УДК 681.3.06

А.О. Бойко

Приватне акціонерне товариство "Інститут інформаційних технологій"

УДОСКОНАЛЕННЯ МЕТОДУ УНІВЕРСАЛЬНОГО ГЕШУВАННЯ НА ОСНОВІ ОБЧИСЛЕННЯ ЗНАЧЕННЯ ПОЛІНОМА У СКІНЧЕННИХ ПОЛЯХ

Запропоноване удосконалення методу універсального гешування на основі обчислення значення полінома у скінченних полях шляхом використання паралельних обчислень . Удосконалений метод гешування дозволяє збільшити швидкодію пропорційно кількості обчислювальних ядер. Удосконалений метод універсального гешування на основі обчислення значення полінома у скінченних полях з використанням паралельних обчислень дозволяє гнучко налаштовувати параметри обчислень зберігаючи при цьому сумісність між системами з різними параметрами. роботі наведені розроблений паралельний алгоритм гешування і результати експериментальних вимірювань швидкодії і характеристик паралельних обчислень.

***Ключові слова**: універсальне гешування, паралельні обчислення.*

Вступ

Забезпечення автентичності інформації, яка передається і обробляється у інформаційно-телекомунікаційних системах – одна з основних задач криптографічного захисту інформації. У випадку, коли відправник і отримувач інформації довіряють один одному, то вони для контролю автентичності повідомлень у ході інформаційного обміну можливо використовувати коди автентифікації повідомлень.

Кількість інформації, яка передається каналами зв'язку, невпинно зростає, що висуває вимоги до швидкодії засобів контролю автентичності даних. В той же час, виробники процесорів для збільшення швидкодії перестали нарощувати тактову частоту, а замість того, зробили акцент на нарощуванні кількості обчислювальних ядер. Тому актуальною є проблема збільшення швидкодії методів автентифікації даних за рахунок використання паралельних обчислень.

Постановка задачі. Вперше використання паралельних обчислень для збільшення швидкодії функцій гешування було запропоновано у роботі Дамгарда [1]. Однак, через відсутність у той час багатоядерних процесорів та багатопроцесорних ЕОМ у широкому вжитку на той час, запропонований ним підхід обчислення геш-значення по дереву широко не обговорювався. Розвитком роботи Дамгарда в частині використання паралельних обчислень є робота [2], у якій запропоновано використовувати бінарне дерево обмеженої висоти, як більш зручне з практичної точки зору, і запропонований алгоритм симуляції бінарного дерева висоти t на 2^s обчислювальних ядрах, де $s \le t$. Однак недоліками запропонованого у [2] метода є низька ефективність використання паралельних обчислень на процесорах, у яких кількість обчислювальних ядер не кратна 2^t. Вирішення цієї проблеми запропоновано у роботі [3], де запропоновано параметри дерева геш-значень, які дозволяють досягнути оптимальної ефективності паралельних обчислень на довільному числі обчислювальних ядер.

Все сказане вище справедливе і для кодів автентифікації повідомлень, які є частковим випадком функцій гешування, а саме ключовими функціями гешування. Одним із перспективних підходів до побудування високошвидкісних кодів автентифікації повідомлень є використання універсальних функцій гешування. Реалізацією цього підходу є стандарт UMAC, в основі якого лежить універсальна функція гешування PolyCW. Універсальна функція гешування PolyCW є реалізацією методу універсального гешування на основі обчислення значення полінома у скінченних полях. Обчислення значення полінома у скінченних полях здійснюється за схемою Горнера, тобто алгоритм є послідовним. Використання цього методу разом з архітектурою, запропонованою у роботі [3] має недолік, який полягає у несумісності реалізацій з різними параметрами дерева геш-значень. Тому актуальною є задача удосконалення методу універсального гешування на основі обчислення значення полінома у скінченних полях шляхом використання архітектури, запропонованої у роботі [3] із збереженням сумісності з існуючими послідовними реалізаціями методу.

Удосконалення функції універсального гешування на основі обчислення значення полінома у скінченних полях

Функція універсального гешування на основі обчислення значення полінома у скінченних полях

(далі – функція поліноміального гешування) обчислюється за формулою

$$h_x(m) = \sum_{i=0}^{k} m_i x^i \bmod p, \quad (1)$$

де p – просте число і k – ціле число, $k > 0$.

Формула (1) обчислюється з схемою Горнера:

$$h_x(m) = ((...(m_1 x + m_2)x + ...)x + m_k) \bmod p. \quad (2)$$

Як видно з формули (2), алгоритм є послідовним. Для того, щоб побудувати паралельну версію алгоритму обчислення, можна використати той факт, що обчислення полінома може бути виконане наступним чином:

$$\sum_{i=0}^{k}\left(x^i m_i\right) \bmod p = \\ = \sum_{j=0}^{\lceil k/n \rceil} x^{jn}\left(\sum_{g=0}^{n-1} x^g m_{j*n+g}\right) + \sum_{g=0}^{k \bmod n}\left(x^g m_i\right), \quad (3)$$

де k – число блоків повідомлення, n- число процесорних ядер.

