTCP

Для прийняття MPTCP, як стандарту, перш за все потрібно вирішити загрозу ADD_ADDR атаки. Потенційне рішення — використання HMAC (Hash-based message authentication code, хеш-код аутентифікації повідомлень), тобто використовувати адресу, що додається разом з ключем для генерації HMAC. Це прирівняє рівень захищеності ADD_ADDR пакету до рівня захищеності JOIN пакету. Також з цього випливає, що при поліпшенні захищеності алгоритму генерації і обміну ключами, буде поліпшений рівень захисту і MP_JOIN, і ADD_ADDR.

Ініціалізація сесії

Біт B заголовку опції MP_CAPABLE відповідає за наявність зовнішніх ключів для аутентифікації підшлях. Заголовок MP_CAPABLE показано на Рис 1. 1 — означає наявність ключа. Якщо з обох сторін біт B встановлено в 0, то для аутентифікації використовується механізм описаний в [3].

Багатошляхове TCP API

Для взаємодії додатку, що надає ключі і рівню MPTCP необхідно ввести 2 додаткові socket опції:

MPTCP_ENABLE_APP_KEY — опція вказує на те, що зовнішні ключі будуть використані, для захисту встановлення нових підшляхів. Ця опція повинна бути встановлена через ініціалізацією сесії.

MPTCP_KEY — ця опція надає можливість зовнішній програмі передати ключі в MPTCP рівень. Обидві сторони комунікації повинні встановити цю опцію, для надання дозволу MPTCP рівню створювати нові підшляхи.

Ініціалізація нового підшляху

Початковий обмін рукостисканнями аналогічний до описаного в [4] з деякими змінами. По-перше, HMAC генерується використовуючи надані зовнішні ключі. По-друге, Токен-А і випадкове число клієнта передається разом з 3-м АСК для дозволяння безстанової обробки пасивного ініціалізатора додаткових підшляхів.

Для того, щоб вмістити Токен-В і R-A в 3-й АСК, HMAC-A повинен бути обрізаною версією 160-бітного HMAC-SHA1. Тому HMAC-A обрізається (лівіше перших 128 бітів), як показано на Рис. 2.

Формат повідомлень MP_JOIN в SYN і SYN/ACK аналогічний до [3].

Розгортання на поточній інфраструктурі

По-перше модифікація TLS/ SSL-рівня, для розгортання MPTCP рукостискання з використанням HMAC.

По-друге, зовнішні ключі мають бути надані MPTCP рівню за допомогою сокет опції.

Висновок

MPTCP перспективний розвиток TCP, але сам принцип багатошляховості створює нові проблеми в захищеності протоколу. Було класифіковано зловмисників за поведінкою і місцем знаходження під час атаки, а також описано типи атак, до яких додатково вразливий MPTCP (виключаючи атаки до яких вразливий звичайний TCP).

Було запропоновано спосіб вирішення ADD_ADDR атаки, а також вказано на існуючу проблему SYN/JOIN атаки, що зумовлена необхідністю підтримки NAT.

Перелік посилань

1. Raiciu C., Paasch C., Barre S., Ford A., Honda M., Duchene F., Bonaventure O., Handley M., How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP // www.cs.princeton.edu 2012. [Електронний ресурс]. URL: https://www.cs.princeton.edu/courses/archive/fall14/cos561/papers/MPTCP12.pdf – назва з екрану.

2. Ford A., Raiciu C., Handley M., Barre S., Iyengar J., Barre S., Iyengar J., Architectural Guidelines for Multipath TCP Development // tools.ietf.org 2011. [Електронний ресурс]. URL: https://tools.ietf.org/html/rfc6182 – назва з екрану.

3. Ford A., Raiciu C., Handley M., Bonaventure O., TCP Extensions for Multipath Operation with Multiple Addresses draft-ietf-mptcp-multiaddressed-10 // tools.ietf.org 2012. [Електронний ресурс]. URL: https://tools.ietf.org/html/draft-ietf-mptcp-multiaddressed-10 – назва з екрану.

4. Paasch C., Bonaventure O., MultiPath TCP Low Overhead draft-paasch-mptcp-lowoverhead-00 // tools.ietf.org 2012. [Електронний ресурс]. URL: https://tools.ietf.org/html/draft-paasch-mptcp-lowoverhead-00 – назва з екрану.

УДК 004.852(622.2)

ШОЛОМ Д. Л.
ТЄЛИШЕВА Т. О.

СПОСІБ АДАПТИВНОГО УПРАВЛІННЯ ПРОЦЕСОМ БУРІННЯ В УМОВАХ НЕВИЗНАЧЕНОСТІ

В поточній статті розглянуто застосування сучасних моделей машинного навчання та використання принципів методу міркування на основі прецедентів для вдосконалення та підвищення ефективності адаптивного управління процесом буріння свердловин. Наведено основні висновки з обширного дослідження особливостей процесу буріння в умовах невизначеності. Загальний підхід та пропонований спосіб прогнозування параметрів буріння з елементами самовдосконалення на основі існуючих знань демонструється на прикладі визначення оптимального показника осьового навантаження на долото.

