УДК [611.018.51+615.47]:612.086.2

УСКОРЕНИЕ ОБЪЕДИНЕНИЯ РАСПРЕДЕЛЕННЫХ НАБОРОВ ДАННЫХ ПО ЗАДАННОМУ КРИТЕРИЮ

Е.С. Тырышкина
Аспирантка НИУ ВШЭ,
разработчик хранилища данных VK

Московский институт электроники и математики
Национального Исследовательского Университета «Высшая Школа Экономики»
ООО «ВК», Российская Федерация
E-mail: tyryshkina_evgeniya@mail.ru

Е.С. Тырышкина

Родилась в Алматы, Казахстан, в 1994 году. В 2016 году получила степень бакалавра техники и технологий в области информационных систем в Казахском национальном университете им. аль-Фараби в Алматы и степень магистра в 2018 году в Московском институте электроники и математики Национального Исследовательского Университета «Высшая Школа Экономики» (МИЭМ НИУ ВШЭ) по специальности «Вычислительная техника». Магистерскую диссертацию писала на тему радиационно-индуцированной проводимости в космических диэлектрических материалах. В настоящее время является аспиранткой в МИЭМ НИУ ВШЭ в области вычислительной техники.

Аннотация. В данной работе исследуется вопрос снижения затрат машинного времени за счет разработки и внедрения метода ускорения операции соединения распределенных массивов данных по заданному критерию. Был проведен обзор литературы по архитектуре распределенных хранилищ данных и алгоритмам параллельных вычислений в результате которого выделены лимитирующие стадии, замедляющие процесс выполнения операции соединения, которые были исключены в предлагаемом в данной работе методе, на основе которого создан алгоритм и реализована библиотека, расширяющая функционал коммерческого программного продукта. Для оценки результата проведены экспериментальные исследования. Работа данного метода сравнивалась со стандартной библиотекой Spark SQL и показала сокращение времени на ~ 37% для данных размером 2 ТБ и ~ 47% для данных 7 ТБ.

Ключевые слова: MapReduce, распределенные вычислительные системы, Apache Spark, хранилища данных, аналитика.

Введение.

Разработанный метод реализован на базе Hadoop Distributed File System (HDFS) [1]. HDFS – это файловая система, предназначенная для хранения файлов больших размеров поблочно расположенных по узлам вычислительного кластера с возможностью потокового доступа к информации. В данной файловой системе можно не только хранить данные, но и выполнять над ними некоторые вычисления. Для обработки информации в файловой системе Hadoop была разработана вычислительная модель MapReduce [2, 3], получившая название по 2м этапам данной концепции: Map - чтение и преобразование и Reduce – агрегация и запись.

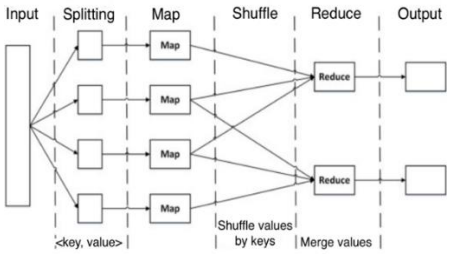

Рисунок 1 – Вычислительная модель MapReduce

Предлагаемый мной алгоритм был разработан во фреймворке Apache Spark, в основе которого также лежит концепция MapReduce. Особенность вычислительного движка Apache Spark в том, что исключаются многократные операции чтения и записи на диск за счёт более эффективному использованию оперативной памяти. Выбор данного фреймворка был обусловлен рядом преимуществ, которыми он обладает: предоставляет удобный интерфейс разработчика и высокоуровневые инструменты для создания собственных расширений и утилит; распространяется под свободной лицензией с открытым исходным кодом; большое сообщество разработчиков; высокая эффективность; подтвержденная независимыми исследованиями разных компаний [4].