Частини, що наведені у формулі (3) у дужках можуть бути обчислені паралельно максимум у $\lceil k/n \rceil$ потоків. При цьому на значення n не накладається жодних обмежень, що дозволяє гнучко підлаштовувати процес обчислення під властивості конкретної системи. Більш того, оскільки кінцеве геш-значення залишається незмінним згідно із формулою (3), то на різних системах, що взаємодіють, можливо побудувати процес обчислень по-різному, оптимально для кожної з систем, без втрати сумісності, чого не вдавалось досягти із звичайними геш-функціями.

Як показано у [3], така архітектура обчислень є оптимальною з точки зору паралельних обчислень.

Щодо імовірності колізій, то оскільки наведений паралельний варіант (3) для всіх значень ключів і повідомлень дає ті ж самі геш-значення, що і послідовний (2), то для нього залишаються вірними всі оцінки імовірності колізії.

Опис практичного паралельного алгоритму гешування

Алгоритм 1.

Вхід:

NumOfThreads – кількість потоків, що працюють одночасно;

m – повідомлення, до i-го блоку повідомлення можна звертатися як m_i;

Key – ключ;

SbSize – розмір суперблока (під суперблоком мається на увазі послідовність блоків повідомлення, що обробляються одним потоком як єдине ціле).

Вихід:

h – геш-значення.

1 $\text{SuperBlockKey} = \text{Key}^{(\text{SbSize}*(\text{NumOfThrreads}-1))}$

2 Виділити пам'ять для проміжних геш-значень по числу потоків ThreadHTemp

3 Паралельно у NumOfThreads потоків виконувати наступне (ThreadNum – номер потока)

3.1 $\text{ThreadSbNumber}_{\text{ThreadNum}} = \text{ThreadNum}$

3.2 Поки наявні дані

3.2.1 $\text{ThreadHTemp}_{\text{ThreadNum}} = \text{ThreadHTemp}_{\text{ThreadNum}} * \text{SuperBlockKey}$

3.2.2 $\text{ThreadHTemp}_{\text{ThreadNum}} = \sum_{i=0}^{\text{SbSize}} \text{Key}^i \cdot m_{i+\text{SbSize}\cdot\text{ThreadSbNumber}} \bmod p$

3.2.3 $\text{ThreadSbNumber}_{\text{ThreadNum}} = \text{ThreadSbNumber}_{\text{ThreadNum}} + \text{NumOfThreads}$

4 $h = \sum_{i=0}^{\text{NumOfThreads}} \text{Key}^{i*\text{SbSize}} \cdot \text{ThreadHTemp}_i \bmod p$

В цьому алгоритмі поток номер ThreadNum обробляє суперблоки $\text{ThredNum} + k * \text{NumOfThreads}$, де k – ціле, $k \geq 0$.

Для оцінки теоретичного часу роботи цього алгоритму можна скористатися законом Амдала [4]:

$$T_{mt} = \frac{T_{st}}{N_{th}} + T_{tc} + T_{post}, \quad (4)$$

де T_{st} – час виконання однопоточної версії; T_{tc} – час на створення і запуск кожного потока; N_{th} – кількість потоків; T_{post} – час кінцевої обробки.

У період T_{tc} зараховуються кроки 1 і 2, у період T_{post} – крок 4. Як видно $T_{tc} + T_{post}$ значно менше, ніж T_{st}, тому

$$T_{mt} \approx \frac{T_{st}}{N_{th}}. \quad (5)$$

При практичній реалізації досліджувався вплив розміру суперблока і кількості потоків на час гешування. Було реалізовано три варіанти з розміром суперблоку 1, 10^3, 10^5 блоків. Кількість потоків змінювалась від 1 до 8. Розмір блоку складав 4 байта, сукупний розмір даних, що оброблялися алгоритмом приблизно 1,56 Гб ($4{,}2*10^8$ блоків по 4 байта). У табл. 1 наведено час гешування у секундах.

Для порівняння реалізовано однопоточну версію алгоритму. Час виконання однопоточної версії на тих же даних склав 3,34 с. В якості тестового стенду використовувався сервер з 8 обчислювальними ядрами під керуванням ОС Linux.

На рис. 1 показано графіки зміни часу виконання алгоритму в залежності від кількості потоків. Як видно, версія з малим суперблоком (1 блок) внаслідок значних накладних витрат на паралельну обробку (в основному, промахи мимо кеш-пам'яті процесора) має дуже великий час виконання і відстає навіть від однопоточної версії алгоритму.