The subject of the articleis an application of the modern methods of machine learning and case-based recognition to a task of improvement andefficiency raise of adaptive well drilling control. The main conclusions from research of drilling process and existing approaches are presented. The proposed method to prediction of main parameters of well drilling process with machine leaning approach is demonstrated using determination of optimal bit on weight parameter as example.

Актуальністьпроблеми

Проблема підвищення ефективності та зниження вартості буріння нафтових та газових свердловин вже тривалий час є предметом наукових та прикладних досліджень, проте залишається актуальною й на сьогодні. Незважаючи на те, що є чимало теоретичних розробок та певних практичних реалізацій систем, що дозволяють автоматизувати процес буріння майже повністю та дозволяють обійтися без втручання людини в процес, на практиці все ж поширеніший є ручне або напівавтоматичне управління процесом. Але оскільки, по перше, процес буріння є складним і динамічним, повна автономність його інтелектуальних складових в автоматизованих системах процесу буріння не є доцільною. По друге, процес поглиблення завжди проходить в умовах неповноти інформації, що дозволяє віднести задачу оптимального управління поглибленням до проблеми управління динамічними об'єктами в умовах невизначеності. А отже, як і показує досвід, роль фахівця – бурового майстра в управлінні процесом є ключовою[1,2].

Основна задача бурового майстра полягає в моніторингу за зміною параметрів буріння з метою забезпечення оптимального протікання процесу, а також відслідковуванню непередбачуваних та виключних ситуацій. Виходячи з отриманої інформації і здійснюється адаптивне управління буровою установкою. В випадку більшості існуючих варіантів управління можна виділити декілька пов’язаних з таким підходом проблем.

Спостереження за процесом на об’єкті буріння виконується через моніторинг об’ємного графу параметрів в режимі реального часу, що означає необхідність постійної концентрації уваги за поточним станом. Проте при великому об’ємі даних з різних джерел це спричиняє велике навантаження на інженера і досконало володіти ситуацією є практично неможливим. Певні важливі симптоми можуть бути пропущені. По друге, в випадку певних позаштатних ситуацій чи неефективності процесу, процес може зупинятися на тривалий час, оскільки оцінка ситуації та пошук рішень може вимагати затрат значного часудля аналізу раніше отриманого масиву параметрів з залученням експертів [3].

Предмет та мета дослідження

Основна мета даного дослідження – знаходження способу підвищення ефективності адаптивного управління процесом буріння з використанням новітніх інформаційних технологій.

Для більш глибокого вивчення даного питання було проведено дослідження ряду робіт в галузі оптимізації управління процесом буріння [1,2,3]. Розглянуто залежність різних показників ефективності процесу буріння, включаючи контрольовані, такі як навантаження на долото, частота обертання ротора; та впливу факторів середовища, таких як температура, тиск, густина та природа матеріалів.Визначено, що в сучасній науці так само гостро постає тема забезпечення надійності, коректності, стійкості та відновлюваності програмного забезпечення, що використовується в процесі буріння для підвищення дляефективності[4].

В результаті систематизації матеріалів дослідження було встановлено, що значна кількість розробок, які фокусуються на оптимізації параметрів буріння шляхом аналізу даних в режимі реального часу[5]. Проте існуючі рішення часто базуються на вдосконаленні та адаптації давно розроблених моделей, як наприклад оптимізація параметрів буріння методом множинної регресії на основі проведення експериментів[6]. Швидкі темпи й глобалізація процесів розвитку інформаційних технологій дозволяє застосовувати й випробовувати зовсім інші методи для отримання оптимальних результатів.

Розгляд інноваційних наукових підходів до показав велике практичне значеннята значні успіхи у застосуванні наукових знань про великі масиви даних та штучний інтелект у різних галузях та сферах економічної діяльності. На основі цього пропонується спосіб адаптивного управління бурінням свердловини з використанням методів машинного навчання(machinelearning, ML)та міркування на основі прецедентів (case-basedreasoning, CBR). Такий спосіб дозволить зняти перераховані вище обмеження та використати досвід для прогнозування параметрів ефективного процесу в залежності від спостережуваних факторів, а також додатково надає якісну оцінку розрахованому прогнозу.

Міркування на основі прецедентів

Суть підходу полягає в базуванні на накопиченні досвіду та подальшої адаптації рішення відомої задачі до нової. Прецедентний підхід дозволяє спростити процес прийняття рішення в умовах часових обмежень та наявності різного роду невизначеностей в процесі, вихідних даних та експертних знаннях, а також в випадку виникнення різного роду нештатних та аномальних ситуацій[7].

Подібні ситуації часто виникають в процесі буріння свердловини протяжністю в декілька кілометрів через різні геологічні формування та пласти. І для кожної свердловини характерним є виникнення як нових проблем, так і таких що вже мали місце в минулому. Підхід міркування на основі прецедентів дає можливість перевикористання попередніх знань про конкретні ситуації для запобігання та ефективного вирішення проблем[8]. На рисунку 1 подана структурна схема загального випадку CBR-циклу, адаптованої для використання в прогнозуванні ситуацій та підтримці прийняття рішення в процесі моніторингу буріння.

