3-бөлім **Раздел 3** **Section 3**

Информатика **Информатика** **Computer Science**

МРНТИ 50.05.13 DOI: https://doi.org/10.26577/JMMCS.2020.v107.i3.05

Б.С. Дарибаев[1,2*], **Д.В. Лебедев**[1], **Д.Ж. Ахмед-Заки**[2]
[1]Казахский национальный университет имени аль-Фараби, г.Алматы, Казахстан
[2]Университет международного бизнеса (UIB), г. Алматы, Казахстан
*e-mail: beimbet.daribayev@gmail.com

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ИЗВЛЕЧЕНИЯ N-GRAM ИЗ ТЕКСТА НА ФУНКЦИОНАЛЬНОМ ЯЗЫКЕ

В данной статье рассматривается реализация параллельного алгоритма извлечения N-gram из слабоструктурированного текста на функциональном языке системы LuNA реализующий технологию фрагментированного программирования. Алгоритм извлечения N-gram относится к задачам NLP. Проведен анализ других реализаций рассматриваемого параллельного алгоритма с использованием технологий MPJ Express, Apache Spark и Apache Hadoop. На основе анализа предлагается выбрать систему LuNA из-за того, что она умеет автоматически настраивать алгоритм на конкретную вычислительную систему за счёт используемой модели алгоритма в виде множества последовательных информационно зависимых задач, которые динамически распределяются по процессорам и ядрам вычислителя. В работе описывается схема реализации данного алгоритма, с применением технологии фрагментированного программирования. В статье была описана схема разделения на фрагменты данных и фрагменты вычислений. Приведена схема реализации алгоритма извлечения N-gram. Проведено тестирование на различном количестве процессоров для извлечения N-gram по словам. При извлечении токенов были удалены все стоп слова, которые задаются заранее в отдельном текстовом хранилище. Тестирование показало хорошую эффективность предлагаемого подхода по реализации алгоритмов с использованием системы LuNA.

Ключевые слова: параллельный алгоритм, функциональный язык, LuNA, N-gram, фрагментированное программирование.

Б.С. Дарибаев[1,2*], Д.В. Лебедев[1], Д.Ж. Ахмед-Заки[2]
[1]Әл-Фараби атындағы Қазақ Ұлттық университеті, Алматы қ., Қазақстан
[2]Халықаралық бизнес университеті (UIB), Алматы қ., Қазақстан
*e-mail: beimbet.daribayev@gmail.com

Функционалды тілде мәтіннен N-gram шығаруға арналған параллель алгоритмді жүзеге асыру

Берілген мақалада фрагменттелген программалау технологиясын қолданатын LuNA жүйесінің функционалдық тілінде әлсіз құрылымды мәтіннен N-gram-ды шығарып алу параллельді алгоритмінің жүзеге асырылуы қарастырылады. N-gram-ды шығарып алу NLP есептеріне жатады. MPJ Express, Apache Spark және Apache Hadoop технологияларын қолдану арқылы қарастырылып отырған параллель алгоритмнің басқа жүзеге асыруларына талдау келтірілген. Талдаудың негізінде LuNA жүйесін таңдау ұсынылады, себебі бұл жүйеде есептеуіштің процессорлары мен ядроларында динамикалық түрде үлестірілетін, ақпараттық байланысқан есептердің тізбектелген көпмүшесі түрінде қолданылатын алгоритмнің моделінің негізінде алгоритмді нақты есептеу жүйесіне автоматты баптай алу

мүмкіншілігіне ие бола алады. Жұмыста фрагменттелген программалау технологиясын қолдану арқылы берілген алгоритмның жүзеге асырылуының схемасы сипатталады. Мақалада мәліметтер фрагменттері мен есептеу фрагменттерінің бөліну схемасы сипатталған. N-gram-ды шығарып алу алгоритмінің жүзеге асырылу схемасы келтірілген. Сөздер бойынша N-gram-ды шығарып алуға процессорлардың әртүрлі мөлшерінде тестілеулер жүргізілген. Токендерді шығарып алу кезіне алдын-ала жеке мәтіндік қоймада берілетін барлық стоп сөздер жойылған. Тестілеудің нәтижесі LuNA жүйесін қолдану арқылы жүзеге асырылған алгоритмдерге қатысты ұсынылып отырған әдіс жақсы тиімділікті көрсетті.

