УДК 519.6

М. В. Гладкий
Белорусский государственный технологический университет

МОДЕЛЬ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ MAPREDUCE

Предмет исследования данной статьи – алгоритмы и методы эффективной обработки и анализа данных большого объема, основанных на использовании парадигмы распределенных вычислений в компьютерных кластерах MapReduce.

Рассмотрена общая схема работы, область применения и принципы параллельной реализации MapReduce вычислений на кластерных системах. Выявлены основные стадии обработки информации в исследуемой модели: Map (предварительная обработка данных), Combine (частичная агрегация данных по ключу), Shuffle (разделение данных на секции и сортировка промежуточных пар по ключу), Reduce (свертка обработанных данных).

Разработана классификация алгоритмов и методов решения задач, использующих описанную модель распределенных вычислений. В качестве классификаторов выступают стадии обработки MapReduce, а также возможные сценарии прикладного использования решаемых задач. Данная классификация позволила разбить совокупность решаемых задач на четыре категории: MapReduce (задачи, для решения которых применяются функции Map и Reduce), MapOnly (задачи, для решения которых можно обойтись только стадией Map), цепочки MapReduce (задачи, для решения которых необходимо последовательно использовать несколько моделей MapReduce) и ReduceJoin (задачи, в которых необходимо объединять содержимое нескольких документов по ключу). Для каждой из групп приводятся различные примеры и сценарии.

Ключевые слова: распределенные вычисления, компьютерный кластер, MapReduce, Combine, Shuffle.

M. V. Gladkiy
Belarusian State Technological University

THE MODEL OF DISTRIBUTED COMPUTING MAPREDUCE

The subject of the research in this article is algorithms and methods of effective processing and analysis data. It is based on the paradigm of distributed computing in computer clusters MapReduce.

The General scheme of work, scope and principles of parallel implementations of the MapReduce computing on cluster systems have been considered. The main stages of the information processing identified in the study model: Map (data preprocessing), Combine (partial data aggregation key), Shuffle (time-dividing the data into sections and sorting the intermediate pairs by key), Reduce (compress the processed data).

Algorithms classification and methods for task solution, which use the model of distributed computing has been developed. The stages of MapReduce processing act as classifiers, and possible scenarios of application of tasks. This classification allowed us to divide the aggregate a set of tasks into four categories: MapReduce (tasks for which apply Map and Reduce functions), MapOnly (for which you can get by with just a stage Map), chaining MapReduce (tasks, for the solution of which is necessary to consistently apply multiple models MapReduce) and ReduceJoin (tasks that you want to merge the contents of multiple documents by key). Various examples and scenarios are given for each group.

Key words: distributed computing, cluster computing, MapReduce, Combine, Shuffle.

Введение. В современном мире все большую роль играют технологии, обеспечивающие эффективную обработку больших массивов данных. Мировой объем информации увеличивается более чем в 2 раза каждые два года, что говорит о лавинообразном росте общего количества данных. Современные программные средства предъявляют серьезные требования к вычислительным ресурсам, значительно превышающие возможности отдельных компьютеров. Особое значение уделяется алгоритмам и методам, применяемым для обработки и анализа данных с использованием компьютерных кластеров, состоящих из сотен или тысяч узлов. Однако реализация процедур обработки данных на кластерных системах сопряжена с решением таких задач, как разбиение и распределение данных между процессорами, балансировка нагрузки, обработка отказов, сбор и агрегация промежуточных результатов [1].

Кроме того, на данном этапе развития средств распараллеливания вычислений отсутствует детальное описание алгоритмов, методов и подходов распределенной обработки данных, что связано с бизнес-моделью многих компаний. Данная область исследования требует

детального изучения: необходимо рассмотреть и исследовать парадигму эффективной обработки и анализа данных большого объема MapReduce; классифицировать совокупность решаемых задач в исследуемой области по различным критериям.

Основная часть. Одним из самых эффективных методов обработки больших объемов данных в распределенных средах является парадигма MapReduce, предложенная компанией Google в начале 2000-х для сканирования и обработки большого количества страниц из сети Интернет. Впервые такая парадигма была реализована в составе распределенной файловой системы GFS (Google File System) и в высокопроизводительной нереляционной базе данных Big Table [1].