Таким чином, використання методу міркуванняна онові прецедентів та аналогій забезпечує спрощення та поліпшення ефективності адаптивного управління бурінням в умовах невизначеності. Забезпечується цешляхом аналізу потоку даних та параметрів в режимі реального часу, розпізнавання шаблонів, систематизація та візуалізація даних; передбачення проблем, пошук рішень та надання порад інженеру як адаптувати процес для підвищення ефективності та уникнення позаштатних ситуацій.

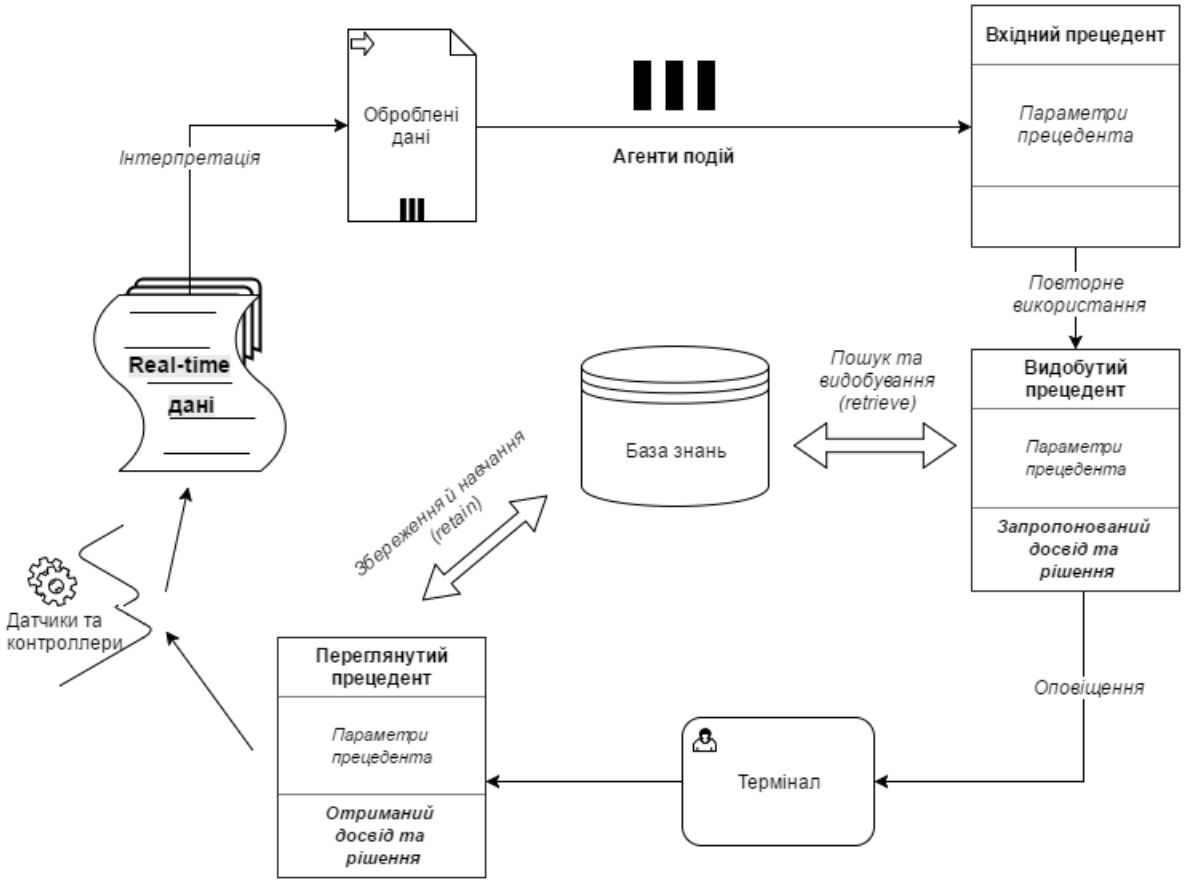


Рис 1. Структурна схема циклу міркування на основі прецедентів

Постановка задачіпрогнозуванняпараметрівбуріння (в загальномувигляді) за допомогоюML

Множина вхідних даних(навчальна вибірка) задається у вигляді вектора$x = \{X_1, X_2,..., X_i,..., X_M\}$та відповідних їм значень вихідної змінної$y = \{y_1, y_2,..., y_i,..., y_M\}$, де *i*-порядковий номер точки спостереження.

Спрощене рівняння швидкості проходки свердловини можна представити у вигляді

$$ROP = C_1 \cdot DR \cdot WOB^{c2} \cdot RPM^{c3},$$

де *ROP*(rateofpenetration)– швидкість проходки;
DR(drilling resistance) – буримість породи;
WOB(weightonbit) – навантаження на долото;
RPM(rotations per minute)– частота обертання ротора;
C1, C2, C3 – сталі.