Таблиця 1

Час гешування

Кількість потоків	Розмір суперблоку (блоків)		
	1	10^3	10^5
	Час виконання, с		
1	9,34	4,89	4,88
2	10,2	3,06	3,31
3	9,8	3,14	3,14
4	11,0	3,23	3,25
5	11,52	3,24	3,13
6	12,2	2,89	3,07
7	13,73	2,8	2,7
8	15,4	2,33	2,27

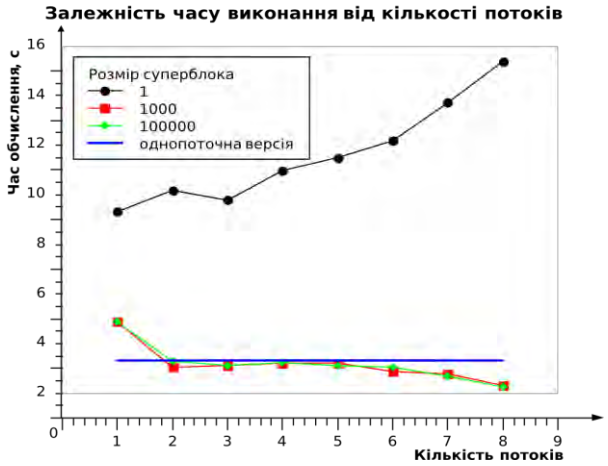

Рис. 1. Графік залежності часу виконання від кількості потоків

Однак між варіантами з розмірами суперблоку 10^3 і 10^5 блоків різниці практично немає, і обидва варіанта вже з 2-х потоків забезпечують виграш по швидкодії у однопоточної версії алгоритму.

Це свідчить, що розміру суперблока у 10^3 достатньо для досягнення максимальної швидкодії при гешуванні. Наведені дані представляють результати роботи реалізації "proof-of-concept", і у конкретних обчислювальних системах можливо вибрати більш оптимальні стратегії роботи з пам'яттю в залежності від способу подання даних для гешування, що забезпечить прискорення, близьке до теоретичних оцінок

Висновки

Удосконалено метод універсального гешування на основі обчислення значення полінома у скінченних полях, який відрізняється від прототипу тим, що передбачає гешування повідомлення у n паралельних потоків суперблоками з k блоків з деяким ключем x з наступним гешуванням отриманих проміжних геш-значень з ключем x^k, що дозволило збільшити швидкодію у n разів, де n -число потоків.

Розмір суперблока 10^3 блоків достатньо для досягнення максимальної швидкодії при паралельному гешуванні.

У конкретних обчислювальних системах можливо вибрати більш оптимальні стратегії роботи з пам'яттю в залежності від способу подання даних для гешування, що забезпечить прискорення, близьке до теоретичних оцінок.

Список літератури

1. *Damgård, A Design Principle for Hash Functions // In Proceedings of CRYPTO. – 1989, 416-427.*

2. *Sarkar P. A parallelizable design principle for cryptographic hash functions / P. Sarkar, P. Schellenberg. [Електронний ресурс]. – Режим доступу: www/ URL: http://eprint.iacr.org/2002/031.ps – Загол. з екрану.*

3. *Горбенко І.Д. Обґрунтування архітектури функції гешування з використанням паралельних обчислень / І.Д. Горбенко, А.О. Бойко // Праці науково-технічної конференції з міжнародною участю КМНТ-2010, ч. 1. – 2010. – С. 111-113.*

4. *Amdahl G. Validity of the single processor approach to achieving large scale computing capabilities / G. Amdahl [Електронний ресурс]. – Режим доступу : \WWW/ URL http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf – Загол.з екрану.*

Надійшла до редколегії 22.05.2012

Рецензент: д-р техн. наук, проф. І.Д. Горбенко, Харківський національний університет радіоелектроніки, Харків.

УНИВЕРСАЛЬНЫЕ ФУНКЦИИ ХЕШИРОВАНИЯ НА ОСНОВЕ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЯ ПОЛИНОМА В КОЛЬЦАХ ЦЕЛЫХ ЧИСЕЛ ПО МОДУЛЮ 2^n

А.А. Бойко

Предложено усовершенствование метода универсального хеширования на основе вычисления значения полинома в конечных полях путем использования параллельных вычислений. Усовершенствованный метод универсального хеширования позволяет увеличить быстродействие пропорционально количеству вычислительных ядер. Усовершенствованный метод универсального хеширования позволяет гибко настраивать параметры вычислений, сохраняя при этом совместимость между системами с различными параметрами. В работе приведен разработанный параллельный алгоритм хеширования и результаты экспериментальных измерений быстродействия и характеристик параллельных вычислений.

***Ключевые слова**: универсальное хеширование, параллельные вычисления.*

UNIVERSAL HASHING FUNCTIONS BASED ON POLYNOMIAL VALUE COMPUTING OVER RINGS OF INTEGERS BY MODULO 2^N

A.O. Boiko

The improvement of universal hashing technique basen on polynomial evaluation over finite fields by using parallel computing is proposed. Improved universal hashing technique is able to increase the speed of computing proportionally to number of CPU cores. Improved universal hashing technique computation parameters can be finely tuned preserving interoperability between systems with different parameters. This article describes developed parallel hashing algorithm. Results of experimental measurements of speed and parallel computation characteristics are also placed here.

Keywords: *universal hashing, parallel computing.*