Данная модель отличается простотой и удобством использования, скрывает от пользователя детали организации вычислений на кластерной системе. Преимущество MapReduce заключается в том, что она позволяет распределенно выполнять операции предварительной обработки и свертки. Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно. Аналогичным образом множество рабочих узлов осуществляют операцию свертки – для этого необходимо, чтобы все результаты предварительной обработки с одним конкретным значением ключа обрабатывались одним рабочим узлом в один момент времени [2].

Параллелизм также дает некоторые возможности восстановления после частичных сбоев серверов: если в рабочем узле, производящем операцию предварительной обработки или свертки, возникает сбой, то его работа может быть передана другому рабочему узлу (при условии, что входные данные для проводимой операции доступны). Пользователю достаточно описать процедуру обработки данных в виде нескольких функций, после чего система автоматически распределяет вычисления по кластеру, обрабатывает отказы машин, балансирует нагрузку и координирует взаимодействия между машинами.

В рамках концепции MapReduce предполагается, что данные организованы в виде некоторого набора упорядоченных записей, а их обработка происходит в три стадии: Map, Shuffle и Reduce (рис. 1).

Стадия Map. На этой стадии выполняется предварительная обработка и фильтрация данных при помощи функции Map, которую определяет пользователь. Принцип работы подобен операции Map в функциональных языках программирования – пользовательская функция применяется к каждой входной записи и возвращает множество пар ключ – значение. Все запуски функции работают независимо и могут работать параллельно, в том числе на разных машинах кластера. Функция Map, как правило, применяется на той же машине, на которой хранятся данные. Это позволяет снизить передачу данных по сети (принцип локальности данных) [3].

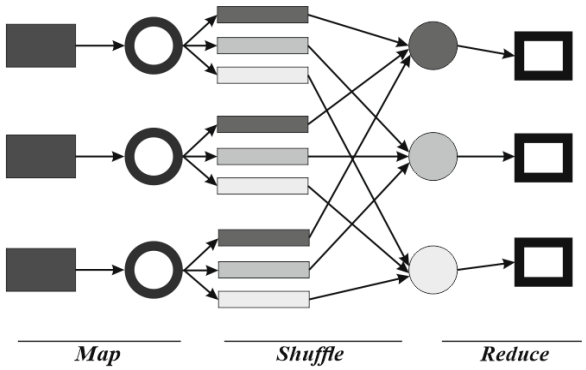

Рис. 1. Стадии работы парадигмы MapReduce

Стадия Shuffle. На этой стадии вывод функции Map разбивается на специальные секции (корзины). Каждая корзина соответствует одному ключу вывода стадии Map. Кроме того, она принимает на вход совокупность записей, соответствующих данному ключу, и общее количество Reduce-задач, а возвращаемым значением является номер задачи, в которой обрабатывалась каждая запись. Каждая секция формируется на основе функции хеширования, которая вызывается для каждого ключа и зависит от определенных критериев, например от номера задачи. Для ускорения процесса обработки информации очень часто на данной стадии применяют алгоритмы параллельной сортировки. В первую очередь они требуются в тех случаях, когда разные атомарные обработчики возвращают наборы с одинаковыми ключами, при этом правила сортировки на этой фазе могут быть заданы программно и использовать какие-либо особенности внутренней структуры ключей разделения (partition key) [4].

Стадия Reduce. Каждая корзина со значениями, сформированная на стадии Shuffle, попадает на вход функции Reduce. Эта функция вычисляет финальный результат для каждой отдельной секции. Все запуски Reduce, как и функция Map, работают независимо и могут работать параллельно, в том числе на разных машинах кластера. Для некоторых видов обработки свертка не требуется, и каркас возвращает в этом случае набор отсортированных пар, полученных базовыми обработчиками.

Парадигма MapReduce достаточна гибкая и может легко адаптироваться под разные типы задач, включая в себя дополнительные стадии обработки информации. Например, стадия *Combine* применяется в тех случаях, когда в результатах функции Map содержится значительное число повторяющихся значений промежуточного ключа, а определенная пользователем задача Reduce является коммутативной и ассоциативной. В таких случаях необходимо осуществить частичную агрегацию данных до их передачи по сети. Функция Combine выполняется на той же машине, что и задача Map. Результаты функции Combine помещаются в промежуточные файлы, которые впоследствии пересылаются в задачи Shuffle или Reduce [5].