Түйін сөздер: параллель алгоритм, функционалды тіл, LuNA, N-gram, фрагменттелген программалау.

B.S. Daribayev[1,2*], D.V. Lebedev[1], D.Zh. Akhmed-Zaki[2]
[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan
[2]University of International Business (UIB), Almaty, Kazakhstan
*e-mail: beimbet.daribayev@gmail.com

Implementation of A Parallel Algorithm to Extract N-gram from Text in a Functional Language

This paper discusses the implementation of a parallel algorithm for extracting N-grams from a semi-structured text in the functional language of the fragmented programming LuNA system. The N-gram extraction algorithm relates to NLP tasks. The analysis of other considered implementations of the parallel algorithm using MPJ Express, Apache Spark and Apache Hadoop technologies were carried out. Based on the analysis, it is proposed to choose the LuNA system due to the fact that it is able to automatically configure the algorithm for a specific computer system due to the algorithm model used in the form of a set of sequential information-dependent tasks that are dynamically distributed among the processor and processor cores. The paper describes the implementation scheme of this algorithm using fragmented programming technology. In this paper the scheme of division into data fragments and fragments of calculations is described. The implementation scheme of the N-gram extraction algorithm is presented. Testing was conducted on a different number of processors to extract N-gram by words. When extracting tokens, all stop words that were set in advance in a separate text storage were deleted. Testing showed good efficiency of the proposed approach for the implementation of algorithms using the LuNA system.

Key words: parallel algorithm, functional language, LuNA, N-gram, fragmented programming.

1 Введение

В настоящее время обработка больших текстовых информации используется в различных областях. Один из основных проблем такого рода задачи является высокопроизводительная обработка текста, который выделяет актуальность темы данного исследования. Многие ведущие ученые занимаются этой проблемой по сей день. Высокопроизводительная обработка больших текстовых информации является основной проблемой в области синтеза естественных языков (NLP). Одним из важных задач обработки NLP является извлечение N-gram из текста. Целью данного исследования является разработка высокопроизводительного алгоритма извлечения N-gram из текста. Основной задачей является реализация параллельного алгоритма извлечения N-gram из слабоструктурированного текста на функциональном языке системы LuNA [1,2]. LuNA – это система которая позволяет автоматически настраивать алгоритм на конкретную вычислительную систему за счёт используемой модели алгоритма в виде множества последовательных информационно зависимых задач, которые динамически распределяются на вычислительные ресурсы.

2 Обзор литературы

Такие гиганты как Google и Yandex используют в своих сервисах алгоритмы анализа, обработки и синтеза NLP [3]. Для обработки больших данных в NLP используются различные технологии машинного обучения [4,5]. Кроме того, используются различные высокопроизводительные технологии такие как CUDA [6, 7], Apache Spark [8], Apache Hadoop [9] и т.д. Также нами была рассмотрена работа [10], в котором авторы реализовали параллельные алгоритмы извлечения N-gram на технологиях MPJ Express [11,12], Apache Spark [13,14] и Apache Hadoop [15,16]. В результате тестирования были получены следующие выводы: MPJ Express – очень гибкий, высокая производительность, трудно реализовать параллельный алгоритм и отладить программный код; Apache Spark – по производительности не сильно уступает предыдущей, сравнительно чистый и короткий код, легко настраивается; Apache Hadoop – медленно работает на маленьких наборах данных, трудно настраивается. Учитывая эти недостатки, выбрали систему LuNA для реализации параллельного алгоритма задачи извлечения N-gram из текста.

3 Материал и методы

В системе LuNA используется модель вычислений, называемая фрагментированной программой (ФП). В этой модели данные задачи представляются как множество отдельных единиц, называемых фрагментами данных (ФД). ФД иммутабельны и являются переменными единственного присваивания. Значения ФД могут иметь как базовый тип (целочисленный, вещественный, и т.п.), так и составной (фрагмент сетки, вектор, и т.п.).

ФП задается множеством фрагментов вычислений (ФВ), каждый из которых связывается с набором входных и выходных ФД и вычисляет значения выходных ФД из значений входных. ФВ является процедурой без побочных эффектов.

Вычислительный процесс состоит в том, что ФВ, для которых известны значения всех их входных ФД и неизвестны значения выходных, исполняются, что приводит к вычислению новых ФД. Как следствие, новые ФВ могут быть исполнены, и т.д. Вычислительный процесс заканчивается, когда все ФВ не исполнены.

