УДК 004.75

А.М. Волокита, Ву Дик Тхінь, І.А. Чесніший, А.А. Коротенко, Г.В. Ісаченко

ПЛАНУВАЛЬНИК ЗІ СПЕЦІАЛІЗОВАНИМИ ЧЕРГАМИ ДЛЯ РОЗПОДІЛЕНОЇ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ

Анотація: розроблено програмне забезпечення планувальника в розподілених комп'ютерних системах з спеціалізованими чергами обслуговування для задач проблемної орієнтації, виконані експериментальні дослідження запропонованої розподіленої системи.

Ключевые слова: планувальник, спеціалізовані черги.

Вступ

Розподілення задач по обчислювальних ресурсах та створення спеціалізованих віртуальних каналів обслуговування для задач проблемної орієнтації є однією з проблем підвищення ефективності хмарних обчислень. Наприклад, відомо, що використання графічних процесорів GPU замість класичних CPU для певного класу задач дозволяє отримати значний приріст в продуктивності. Це пояснюється тим, що архітектура SIMD більшості сучасних графічних процесорів розрахована на задачі, які розпаралелюються на велику кількість вузлів. Прикладом таких задач є рендеринг, обробка зображень, блочне шифрування. Розроблені та широко використовуються мови програмування, які дозволяють створювати програми, здатні виконуватись як на графічних, так і на звичайних процесорах. Прикладом такої мови програмування є openCL, що забезпечує паралелізм на рівні інструкцій та рівні даних та є реалізацією техніки GPGPU - використання графічного процесору на відеокарті для проведення не графічних розрахунків. Використання подібних до openCL технологій для описання обчислювальних задач дозволить динамічно встановлювати цільовий пристрій для виконання обчислень.

Розроблене програмне забезпечення планувальника

Створена модель хмарних обчислень реалізовує клієнт-серверну архітектуру. Веб сервер розроблено за допомогою мови програмування Java та відкритої бібліотеки Netty. Netty використовує технологію не блокуючого вводу\виводу java NIO та широко використовується у побудові високонавантажених веб-серверів, в яких необхідно забезпечити швидку обробку пакетів від великої кількості користувачів (до ста тисяч одночасних користувачів). Протокол взаємодії між сервером та клієнтом створено за допомогою відкритої технології protocol buffers від Google.

CPU - Intel i3 2120; GPU - AMD RADEON 7770 1GHz

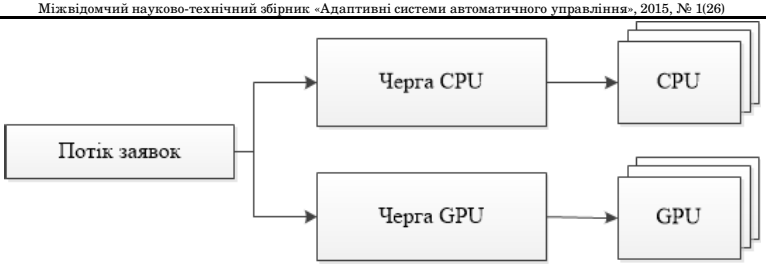

Рис. 1 – Структура спеціалізованих черг у гетерогенній системі

Для стабільної роботи центру хмарних обчислень слід забезпечити дві задачі: безперебійне функціонування веб-серверу, що відповідає на запити користувачів, та забезпечення роботи власне обчислювальних ресурсів. Для вирішення першої задачі пропонується виділити з ресурсів центру один чи декілька CPU виключно під потреби серверу, що дозволить навіть при максимальній навантаженості оперативно реагувати на запити користувачів: надавати інформацію про стан заявок, сервісні повідомлення про перенавантаження тощо.

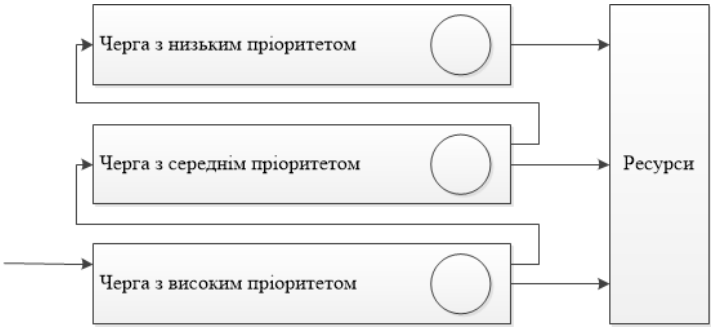

Рис. 2 – Структура багаторівневої черги із зворотними зв'язками

Друга задача - розподіл задач за типами по обчислювальних ресурсах. Маркування задач за типами дозволяє робити це максимально швидко. Передбачено наступні типи: CPU - задача, що може виконуватись виключно на CPU; GPU - задача, що може виконуватись виключно на GPU;

PREFFERED_GPU – задача, призначена для виконання на графічному прискорювачі, але така що може бути в разі завантаженості усіх GPU бути виконаною на звичайному процесорі. Аналогічно PREFFERED_CPU.