Рівняння не описує процес буріння максимально адекватно, про те було вибрано, щоб абстрагуватися від надмірних деталей та складності та показати, що швидкість проходки залежить від певних контрольованих параметрів буріння. Одна з задач бурового майстра й полягає в контролюванні параметрів, щоб швидкість проходки залишалася оптимальною. В свою чергу, для кожного з параметрів можна встановити залежність від інших спостережуваних показників буріння. На основі цього й необхідно, маючи достатньо велику навчальну вибірку співвідношень спостережуваних показників та конкретного оптимального значення відповідного параметру, побудувати таку модель $Y_i = Y(X_i)$, яка зможе достатньо точно прогнозувати оптимальне значення параметру для будь якого набору спостережуваних показників.

ЗастосуванняML для визначення оптимального показника на прикладінавантаження на долото

Для достатньо великого масиву даних з навчальної вибірки, та отриманих показників сформуємо гіпотезу(в загальному вигляді):

$$h_\theta(x) = \theta^T x$$

де *x* – вектор параметрів, θ– вектор відповідних коефіцієнтів.

Для оцінки точності моделі на базі навчальної вибірки сформуємо функцію втрат E(θ):

$$E(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 (1)$$

Таким чином задача побудови оптимальної моделі зводиться до пошуку таких коефіцієнтів θ, за яких $E(\theta)$ прийматиме мінімальне значення.

Оскільки $E(\theta)$ було підібрана таким чином, що являє собою опуклу функцію з одним глобальним мінімумом, доцільним для визначення оптимальних коефіцієнтів θ є використання методу градієнтного спуску. Метод градієного спуску є ітеративним методом, який можна ефективно реалізувати в програмному вигляді, добре працює для великого значення n(кількості параметрів, що розглядаються) та є ефективним при правильному виборі коефіцієнту α.

Крок методу градієнтного спуску можна задати у вигляді функції:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} E(\theta), \text{ для } j=0,..,n (2)$$

Підставивши рівняння (1) в рівняння (2), отримаємо

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Для запобігання перенавчання виконаємонормалізацію моделі методом статистичної регуляризації:

$$\theta_j := \theta_j - \alpha \frac{1}{m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\beta}{m}\theta_j\right] =>$$

$$\theta_j := \left(1 - \alpha \frac{\beta}{m}\right)\theta_j - \alpha \frac{1}{m} \times\times \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Для застосування моделі, необхідно визначити, які ж саме параметри можна використати для визначення навантаження на долото. Для кращої демонстрації принципів даного способу без надмірного ускладнення прикладу було вирішено обмежитися випадком роторного буріння для твердих порід. При бурінні твердих крихких порід відбувається структурне руйнування породи по всьому об'єму(ΔV) на дрібні шматки різнорідної форми.

Таким чином, як відомо, роботу долота можна подати як

$$A = Fv\Delta t ,$$

де $\Delta t -$ час роботи долота,

а енергію, як

$$\Delta W = \Delta V \sigma_{\text{пов}} k = v\Delta t \cdot d \cdot l \cdot \sigma_{\text{пов}} k_n,$$

де $\sigma_{\text{пов}}$ – коефіцієнт поверхневого натягу матеріалу породи. Він характеризує сили міжмолекулярного тяжіння та визначає силу, необхідну для розриву міжмолекулярних зв'язків;

а k_n – коефіцієнт об'ємного руйнування, що характеризує крихкість породи, її здатність до розколювання на шматки аж до порохоподібного стану. Він залежить як від властивостей самої породи, так і від параметрів буріння та властивостей рідини для промивання – густини ρ, пластичної в'язкості n, динамічного напруження зсуву τ, тощо.

Таким чином можна показати, що сила руйнування породи, а відповідно й навантаження на долото залежить від зазначених вище параметрів, які можна враховувати при побудові моделі прогнозування оптимального навантаження на долото (для оптимізації швидкості проходки).

$$F = \frac{v\Delta t \cdot d \cdot l \cdot \sigma_{\text{пов}} k_n}{v\Delta t} = d \cdot l \cdot \sigma_{\text{пов}} k_n$$

Висновки

На основі аналізу предметної області та різноманітних способів, було показано на практичному прикладі можливість й доцільність застосування інтелектуальних технологій для оптимізації управління бурінням. В подальшому є доцільним проведення дослідження інших параметрів та формалізація їх у вигляді функцій (залежності контрольованих параметрів, що впливають на ефективність поглиблення свердловини) для формування аналогічних моделей їх адаптивного налаштування, а також дослідження способів вибору оптимальнішої моделі для гіпотези для здійснення ще точнішого прогнозування.

