УДК 681.301

К.т.н, доцент Петрашенко А.В., магістрант Буряк М.М.

**Національний технічний університет України
«Київський політехнічний інститут»**

АПАРАТНЕ ПРИСКОРЕННЯ АЛГОРИТМУ ШИНГЛІВ ДЛЯ ПОШУКУ НЕЧІТКИХ ДУБЛІКАТІВ ТЕКСТІВ

Abstract

Andrij V. Petrashenko, assoc. prof., PhD; Mykola Buryak, student
Hardware acceleration of shingle algorithm for searching textual near-duplicates
This paper concerns the idea for hardware acceleration of searching textual near-duplicates.Parallelization of shinglealgorithm operating allowed using the power of GPU-kernels. Significant increase of performance was achieved by using OpenCL technology.

Вступ

Головною рисою, що відрізняє сучасне інформаційне суспільство, є збільшення ролі інформації і знань в житті людей.

Постійне зростання обсягів інформації, що оброблюється засобами обчислювальної техніки, спричиняє,зокрема, поширення плагіату. Задача знаходження нечітких дублікатів текстових документів може бути вирішена за допомогою алгоритму шинглів. Проте, великі об’єми текстової інформації значно уповільнюють процес пошуку дублікатів і тому виникає необхідність прискорення роботи існуючих алгоритмів.

На сьогодні вже існують певні методи апаратного прискорення пошуку нечітких текстових дублікатів. Але вони базуються на збільшенні кількості процесорних модулів. Такий підхід дуже витратний з огляду на обчислювальні ресурси.

Постановка задачі

Метою даної роботи є прискорення існуючого алгоритму шинглів для пошуку нечітких дублікатів та його розпаралелення. Як результат, даний алгоритм можна буде апаратно прискорити за допомогою технології *OpenCL* [1].

Термінологія

Нечіткий дублікат текстового документу – текст, створений шляхом копіювання оригіналу та перестановки в ньому слів, речень, абзаців місцями.

API (від англ. *Application Programming Interface*) – інтерфейс програмування додатків.

OpenCL (від англ. *Open Computing Language*) – це *API* низького рівня для гетерогенних обчислень, який працює з архітектурою *CUDA*.

Алгоритм шинглів

Алгоритм шинглів дозволяє перевірити, чи є текст нечітким дублікатом. Реалізація алгоритму складається з 4-х етапів: канонізація текстів, розбиття текстів на шингли, знаходження значень контрольних функцій для отриманих шинглів та пошук однакових послідовностей.

Етап канонізації текстів полягає в видаленні з тексту всіх розділових знаків та частин мови, які не несуть змістового навантаження.

Розбиття на шингли відбувається шляхом поділу тексту на групи. Довжина шинглів може варіюватися.

Для кожного шинглу знаходиться контрольна сума. Для цього використовуються функції хешування. Знаходження хеш-функцій шинглів дозволяє збільшити швидкодію алгоритму: на етапі пошуку однакових послідовностей будуть порівнюватися значно менші об’єми даних.

Отримані хеші текстів порівнюються. По кількості збігів можна стверджувати про однаковість текстів.

OpenCL

OpenCL [1] дозволяє розробникам програмувати обчислювальні ядра за допомогою *C*-подібною мови та використовувати переваги паралельних обчислень *GPU* для створення обчислювальних додатків.

GPU-обчислення представлені спільним використанням *CPU* та *GPU* в гетерогенній моделі обчислень. Стандартна частина додатку виконується на *CPU,* а більш вимоглива до обчислень частина обробляється на *GPU* з прискоренням.

Використання технології *OpenCL* [1] дозволяє значно збільшити швидкодію програм. Архітектура *GPU* складається з декількох сотень процесорних ядер, паралельна робота яких забезпечує високу продуктивність обчислень над числами з плаваючою комою. Для ефективного використання *GPU* необхідно максимального розпаралелити

обчислення. Існуючий алгоритм шинглів можна адаптувати для виконання на *GPU*.

Розпаралелення алгоритму шинглів

Алгоритм шинглів дозволяє ефективно використовувати засоби паралельного обчислення. На рис. 1 представлено етапи роботи алгоритму та блоки, на яких будуть виконуватись обчислення.