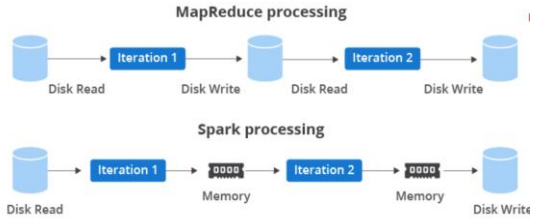

Рисунок 2 – Сравнение вычислительной модели MapReduce и Apache Spark

Задача соединения данных, которая исследовалась в данном исследовании, представляет собой одну из важных и тяжелых вычислительных задач анализа данных. Множество исследований ведется в направлении разработки методов выполнения операции соединения с улучшением производительности и сокращением скорости выполнения. В основу фреймворка спарк заложено 3 базовых подхода: hash join, sort-merge join, broadcast join [5]. Выбор метода делается на основе характеристик входных данных. Так, например, hash join, sort-merge join используется, когда оба набора данных большого объема, в то время как для применения broadcast join необходимо, чтобы один набор данных был достаточно маленьким, чтобы поместиться в оперативную память.

Таким образом, broadcast join ограничен объемом входных данных, sort-merge join – наличием этапа сортировки, sort-merge join и hash join – наличием этапа перемещения, реорганизации данных в файловой системе перед выполнением операции соединения. Следовательно, для решения моей задачи мне необходимо было исключить лимитирующие стадии данных алгоритмов. То есть исключить этап перемещения данных, сортировки и снять ограничения по объему наборов данных для соединения.

Особенности предлагаемого метода.

Предлагаемый метод работает в два этапа. На первом этапе, производится реорганизация хранения данных первого датасета (RDD1). Производится это паралелльным чтением строк датасета и записью в набор файлов. Попадание определенной записи в файл выбирается на освное критерия соединения. Данная операция выполняется без промежуточного этапа перемещения данных между стадиями Map и Reduce. На втором

шаге выполняется репартиционирование на логическом уровне также на основе заданного критерия объединения без физического перемещения данных по кластеру. Далее для каждой партиции данных паралелельно выполняется чтение и загрузка в память файлов первого набора RDD1, где уже локально производится операция сравнения ключей строк, последующее соединение и запись результата. Важно не допустить перекоса данных при партиционировании [6].

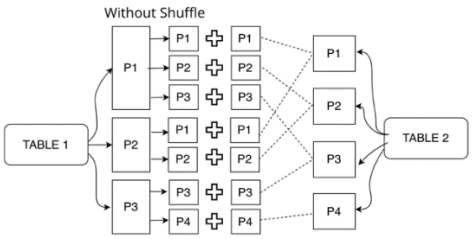

Рисунок 3 – Схема предлагаемого метода

В данном методе партиции P1, P2, P3 таблицы 1, партиционируются по критерию объединения только на логическом уровне, без физического перемещения данных. Таблица 2 записывается на диски пофайлово, также с партиционированием по ключу объединения. Далее файлы таблицы 2 загружаются в память в структуру HashMap на те хосты кластера, которые хранят соответствующие партиции таблицы 1, как это делается при броадкаст соединении. Партиции таблицы 1 читаются построчно и для каждой записи производится поиск записей из HashMap файлов таблицы 2. В результате получаем левостороннее объединение массивов данных.

Результаты экспериментов.

Экспериментальные расчеты проводились на текстовых данных, сжатых с помощью алгоритма GZIP. Для проведения расчетов были задействованы 128 исполнителей с 6 ядрами и 16Гб памяти для программ-исполнителей и 8Гб памяти для программ-драйвера. Было проведено несколько тестов нового метода объединения и проведено его сравнение со стандартным sort-merge объединением, используемым в Spark SQL.