Перелік посилань

1. Семенцов Г. Н., Горбийчук М. И., Телышева Т. А. Способ и алгоритмы оперативного управленияпроцессомуглублениякважин. // Нефтянаяпромышленность. Серия: Автоматизация и телемеханизация в нефтянойпромышленности. – 1984.- №1. – с. 32-35.
2. Горбійчук М. І. Оптимізація процесу буріння глибоких свердловин : монографія / М. І. Горбійчук, Г. Н. Семенцов. – ІваноФранківськ : Факел, 2006. – 493 с
3. Crichton, M. T.; Lauche, K.; andFlin, R. 2005. IncidentCommandSkillsintheManagementofanOilIndustryDrillingIncident: A CaseStudy. JournalofContingenciesandCrisisManagement 13(3): 116–128.
4. Martha Belem, Saldivar Márquez, Islam Boussaada, HuguesMouni.Analysis and Control of Oilwell DrillingVibrations: A Time-Delay Systems. – Switzerland 2015
5. Booth, J. 2011. Real-TimeDrillingOperationsCenters: A HistoryofFunctionalityandOrganizationalPurpose — TheSecondGeneration. SPE Drilling&Completion 26(2): 295–302.
6. Bourgoyne A.T. Jr., Young F.S., "A Multiple Regression Approach to Optimal Drilling and Abnormal Pressure Detection", SPE 4238, August 1974
7. Aamodt, A., Plaza, E., Case-BasedReasoning: FundamentalIssues, MethodologicalVariations, andSystemApproaches. ArtificialIntelligenceCommunications, Vol. 7, No. 1, pp. 39-59, 1994.
8. Marling, C., Rissland, E., Aamodt, A., Integrationswithcase-basedreasoning. KnowledgeEngineeringReview, CambridgeUniversityPress. , 2005. 20(03): p. 241-245.

УДК 004.7

ЯЦУН В.О.,
ДОЛГОЛЕНКО О.М.

БЛОК ДОДАВАННЯ ТА ВІДНІМАННЯ МАНТИС З РЕКОНФІГУРОВАНОЮ СТРУКТУРОЮ

В цій статті описаний блок додавання/віднімання мантис з реконфігурованою структурою для суматора з плаваючою крапкою. Розроблений блок представлений як набір комбінаційних схем, без використання елементів пам’яті й не потребує мікропрограмного керування.

This article describes block addition / subtraction of mantissa adder reconfigurable structure to float. Developed block is represented as a set of combinational circuits without using memory elements and microprogram management.

Вступ

Починаючи з мікропроцесорів *Intel P6* (1995 р.) та *AmD K5* (1996 р.), для побудови мікропроцесорів з архітектурою *x86-64* використовується технологія *OoOE (Out-of-Order Execution)*, що заснована на реалізації обмеженої архітектури потоку даних [1]. Такі мікропроцесори отримали назву суперскалярних мікропроцесорів [2], або мікропроцесорів з архітектурою *CISC-RISC («CISC-outside RISC-inside»),* де: *CISC - Complex Instruction Set Computing* (набір команд в архітектурі *x86-64*), *RISC - Reduced instruction set computing* (скорочений набір команд, що реалізується множиною операційних пристроїв мікропроцесора.) В процесі роботи таких мікропроцесорів *CISC* – команди, що виконуються відповідно до лічильника команд, декодуються на множину *RISC*– команд [3-5] (їх також називають: *uop, micro-ops*, або подібними термінами).

Планування виконання *RISC* – операцій здійснюється, відповідно до архітектури потоку даних [6,1], на підставі готовності до виконання операндів виконуваних *RISC* – операцій. До набування *RISC* – операціями стану готовності до виконання, вони розміщуються в комірках найбільш швидкодіючої пам'яті мікропроцесора, що носять назву в різних джерелах: *Out-of-Order Window,* комірки універсального планувальника, або комірки станції резервування. Кількість цих комірок постійно збільшується, наприклад, у *Intel:* 168 в *Sandy Bridge* (2011 р.), 192 в *Haswell* (2013 р.), 224 в *SkyLake* (2015 р.). *RISC* – операція, котра набула стану готовності, може бути переданою в вільний операційний пристрій, що може її виконати. Таким чином в сучасних мікропроцесорах організовується динамічний паралелізм на рівні *RISC* – операцій.

Метою роботи є розробка блоку додавання/віднімання мантис з реконфігурованою структурою для швидкодіючого суматора/віднімача з плаваючою крапкою, який не потребуватиме в своїй роботі ніяких додаткових мікропрограмних засобів і може використовуватися для виконання RISC – операцій як в роботі ядра сучасного суперскалярного мікропроцесора так і, наприклад, при побудові спеціалізованих апаратних засобів на ПЛІС, керованих потоком даних.