Для реализации фрагментированного алгоритма в системе LuNA мы создаем два ФВ (Рис.1):

1. Получения списка текстовых файлов из директорий (data) файловой системы;
2. Реализация алгоритма извлечение N-gram из полученных входных ФД (список текстовых файлов).

Фрагменты вычисления каждого процесса работают по готовности входных ФД. Реализация ФВ_GET_FILES – фрагмент вычисления метода получения списка доступных текстовых файлов.

В этом ФВ_GET_FILES мы получаем список текстовых файлов из директорий data. Список файлов является выходным ФД для данного ФВ. Количество текстовых файлов в каждом ФД вычисляется по следующей формуле:

D_SIZE = END - START;

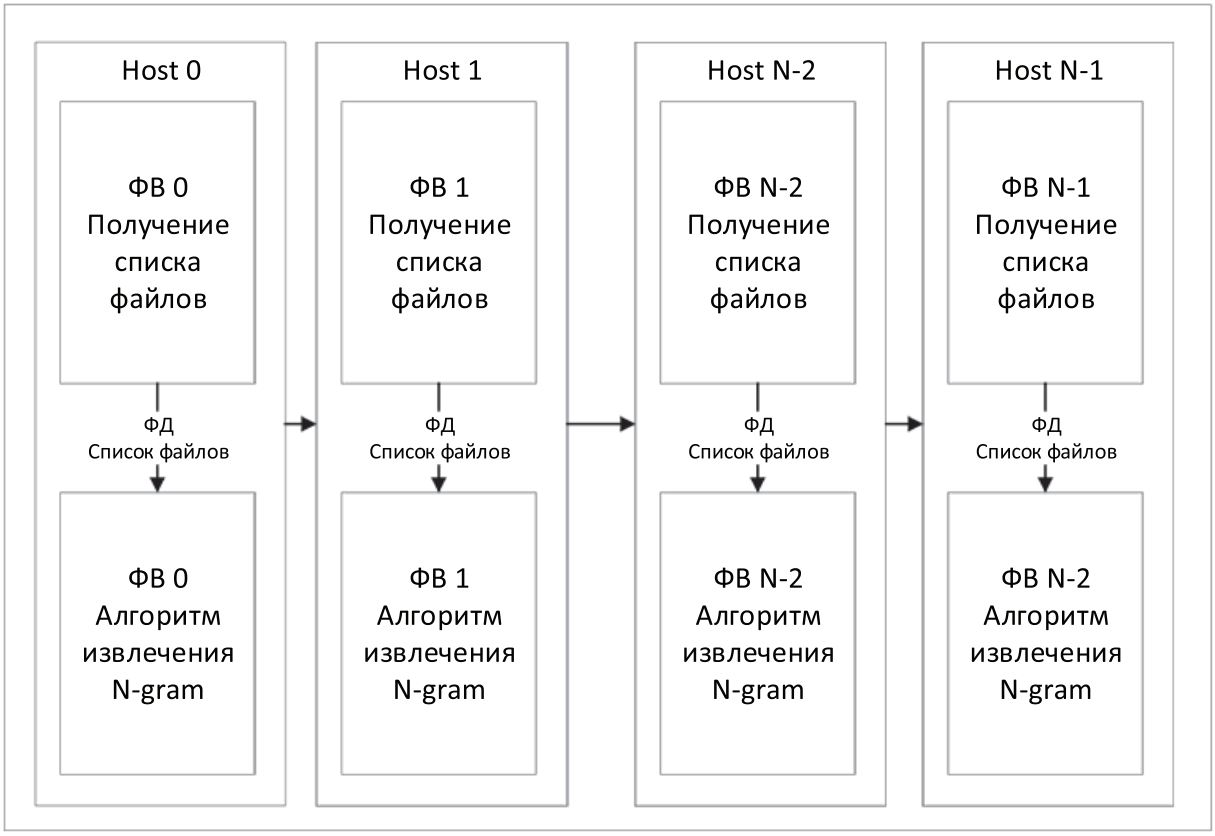


Рисунок 1 – Схема реализации алгоритма извлечения N-gram в системе LuNA

START = rank * N / FC + (rank < N % FC ? rank : N % FC);
END = rank < FC?((rank+1) * N / FC + ((rank+1) < N % FC?(rank+1):N % FC)):FC;

здесь, D_SIZE – количество текстовых файлов в каждом ФД, END – индекс начала списка текстовых файлов каждого ФД, START – индекс конца списка текстовых файлов каждого ФД, rank – ранг (индекс) ФВ, N – общее количество файлов в директорий data, FC – количество фрагментов.

