Параллельная реализация алгоритма Simhash при помощи средств Cilk Plus

Шибаев Павел Павлович

Студент

Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия

E-mail: shiba.p@ya.ru

Научный руководитель *— Колосов Алексей Михайлович*

На данный момент существуют различные подходы к решению задачи поиска дубликатов или похожих записей в большом массиве данных (в бинарных файлах, логах, генетических данных, среди записей в базе данных). В 1997 А. Бродер [1] предложил алгоритм MinHash, который основан на идее вероятностной оценки похожести двух токенизированных текстов. Его основная идея заключается в использовании векторов значений нескольких хэш-функций для аппроксимации индекса похожести Жаккара, посчитанного над множествами токенов, соответствующих двум сравниваемым текстам. Этот алгоритм широко зарекомендовал себя на практике. Более того, он хорошо подвергается распараллеливанию [2].

В начале 2000-х в Google был разработан алгоритм Simhash [3], крайне эффективный как по памяти, так и по времени. Этот алгоритм нашел широкое практическое применение для решения задачи поиска дубликатов в случае веб-поиска. Алгоритм также основан на применении хэш-функций для вероятностной оценки похожести двух множеств. Однако, несмотря на простоту и эффективность этого алгоритма, не предпринималось попыток его параллельной реализации на CPU.

В настоящей работе приводятся результаты параллельной реализации Simhash с помощью инструментов Cilk++. Cilk++ (Cilk Plus) является эффективным языковым расширением C/C++ [4] и предоставляет широкий спектр средств для распараллеливания. В данной работе используется версия Cilk++, поставляемая как часть Intel Parallel Studio XE. Реализована простейшая последовательная версия алгоритма Simhash. В данной работе представляет интерес именно быстродействие алгоритма, поэтому настройка параметров хэш-функций не производится. С помощью инструкций `cilk_for` и `cilk_spawn` собраны различные параллельные конфигурации алгоритма. На основе статистики, полученной с помощью утилиты *cilkscreen*, произведен отбор наиболее оптимальной конфигурации [5]. Произведены пробные запуски оптимизированной програм-

мы при различном количестве ядер на четырехъядерном процессоре *Intel Core i5-10210U*. Важной характеристикой является ускорение - отношение времени работы программы в n-ядерном случае к времени работы при последовательном запуске, а также т.н. параллелизм - прогнозируемая верхняя граница ускорения в случае очень большого числа ядер. Практические запуски показали, что предложенная реализация имеет линейное ускорение в случае малого количества ядер. Верхняя граница ускорения при любом количестве ядер до 32-х также линейна. Согласно данным, полученным с помощью утилиты *cilkview*, верхняя граница ускорения 22.5 может быть достигнута при 32-х ядрах, т.е. далее с ростом количества ядер алгоритм ускорятьсяся не будет. Ниже на Рис.1 представлены данные анализа предложенной параллельной реализации, полученные с помощью утилиты *cilkplot*.

Иллюстрации

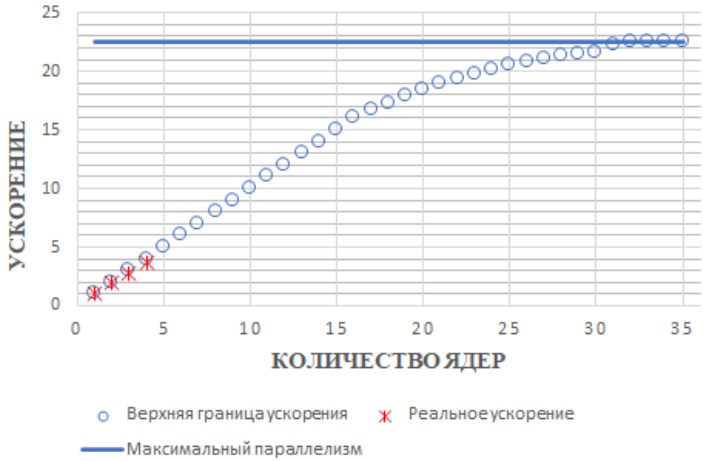

Рис.1 Зависимость ускорения от количества ядер при параллельной реализации Simhash

Литература

1. Broder A. Z. On the resemblance and containment of documents // In Proceedings. Compression and Complexity of Sequences 1997, Salerno, Italy, 1997, P. 21-29.

2. Cruz S., Kozawa Y., Amagasa T., Kitagawa H. GPU Acceleration of Set Similarity Joins. // In Database and Expert Systems Applications. Globe 2015, DEXA 2015. Lecture Notes in Computer Science, vol 9261. Springer, Cham
3. Charikar M. Similarity estimation techniques from rounding algorithms // In STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. Association for Computing Machinery, New York, United States, 2002, P. 380-388
4. Страница сообщества Cilk++:
`https://www.cilkplus.org/`
5. Репозиторий с кодом параллельной реализации Simhash:
`https://github.com/shibaeff/SimHashCilk`