FOURIER, DECODING – спеціалізовані типи заявок, можуть виконуватись на спеціалізованих обчислювачах або на наявних процесорах. Кожен тип обчислювачів в системі (CPU, GPU, спеціалізо-

вані обчислювачі) має власну багаторівневу чергу. Кожна заявка має власний пріоритет, що визначається пріоритетом користувача і залежить від тарифного плану, або важливості чи терміновості певних обчислень.

Алгоритм багаторівневої черги із зворотними зв'язками

Кожна нова заявка розміщується в кінці черги найвищого рівня. Черга працює за принципом FIFO. Заявка, що знаходиться на початку черги вибирається та посилається на обробку на вільний пристрій. Якщо по закінченню виділеного кванту часу заявка виконана, вона вилучається з системи, користувачеві надсилається повідомлення про кінець обробки заявки. Якщо під час виконання заявка добровільно призупиняє власне виконання, вона додається в кінець поточного рівня черги. Якщо заявка використовує весь наданий квант часу, але при цьому не встигає виконатись, вона позбавляється процесорного часу та 1) додається в наступний рівень черги, якщо має низький пріоритет, 2) залишається в поточному рівні протягом ще декількох ітерацій, якщо має високий пріоритет. Процес продовжується поки в черзі не залишиться заявок. Також передбачено відказ від прийому на обробку нових заявок при досягнені певного критичного розміру черги.

Генератор заявок

Для тестування сервера на навантаження було розроблено спеціальний клієнт, який створював декілька з'єднань із сервером та генерував потік заявок розміщених у часі за розподілом Ерланга. Таким чином симулювалася одночасна робота кількох клієнтських робочих станцій.

Засоби захисту

Для забезпечення надійної та захищеної роботи й уникнення втрати важливої інформації використовуються наступні засоби захисту: AES шифрування вхідних задач і результатів та механізм обміну ключами за алгоритмом Диффі-Хелмана. Щоб забезпечити захищену передачу даних між клієнтом і сервером було використано шифрування із закритим ключем за стандартом AES-128. Клієнти мають можливість надсилати також дані у відкритому вигляді, при цьому відповідь також не підлягає шифруванню. Перед тим, як розпочати процедуру шифрованої передачі, клієнт та сервер мають узгодити ключ, що використовується для шифрування. Для цього одразу після з'єднання проводиться процедура рукостискання: клієнт та сервер передають один одному відкриті ключі, на основі яких створюється секретний (алгоритм Диффі-Хелмана) [1].

Алгоритм Диффі-Хелмана. Алгоритм Диффі-Хелмана використовує функцію дискретного піднесення до степеня і схожий на

метод Ель-Гамаля. Спочатку генеруються два великих простих числа n і q. Ці два числа не обов'язково зберігати в секреті. Далі один з партнерів Р1 генерує випадкове число x і посилає іншому учаснику майбутніх обмінів Р2 значення: $A = q^x \bmod n$. По отриманні A партнер Р2 генерує випадкове число y і посилає Р2 обчислене значення: $B = q^Y \bmod n$. Партнер Р1, отримавши B, обчислює $Kx = B^x \bmod n$, а партнер Р2 обчислює $Ky = A^y \bmod n$. Алгоритм гарантує, що числа Ky і Kx рівні і можуть бути використані в якості секретного ключа для шифрування. В даній реалізації використовуються числа n і q довжиною 1024 біт. Таким чином, відкритий ключ, що передається по мережі має довжину 1024 біт.

Алгоритм шифрування AES. Advanced Encryption Standard (AES), також відомий під назвою Rijndael — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт) [2].

Опис протоколу та пакетів

Поля message length та bytes передаються в зашифрованому вигляді у випадку використання опції шифрування. Оскільки протокол використовує обмежену кількість пакетів з різними ідентифікаторами, то вони винесені за межі блоків із шифрованими даними. Таким чином у зловмисника буде менше інформації про вміст зашифрованої частини пакета, а отже менше шансів на отримання секретного ключа та іншого вмісту пакета.