После получения количества текстовых файлов, индексов начала и конца списка, готовим сам список файлов выходного ФД по следующему фрагменту кода на языке C++:

```
out.create(sizeof(string) * D_SIZE);
k = 0;
for(int i = START; i < END; i++)
{
    (out.getData<string>())[k] = inputDataFiles[i];
    k++;
}
```

здесь, out – выходной ФД, inputDataFiles – вектор из списка (имена с расширением)

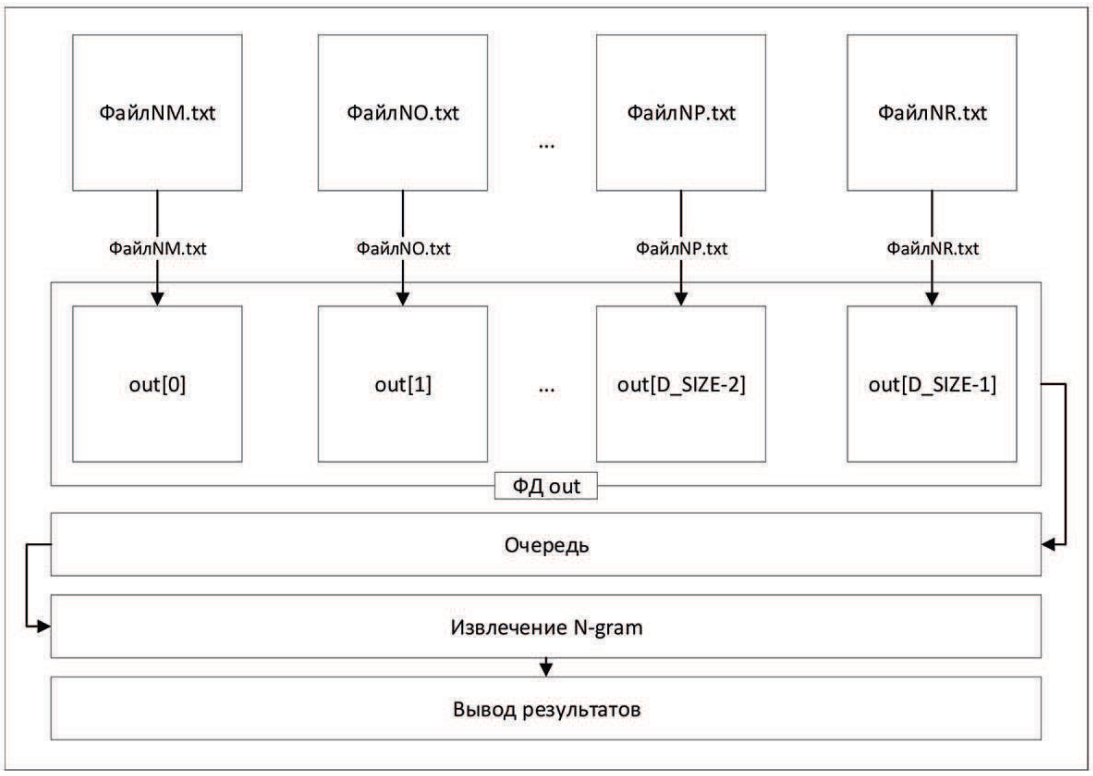


Рисунок 2 – Схема реализации алгоритма извлечение N-gram на ФВ

файлов.

Реализация ФВ_FIND_NGRAMS – алгоритм извлечение N-gram из полученных вектора списка текстовых файлов (Рис.2).

Выходные ФД в ФВ_GET_FILES будут для ФВ_FIND_NGRAMS входными ФД. ФВ_FIND_NGRAMS принимает четыре параметра:

1. ngramN – количество выводимых токенов в N-gram;
2. ngramType – тип извлечения N-gram (посимвольно, по словам и по байту);
3. inputDataFiles – список файлов для каждого входного ФД;
4. inputSize – размер списка файлов для каждого входного ФД.