Розробка блоку додавання/віднімання мантис з реконфігурованою структурою

Розроблюваний блок додавання/віднімання мантис для швидкодіючого суматора/віднімача з плаваючою крапкою орієнтований на обробку всіх форматів чисел з плаваючою крапкою, що передбачені стандартом IEEE Std 754™ [7]. Таких форматів п'ять: половинна точність *(SF)* - 16 розрядів, одинарна точність *(F)* - 32 розряда, подвійна точність *(DF)* - 64 розряда, подвійна розширена точність *(DEF)* - 80 розрядів та учетверину точність *(QF)* (128 розрядів). Для досягнення цієї мети блок додавання/віднімання мантис повинен мати можливість оброблювати, відповідно формату: 12, 25, 54, 66 та 114 бітні мантиси (разом із знаковим та прихованим бітом). Структурна схема такого блоку показана на рис. 1. Схема складається з п'яти суматорів, поєднаних між собою за допомогою однорозрядних двухвходових мультиплексорів. Розглянемо роботу схеми та прикладі виконання операції додавання/віднімання мантис: $\boldsymbol{s=x\pm y}$. На нулеві входи однорозрядних мультиплексорів та на вхід переноса четвертого суматора ($\sum_4$) при виконанні віднімання з блоку керування подається код 1, що забезпечує додавання одиниці до числа, що подається на входи $\boldsymbol{y}$ суматорів. Для того, щоб в результаті цієї операції на входах суматора формувався додатній код мантиси $\boldsymbol{y}$, схема містить п'ять мультиплексорів тієї ж розрядності, шо й відповідний суматор. На керуючі входи цих мультиплексорів з блоку керування подається код 1, що забезпечує передачу на суматори зворотного коду мантиси y. В цілому, ці описані дії забезпечують передачу на суматори при виконанні операції віднімання додаткового коду від зображення мантиси $\boldsymbol{y}$.

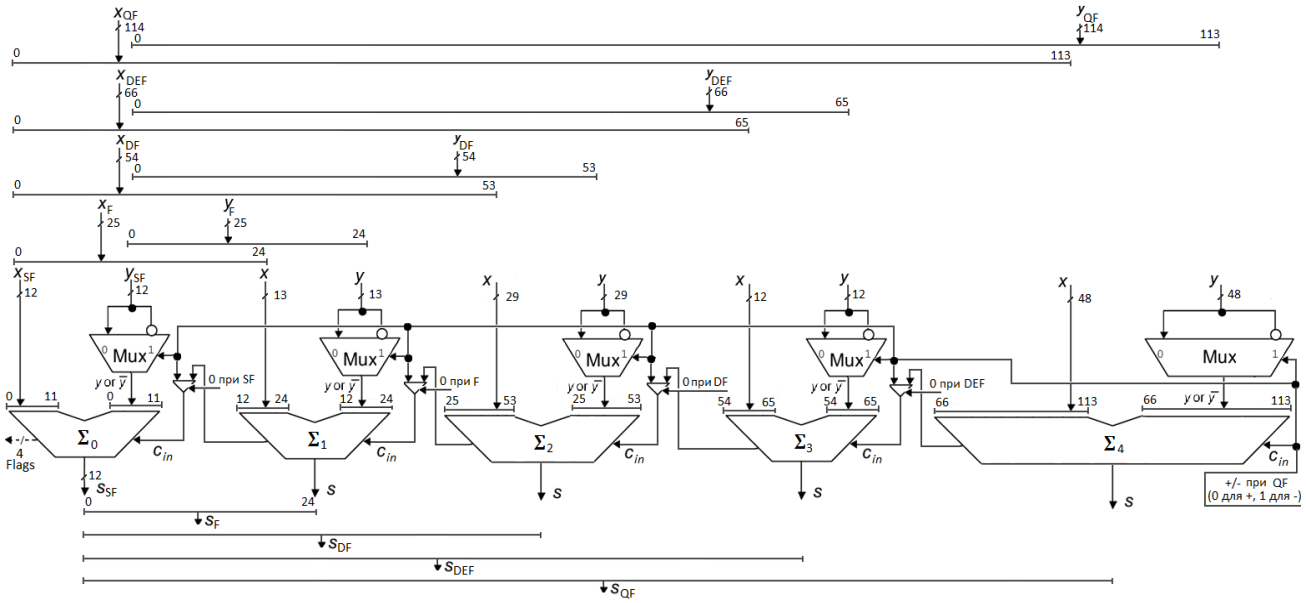


Рис. 1. Структурна схема блоку додавання/віднімання мантис з реконфігурованою структурою

При виконанні операції додавання, з блоку керування на вищеописані входи подається код 0, що забезпечує передачу на суматори прямого коду мантиси *y*.

Мантиси *x* і *y*, при обробці операндів формату *QF*, подаються на входи всіх п'яти суматорів. При цьому на керуючі входи всіх однорозрядних мультиплексорів з блоку керування на вищеописані входи подається код 1, що забезпечує передачу вихідного переносу суматора $\sum_i$, через перший вхід *i*-го однорозрядного мультиплексора на вхід переноса суматора $\sum_{i-1}$.

Мантиси ***x*** і ***y***, при обробці операндів формату *DEF,* через відповідні мультиплексори подаються на входи чотирьох старших суматорів ($\sum_0$ - $\sum_3$). При цьому на керуючий вхід однорозрядного мультиплексору, що зв'язаний своїм виходом із входом переноса $\sum_3$, з блоку керування подається код 0, що забезпечує блокування вихідного переносу суматора $\sum_4$ та передачу на 4 старших суматори схеми, в залежності від виконуваної операції, прямого чи додатного коду від зображення мантиси ***y***.