Таблиця 1. Опис пакетів.

ID	length	ciphered	message length	bytes
ідентифікатор пакета	довжина пакета, вказує скільки байт слід зчитати з каналу	байт, вказує, чи є повідомлення шифрованим	реальна довжина повідомлення	бінарні дані, отримані з повідомлення за допомогою Protocol buffers

Структура протоколу та програмна реалізація обробки пакетів розроблені з урахуванням можливостей для зручного та швидкого масштабування. Додавання обробка нових пакетів виконується наступним чином:

1) в файл-опис протоколу додається новий пакет (формат запису повідомлень та типи полів можна знайти на офіційному сайті Ptotobuf);

2) компілятор перетворює файл-опис у сирцеві коди мовою Java або C#;

3) у сирцевий код проекту додається новий клас-обгортка для пакету, що наслідує клас AbstractMessage та перевизначає метод handle.

Експерименти

Використання для обчислювальних задач процесорів, архітектура яких більше розрахована для конкретного завдання, дозволяє значно зменшити сумарний час обробки заявки. На рисунку 4 приведено приклад розв'язання однієї і тієї ж обчислювальної задачі лінійної алгебри - пошук точки перетину декількох ліній. Така задача часто виникає у фізичних симуляторах або при рендерингу зображень.

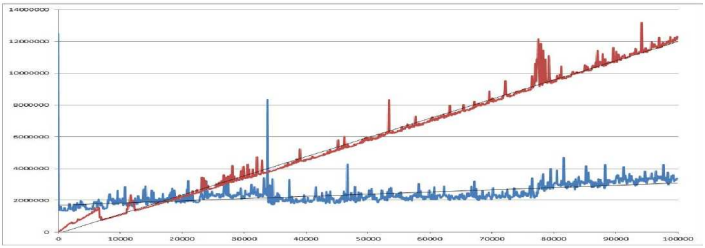

Рис. 3 – Час обчислення на CPU та GPU в залежності від кількості аргументів

Верхня крива (рис. 3) показує час виконання задачі на звичайному центральному процесорі, нижня - на графічному. Завантаження задачі на GPU завжди потребує певного часу, проте коли задача досягає розміру біля 18000 аргументів, час зрівнюється і в подальшому вже GPU безумовно перемагає в плані швидкодії.

Рис. 4 – Кількість відхилених задач в залежності від загальної кількості

Наведений графік (рис. 4) демонструє залежність кількості відхилених заявок від загальної кількості заявок на вході в систему.

Критерієм неможливості подальшого прийому заявок є досягнення певної критичної довжини черги. В системі передбачена фіксація факту надходження заявки під час перенавантаження та інформування користувачів про відновлення можливості обробки нових заявок.

Рис. 5 – Час відгуки системи від кількості заявок

Наступний графік (рис. 5) демонструє час реакції системи на заявки, що поступають. Регулювання максимального розміру черги для кожного типу пристроїв, а також виділення частини ресурсів спеціально під веб-сервер дозволяють досягти близького до сталого часу відклику без залежності від завантаженості системи. Час реакції наведено з врахуванням затримок доставки повідомлення в мережі.

Як видно з рисунку 6, що в разі розміщення системи в глобальній мережі затримки на доставку повідомлення значно перевищують час реакції системи, а більша частина часу реакції системи складається з обробки саме мережевого пакета.

Висновки

Розроблено програмне забезпечення планувальника з спеціалізованими чергами обслуговування для задач проблемної орієнтації, що дозволяє динамічно встановлювати цільовий пристрій для виконання обчислень. Виконані експериментальні дослідження запропонованої системи показали, що розподілення задач по обчислювальних ресурсах та створення спеціалізованих віртуальних каналів обслуговування при регулюванні максимального розміру черги для кожного типу пристроїв, а також виділення частини ре-

Рис. 6 – Час відгуки у урахування затримок мережі

сурсів спеціально під веб-сервер дозволяють досягти близького до сталого часу відклику без залежності від завантаженості системи.

Список використаних джерел

1. RFC 2631 – Diffie–Hellman Key Agreement Method E. Rescorla June 1999.
2. Federal Information Processing Standards Publication 197 November 26, 2001 Specification for the ADVANCED ENCRYPTION STANDARD (AES).

Отримано 17.04.2015 р.