От этих параметров зависит скорость вычисления программы извлечения N-gram из текста. Оптимальное значение ngramN выбирается между 2 и 5. То есть выбирать количество токенов больше пяти не имеет смысла, а единица означает что извлечения N-gram происходит посимвольно. Выбор значения ngramType зависит от требования задач. В текущем исследовании для тестирования результатов была выбрана извлечение N-gram по словам. При извлечении токенов в этом алгоритме удаляются все стоп слова [17], которые задаются заранее в отдельном текстовом хранилище. Значения inputDataFiles и inputSize зависят от количества текстовых файлов в директорий data которые вычисляются в ФВ_GET_FILES.

4 Результаты и обсуждение

Для тестирования результатов вычисления задачи извлечения N-gram из текста был использован ультрабук Lenovo Thinkpad X1 Carbon (6th Gen). В таблице 1 приведены технические характеристики ультрабука (Таблица1). Результаты тестирования напрямую зависят от характеристик процессора (частота процессора, количество физических ядер, объемы памяти кэшов L2 и L3) и объема памяти ОЗУ выбранного устройства. Тестирования проводились на 1, 2, 4 и 8 потоках процессора данного устройства.

Таблица 1 – Технические характеристики выбранного устройства для отладки и тестирования

№	Наименование	Характеристика
1	Модель	Lenovo Thinkpad X1 Carbon (6th Gen)
2	Процессор	Intel Core i7–8550U, 1800 МГц
3	Количество ядер	4 ядра
4	Объем кэша L2	1 МБ
5	Объем кэша L3	8 МБ
6	Операционная система	Ubuntu 18.04 LTS
7	Оперативная память	16 ГБ, LPDDR3, 2133 МГц
8	Встроенная память	1024 ГБ, PCIe SSD

При тестировании алгоритма количество входных текстовых файлов на разных потоках менялось пропорционально. По росту количества потоков время вычисления уменьшалась, только на 8 потоках по сравнению с 4 потоками время вычисления медленнее (Рис.3). Это можно объяснить тем, что в характеристиках процессора (Таблица1) только 4 физических ядра и 4 логических процессоров.

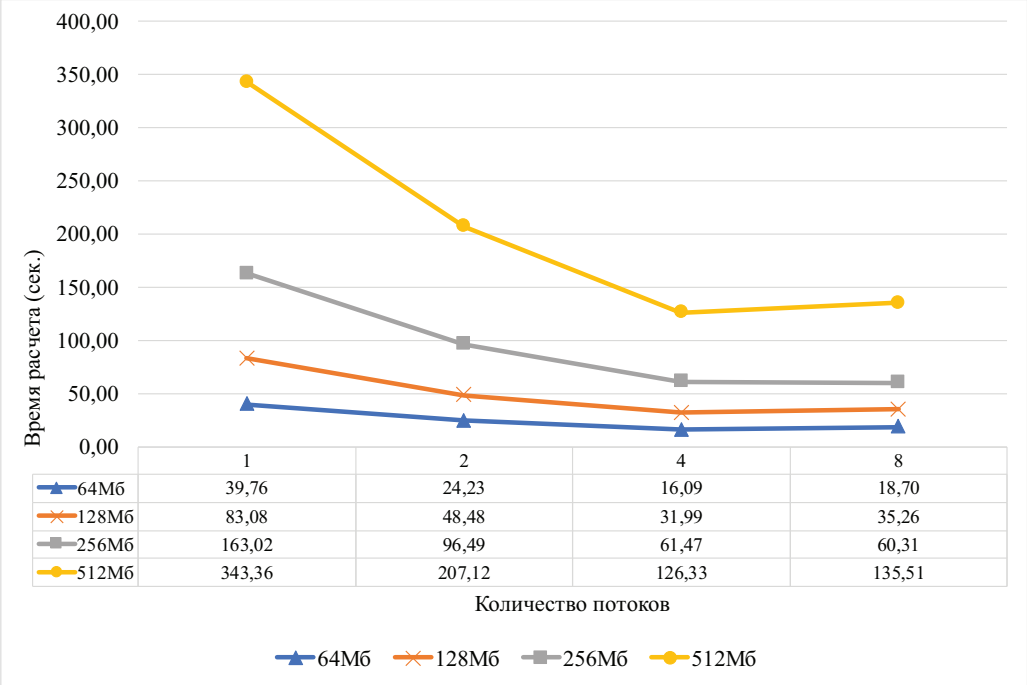
Ускорение вычислительного алгоритма на системе LuNA на 8 (восьми) потоках резко уменьшается, опять это видно, что на это влияет технические характеристики выбранного нами процессора. На Рис.4 показано ускорение вычислительных алгоритмов.