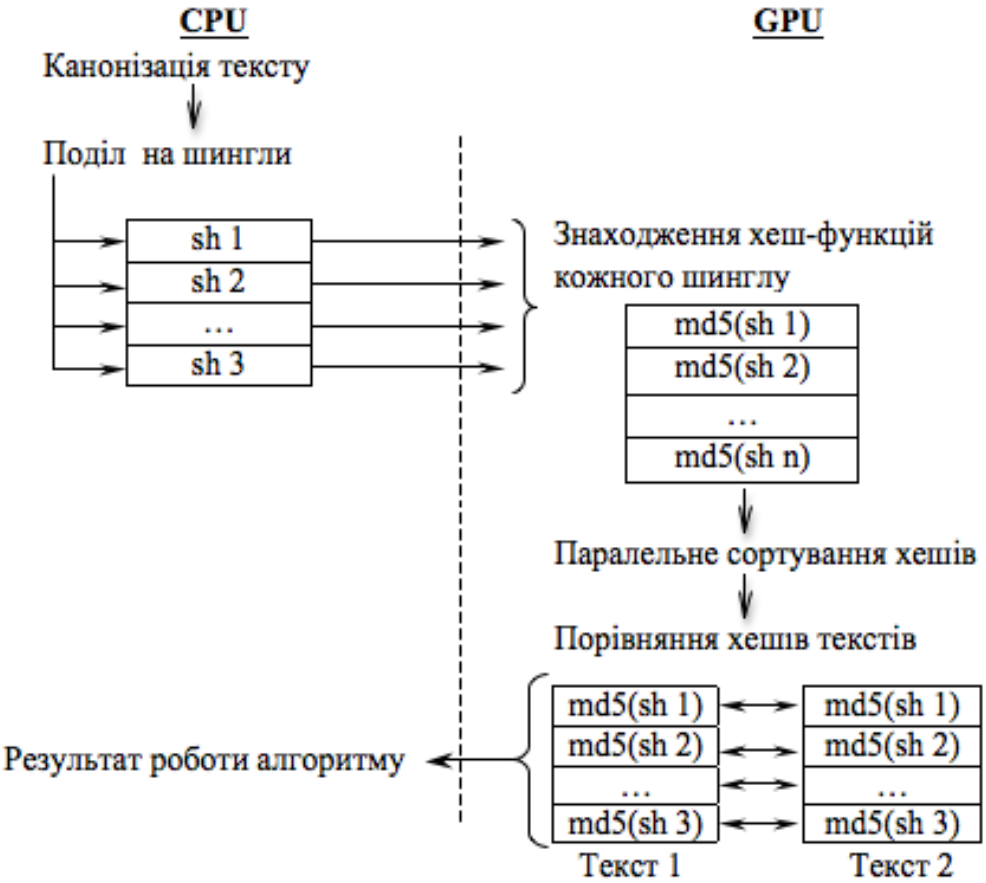

Рис.1. Етапи роботи алгоритму шинглів

Необхідність використовувати мову високого рівня для роботи з рядковими типами даних не дозволяє апаратно прискорити етапи канонізації та поділу на шингли. Апаратне прискорення цих етапів можливе лише шляхом додавання блоків *CPU*.

Після поділу тексту на шингли маємо масив даних, який передається на обробку до *GPU*. Хеш-функція для кожного шинглу обчислюється на окремому ядрі. Якщо *GPU* містить *N* ядер, то можна виконувати *N* потоків операцій паралельно.

Передача даних з оперативної пам’яті *CPU* до оперативної пам’яті *GPU* відбувається дуже повільно. Одним з шляхів прискорення алгоритму пошуку текстових дублікатів є зменшення об’ємів даних, які передаються.

Отриманий масив значень хеш-функцій необхідно відсортувати. Це дозволить в подальшому прискорити процес порівняння двох текстів. Для сортування використовується алгоритм *quicksort.* Він має максимальну швидкодію серед існуючих алгоритмів сортування. В основі алгоритму закладено поділ масиву на підмножини, кожну з яких можна обробляти незалежно на окремому ядрі *GPU.*

На завершальному етапі роботи алгоритму необхідно порівняти отримані масиви значень хеш-функцій для кожного тексту. Після попереднього етапу в буфері *GPU* знаходяться масиви значень хеш-функцій. Процес порівняння масивів має очевидне вирішення і ідеально підлягає розпаралелюванню.

Застосування отриманих результатів

Алгоритм шинглів підходить для пошуку нечітких дублікатів текстових документів. Він може бути застосований в пошукових системах для очищення результатів пошуку та для кластеризації документів за їх схожістю. Апаратне прискорення алгоритму шинглів дозволяє підвищити ефективність обробки текстових документів, кількість яких невпинно зростає.

Висновок

З ростом обсягів інформації у цифровому вигляді обробка текстових документів уповільнюється. Існуючий алгоритм шинглів дозволяє ефективно аналізувати оригінальність текстів, але він ресурсоємний. Пропонується апаратно прискорити роботу алгоритму шляхом використання технології *OpenCL* [1]. Процес роботи алгоритму був максимально розпаралелений. Стало можливим використовувати потужність ядер *GPU* для одночасного виконання великої кількості незалежних обчислень.

Література

1. OpenCL 1.1 Specification (revision 36, September 30, 2010). http://www.khronos.org/registry/cl/