Для мінімізації затримки розповсюдження сигналу в блоці додавання/віднімання мантис був проведений аналіз вибору конкретної схеми реалізації суматорів $\sum_0$ - $\sum_4$. Проведений аналіз показав, що мінімальна затримка досягається при реалізації 12-розрядних суматорів $\sum_0$ та $\sum_3$ по схемі суматора з вибірковими [8] переносами, що показана на рис. 2.

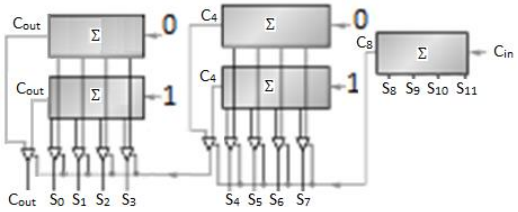


Рис. 2 Структурна схема суматорів $\sum_0$ та $\sum_3$ блоку додавання/віднімання мантис з реконфігурованою структурою

Всі групи суматорів $\sum_0$ та $\sum_3$ з рис. 2 реалізуються по схемі суматора з попередніми переносами та мають затримку в 5 логічних рівнів. Загальна затримка розповсюдження сигналу в кожному з суматорів $\sum_0$ та $\sum_3$ складає 9 логічних рівнів.

На рис. 3 показана структурна схема 29- розрядного суматора $\sum_2$.

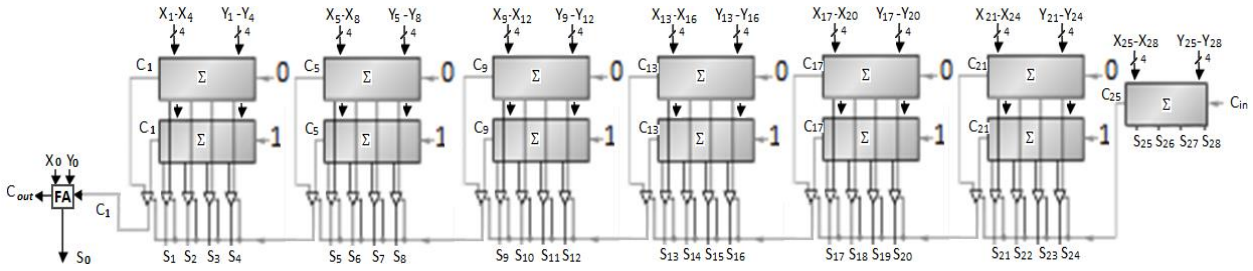


Рис. 3 Структурна схема суматора $\sum_2$ блоку додавання/віднімання мантис з реконфігурованою структурою

Якщо реалізувати 48-розрядний суматор $\sum_4$ також за схемою суматора з вибірковими переносами, близькою до схем з рис. 2 та 3, то він буде складатися з 12-ти чотирьохрозрядних групових суматорів, а загальна затримка розповсюдження сигналу складе 27 логічних рівнів: 5 рівнів затримки в кожному з чотирьохрозрядних групових суматорів (із зменшенням технологічних норм виробництва інтегральних схем зменшуються допустима навантажність логічного елементу по виходу – у зв'язку з цим неможливо побудувати схему суматора з попередніми переносами більше ніж на 4 розряди) плюс 2 рівнів затримки в кожному з 11 мультиплексорів вибору результату наступної групи.

Реалізація суматора $\sum_4$ по схемі практичного *CLA* [8] матиме загальну затримку розповсюдження сигналу: $4\log_2 n+3$ та при n=48 складе також 27 логічних рівнів. Однак, реалізація по схемі 1.3 потребує значно менших апаратурних затрат.

З метою подальшого зменшення апаратурних затрат розглянемо можливість реалізації $\sum_4$ по схемі суматора з обхідними переносами [8]. Для цього розіб'ємо 48-розрядний суматор $\sum_4$ на 13 груп: 2 зовнішні групи складаються з 2 однорозрядних суматорів, а всі 11 внутрішніх груп є чотирьохрозрядними. Всі групові суматори є найпростішими, реалізованими за схемою суматора з наскрізними переносами. Схема такого суматора $\sum_4$ приведена на рис. 4.