Следующий рисунок показывает эффективность вычислительного алгоритмов:

5 Заключение

В результате исследования был реализован алгоритм извлечения N-gram из текста на функциональном языке системы LuNA. В результате тестирования ускорение и эффективность вычислительных алгоритмов на 8 потоках показали худшие результаты из-за недостатков физических ядер процессора на тестируемым устройстве. В целом алгоритм хорошо работает на системе LuNA, так как отсутствуют пересылки данных между процессами.

В дальнейшем планируется исследовать характеристики алгоритма для определения оптимального варианта запуска.



	1	2	4	8
64Мб	39,76	24,23	16,09	18,70
128Мб	83,08	48,48	31,99	35,26
256Мб	163,02	96,49	61,47	60,31
512Мб	343,36	207,12	126,33	135,51

Рисунок 3 – Время вычисления

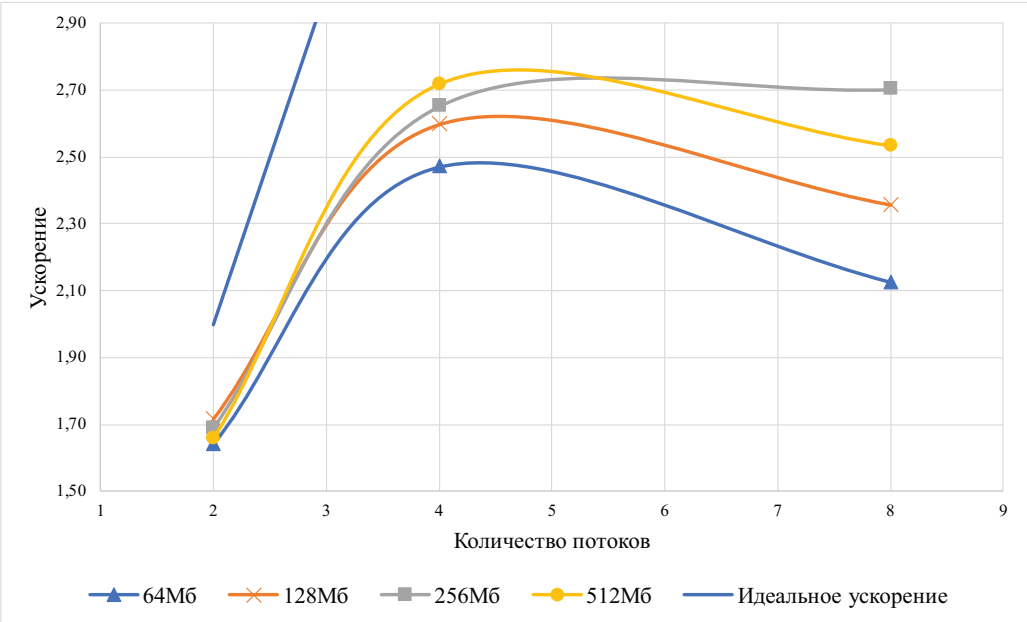


Рисунок 4 – Ускорение вычислительных алгоритмов

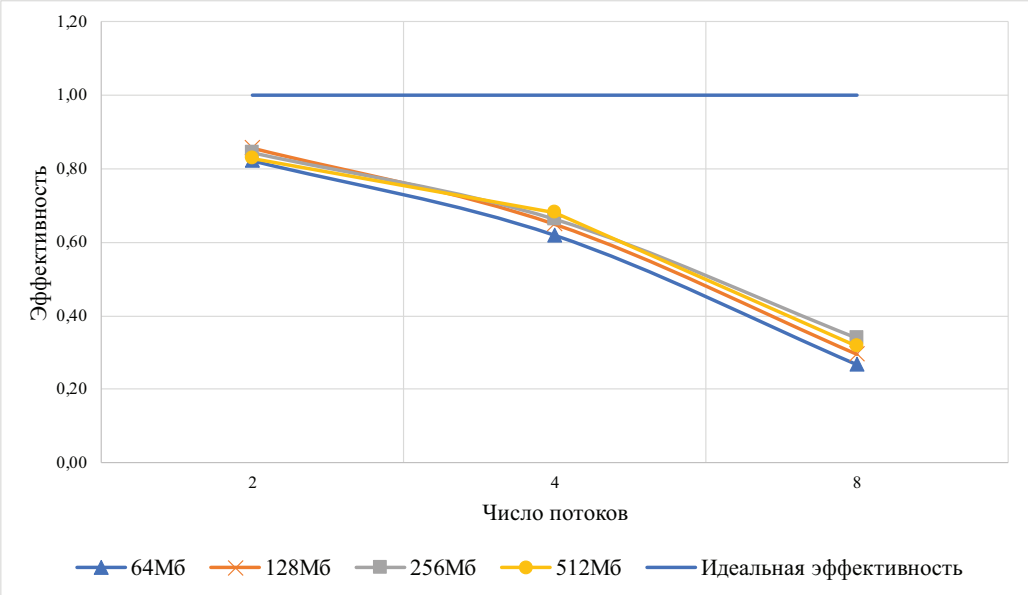


Рисунок 5 – Эффективность

6 Благодарности

Работа выполнена при поддержке грантового финансирования научно-технических программ и проектов Министерством образования и науки Республики Казахстан (грант AP05134651 «Разработка системы управления активными знаниями для автоматизации конструирования высокопроизводительных параллельных программ обработки неструктурированных данных и численного моделирования в задачах фильтрации», 2018-2020 годы).

Список литературы

[1] Малышкин В.Э. Технология фрагментированого программирования // Вестник ЮУрГУ. Сер. Выч. матем. информ. – 2012. - № 1. – С. 45–55.

[2] Malyshkin V., Perepelkin V., Schukin G. Scalable distributed data allocation in LuNA fragmented programming system // Journal of Supercomputing. – Vol.73, N 2. – P. 726-732.

[3] Google Natural Language - https://cloud.google.com/natural-language/

[4] Brants T., Popat A.C., Xu P., Och F.J., Dean J. Large Language Models in Machine Translation // Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. – 2007. – P. 858-867.