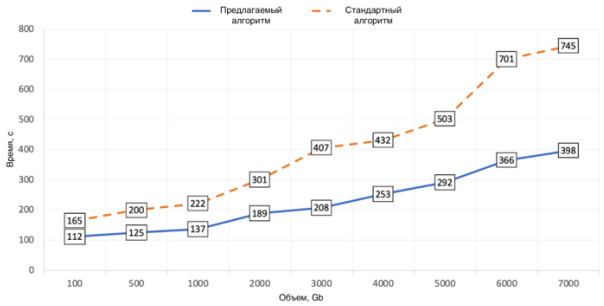

Рисунок 4 – Сравнение времени выполнения расчета относительно количества входных данных для алгоритма, предложенного в работе (сплошная линия) и в стандартном Spark SQL (пунктирная линия)

Заключение.

Таким образом, в данной работе было выполнено:

• исследование литературы по распределенным файловым системам, вычислительным программным продуктам для работы в таких системах и методам операции соединения данных;

• на основе данных исследований были установлены лимитирующие стадии,

замедляющие процесс обработки;

• разработан метод, исключающий выбранные лимитирующие стадии;

• на основе метода создан алгоритм и утилита, расширяющая функционал выбранного ПО;

• выполнены экспериментальные исследования.

Таким образом, по сравнению со стандартной библиотекой Spark SQL получено сокращение затрат машинного времени на ~ 37% для данных размером 2 ТБ и ~ 47% для данных 7 ТБ.

Многие задачи еще предстоит решить и реализовать с помощью этого алгоритма. Например, сейчас рассмотрен только один тип операции соединения — левое (правое) соединение. В дальнейшем необходимо реализовать внутреннее соединение. Также не решается проблема перекоса данных, необходимо реализовать автоматическое определение перекоса. Тем не менее, этот алгоритм может быть полезен во многих вычислительных задачах, так как он достаточно стабилен и показывает высокую скорость выполнения.

Список литературы

[1] S. Ghemawat, H. Gobioff, , and S.-T. Leung. The google file system. [In Proc. of the “19th ACM Symposium on Operating Systems Principles”]. Болтон Лендинг, Нью-Йорк, США, 2003, pp. 29–43.

[2] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Communications of the ACM, 2008, vol. 51, no. 1, pp. 107–113. doi:10.1145/1327452.1327492.

[3] K. Shim. MapReduce algorithms for big data analysis. VLDB Endowment, 2012, vol. 5, no. 12, pp. 2016–2017. doi:10.14778/2367502.2367563.

[4] J. Dittrich, J. Quian´e-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). PVLDB, 2010, vol. 3, no. 1-2, pp. 515-529. doi: 10.14778/1920841.1920908.

[5] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark SQL: Relational data processing in spark. [In Proc. of the “2015 ACM SIGMOD International Conference on Management of Data (SIGMOD’15)”]. Мельбурн, Виктория, Австралия, 2015, pp. 1383–1394.

[6] Y. C. Kwon, M. Balazinska, B. Howe, and J. Rolia. A study of skew in mapreduce applications. [In Proc. of “Open Cirrus Summit”]. Москва, 2011.

ACCELERATE THE JOINING OF DISTRIBUTED DATASETS BY A GIVEN CRITERIA

Y.TYRYSHKINA
Postgraduate student of the HSE University, data engineer in VK

Moscow Institute of Electronics and Mathematics
National Research University Higher School of Economics
VK, Russian Federation
E-mail: tyryshkina_evgeniya@mail.ru

Abstract. In this paper, we study the issue of reducing the cost of computer time by developing and implementing a method for accelerating the operation of joining distributed datasets according to a given criterion. A review of the literature on the architecture of distributed data storages and parallel computing algorithms was carried out, as a result of which limiting stages were identified that slow down the process of performing a joining operation, which were excluded in the method proposed in this paper, on the basis of which an algorithm was created and a library was implemented that expands the functionality of a commercial software product. Experimental studies were carried out to evaluate the result. This method was compared with the Spark SQL standard library and showed ~37% time savings for 2TB data and ~47% for 7TB data.

Keywords: MapReduce, distributed computing systems, Apache Spark, data warehouses, analytics.