Розрахуємо загальну затримку розповсюдження сигналу в цьому суматорі: $4*2+2(13–2) = 30$ логічних рівнів. При цьому найгірший випадок може виникнути коли в двох передостанніх 4-розрядних групах суматора підряд $P=0$, а фактично перенос буде розповсюджуватися в 3 старших суматорах попередньої групи та 3 молодших суматорах наступної групи, що в сумі потребує ще $2*6$ логічних рівнів затримки для формування значень суми S_{3-8}. Сумарна затримка в старшій групі та обхідних ланцюгах передостанньої групи складає всього 6 логічних рівнів затримки. У зв'язку з цим, аби $\sum_4$ формував кінцевий результат обчислень то його сумарна затримку слід було б збільшити на 6 і вона склала б 36 логічних рівнів. Але зміна станів розрядів суми S_{3-8} ніяк не може впливати на значення переносу C_2 з цієї групи, що уже давно був сформований. Поки значення цього переносу буде впливати на обчислення в суматорах $\sum_0 – \sum_3$, розряди суми S_{3-8} будуть мати більш ніж достатньо часу для зміни своїх станів. У зв'язку з цим затримка розповсюдження сигналу в варіанті реалізації суматора $\sum_4$ складає, все таки, 30 логічних рівнів.

Таким чином, реалізація суматора $\sum_4$ блоку додавання/віднімання мантис з реконфігурованою структурою за схемою суматора з обхідними переносами, в порівнянні з іншими розглянутими варіантами його реалізації, має всього на 3 логічних рівня більшу затримку, але потребує набагато менших апаратурних затрат. Враховуючи, що $\sum_4$ буде

використовуватися тільки при обробці молодших розрядів мантис з точністю *QF*, доцільно реалізувати $\sum_4$ саме за схемою з рис. 4.

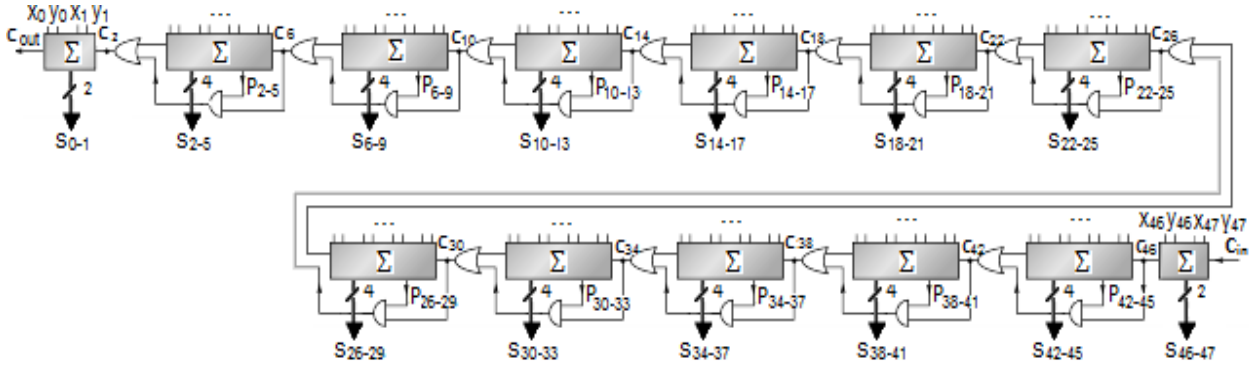


Рис. 4 Структурна схема суматора $\sum_4$ блоку додавання/віднімання мантис з реконфігурованою структурою

Висновки

В цій роботі описана структурна схема блоку додавання/віднімання мантис з реконфігурованою структурою для швидкодіючого суматора/віднімача з плаваючою крапкою. Перевагами розробленої схеми є те, що вона виконана як набір комбінаційних схем, без використання елементів пам'яті й не потребує мікропрограмного керування.

Виконана розробка буде корисною при проектуванні операційного пристрою сумування операндів, як з фіксованою так і з плаваючою крапкою, при побудові ядра мікропроцесора з суперскалярною архітектурою, сумісною з сімейством *x86-64* та при побудові спеціалізованих апаратних засобів на ПЛІС, керованих потоком даних.

Перелік посилань

1. Y. Patt, W. Hwu, et al, Experiments with HPS, a Restricted Data Flow Micro architecture for High Performance Computers, Digest of Papers, COMPCON 86, (March 1986), pp. 254-258.
2. John L. Hennessy, David A. Patterson. Computer Architecture. A Cuantitative Approach, USA, Morgan Kaufmann, 2012 – 497 p.+ add-ins.
3. Kanter, David (September 25, 2010). "Intel's Sandy Bridge Microarchitecture" (http://www.realworldtech.com/sandy-bridge/).
4. Kanter, David (November 13, 2012). "Intel's Haswell CPU Microarchitecture" (http://www.realworldtech.com/ haswell-cpu/).
5. Луцький Г.М., Долголенко О.М., Аксьоненко С.В., Сторожук В.О. Моделювання обмеженої реалізації архітектури потоку даних в структурі суперскалярного процесора. Київ,- Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 60. – с. 83-94.
6. J. Dennis: Data Flow Supercomputers; IEEE Computer, pp. 48-56, Nov. 1980.
7. IEEE 754: Standard for Binary Floating-Point Arithmetic [Электронный ресурс] / 3 апрель 2014. – URL: http://grouper.ieee.org/groups/754/.
8. Behrooz Parhami. Computer Arithmetic. Algorithms and Hardware Designs, New York, Oxford University Press, 2000 – 491 p.