[5] Young T., Hazarika D., Poria S., Cambria E. Recent Trends in Deep Learning Based Natural Language Processing // IEEE Computational Intelligence Magazine. – 2018. – Vol.13, N 3. – P. 55-75.

[6] Srinivasa K.G., Shree Devi B.N. GPU Based N-Gram String Matching Algorithm with Score Table Approach for String Searching in Many Documents // Journal of The Institution of Engineers (India): Series B. - 2017, - Vol.98, No. 5, - P. 467-476.

[7] Banasiak D. Statistical methods of natural language processing on GPU // Advances in Intelligent Systems and Computing. - 2016. - P. 595-604.

[8] Shaikh E., Mohiuddin I., Alufaisan Y., Nahvi I. Apache Spark: A Big Data Processing Engine // 2019 2nd IEEE Middle East and North Africa COMMunications Conference. – 2019.

[9] Bougar M., Ziyati E.H. Addressing Stemming Algorithm for Arabic Text Using Spark Over Hadoop // Advances in Intelligent Systems and Computing. – 2020. – P. 74-82.

[10] Aubakirov S., Trigo P., Ahmed-Zaki D. Comparison of distributed computing approaches to complexity of N-gram extraction // Proceedings of the 5th International Conference on Data Management Technologies and Applications - Volume 1: DATA. – 2016. – P. 25-30.

[11] Carpenter B., Getov V., Judd G., Skjellum A., Fox G. MPJ: MPI-like message passing for Java // Concurrency: Practice and Experience. – 2000. – Vol.12, N 11. – P. 1019-1038.

[12] Baker M., Carpenter B., Shafi A. MPJ Express: towards thread safe Java HPC // IEEE International Conference on Cluster Computing. – 2006.

[13] Meena B., Sarwani I.S.L., Archana M., Supriya P. Comparative Analysis of Apache Spark and Hadoop MapReduce Using Various Parameters and Execution Time // Advances in Intelligent Systems and Computing. – 2020. – P. 719-725.

[14] Sharmila K., Kamalakannan T. Analytics for healthcare using Hadoop MapReduce, Apache Spark and in cloud services // International Journal of Scientific and Technology Research. – 2020. – Vol.9, N 1. – P. 706-710.

[15] Lydia E.L., Satyanarayan S., Kumar K.V., Ramya D. Indexing documents with reliable indexing techniques using Apache Lucene in Hadoop // International Journal of Intelligent Enterprise. – 2020. – Vol.7, N 1(3). – P. 203-214.

[16] Pineiro C., Martinez-Castano R., Pichel J.C. Ignis: An efficient and scalable multi-language Big Data framework // Future Generation Computer Systems. – 2020. – P. 705-716.

[17] Baradad V.P., Mugabushaka A.-M. Corpus Specific Stop Words to Improve the Textual Analysis in Scientometrics // Proceedings of 15th International Society of Scientometrics and Informetrics Conference. – Istanbul, 2015. – P. 999-1005.

References

[1] Malyshkin V.E., "Tehnologija fragmentirovanogo programmirovanija" , *Vestnik JuUrGU. Ser. Vych. matem. inform* no. 1 (2012): 45–55.

[2] Malyshkin V., Perepelkin V., Schukin G., "Scalable distributed data allocation in LuNA fragmented programming system" , *Journal of Supercomputing* 73, no. 2 (2016): 726-732.

[3] "Google Natural Language" , https://cloud.google.com/natural-language.

[4] Brants T., Popat A.C., Xu P., Och F.J., Dean J., "Large Language Models in Machine Translation" , *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (2007): 858-867.

[5] Young T., Hazarika D., Poria S., Cambria E., "Recent Trends in Deep Learning Based Natural Language Processing" , *IEEE Computational Intelligence Magazine* 13, no. 3 (2018): 55-75.

[6] Srinivasa K.G., Shree Devi B.N."GPU Based N-Gram String Matching Algorithm with Score Table Approach for String Searching in Many Documents" , *Journal of The Institution of Engineers (India): Series B* 98, No.5 (2017): 467-476.

[7] Banasiak D. "Statistical methods of natural language processing on GPU" , *Advances in Intelligent Systems and Computing* (2016): 595-604.

[8] Shaikh E., Mohiuddin I., Alufaisan Y., Nahvi I., "Apache Spark: A Big Data Processing Engine" , *2019 2nd IEEE Middle East and North Africa COMMunications Conference* (2019).

[9] Bougar M., Ziyati E.H., "Addressing Stemming Algorithm for Arabic Text Using Spark Over Hadoop" , *Advances in Intelligent Systems and Computing* (2020): 74-82.

[10] Aubakirov S., Trigo P., Ahmed-Zaki D., "Comparison of distributed computing approaches to complexity of N-gram extraction" , *Proceedings of the 5th International Conference on Data Management Technologies and Applications - Volume 1: DATA* (2016): 25-30.

[11] Carpenter B., Getov V., Judd G., Skjellum A., Fox G., "MPJ: MPI-like message passing for Java" , *Concurrency: Practice and Experience* 12, no. 11 (2000): 1019-1038.

[12] Baker M., Carpenter B., Shafi A., "MPJ Express: towards thread safe Java HPC" , *EEE International Conference on Cluster Computing* (2006).

[13] Meena B., Sarwani I.S.L., Archana M., Supriya P., "Comparative Analysis of Apache Spark and Hadoop MapReduce Using Various Parameters and Execution Time" , *Advances in Intelligent Systems and Computing* (2020): 719-725.

[14] Sharmila K., Kamalakannan T., "Analytics for healthcare using Hadoop MapReduce, Apache Spark and in cloud services" , *International Journal of Scientific and Technology Research* 9, no. 1 (2020): 706-710.

[15] Lydia E.L., Satyanarayan S., Kumar K.V., Ramya D., "Indexing documents with reliable indexing techniques using Apache Lucene in Hadoop" , *International Journal of Intelligent Enterprise* 7, no. 1 (2020): 203-214.

[16] Pineiro C., Martinez-Castano R., Pichel J.C., "Ignis: An efficient and scalable multi-language Big Data framework" , *Future Generation Computer Systems* (2020): 705-716.

[17] Baradad V.P., Mugabushaka A.-M., "Corpus Specific Stop Words to Improve the Textual Analysis in Scientometrics" , *Proceedings of 15th International Society of Scientometrics and Informetrics Conference* (2015): 999-1005.