Парадигма распределенных вычислений MapReduce в настоящее время широко используется не только для эффективной обработки больших объемов данных, но и для решения прикладных задач, связанных с расширенной обработкой текста, сортировкой данных, индексированием документов, вычислением индексов цитируемости, статистическим анализом, машинным обучением, обработкой изображений. Классифицировать многообразие этих задач только по области применения не представляется возможным по причине того, что многие области знаний тесно взаимодействуют между собой. Имеет смысл добавить другой критерий, связанный со стадиями обработки данных парадигмой MapReduce. В результате исследования было выделено четыре класса задач, при решении которых применяют данную модель распределенных вычислений.

К первому классу, называемому *MapReduce*, необходимо отнести все возможные методы, алгоритмы, использующие в исследуемой парадигме минимум две стадии: Map и Reduce. При решении практических задач зачастую стадии Shuffle или Combine не нужны. В таком случае общая схема работы парадигмы MapReduce упрощается и будет иметь следующий вид:

1. В модель распределенных вычислений подается коллекция документов (записей).

2. Функция Map применяется к каждой паре входных данных и возвращает набор промежуточных пар.

3. MapReduce-контейнер группирует промежуточные значения, связанные с одним ключом, и передает эти значения функции Reduce. Она преобразует промежуточные значения в окончательный набор значений для данного ключа. Как правило, это одно агрегированное значение, например сумма.

К группе *MapOnly* относят задачи, для решения которых можно обойтись только стадией Map (рис. 2). Примерами таких задач являются фильтрация данных (например, поиск нужной информации в лог-файлах по определенному критерию), преобразование данных (например, удаление определенного свойства в JSON-документе или перевод текста в нижний регистр), загрузка и выгрузка данных из внешнего источника (например, вставка записей в NoSQL-базу данных). В описанных типах задач пользователю требуется получить набор пар ключ – значение, поэтому другие стадии, такие как Combine, Reduce, ему не нужны [6].

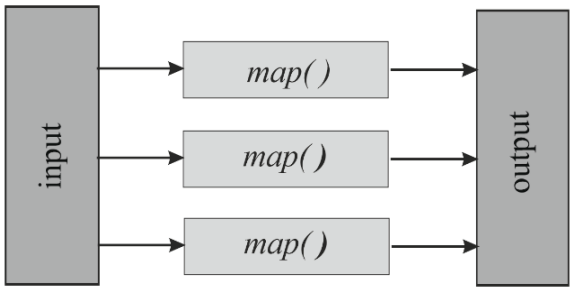

Рис. 2. Схема работы MapOnly

Цепочки MapReduce. К данной группе относят ситуации, когда для решения определенных задач реализации одной MapReduce-модели недостаточно. Тогда их объединяют в цепочки, которые могут выполняться либо линейно, либо представлять собой более сложный направленный ациклический граф. Для линейной цепочки проще всего запускать задания одно за другим, дожидаясь успешного завершения задания перед запуском следующего. Если одно из заданий завершается неудачно, то последующие задания в конвейере выполняться не будут. Когда их последовательность сложнее линейной цепочки, то необходимо определенным образом организовать поток операций, учитывая информацию о зависимости между вызовами MapReduce-задач. Если одно из заданий завершается с ошибкой, планировщик должен прекратить анализ зависимостей и прервать выполнение всей цепочки задач (рис. 3).

Рис. 3. Цепочки MapReduce

ReduceJoin. К данному классу относят задачи, в которых необходимо объединить содержимое нескольких документов по некоторому ключу в выходном потоке данных. Результат работы этих задач очень похож на принцип

работы с реляционными базами данных, где часто используют очень удобную операцию Join, позволяющую совместно обрабатывать содержание некоторых таблиц, объединив их по некоторому ключу. Примером таких задач является объединение двух или более лог-файлов сервера в один итоговый документ либо определение, на какой из двух серверов пользователь чаще заходит по его IP-адресу [4].

Модель ReduceJoin функционирует, используя следующий алгоритм (рис. 4):

1. На вход поступает две коллекции документов (записей).

2. Каждая из коллекций запускает отдельную MapOnly-задачу, преобразующую входные данные к паре ключ – значение. В качестве ключа используется поле, по которому нужно объединять записи коллекций, а в качестве значений выступает пара Type (тип коллекции) и Value (любые дополнительные данные, привязанные к ключу).

3. Результат работы MapOnly подается на вход следующей модели MapReduce. Эта цепочка должна содержать пустую функцию Map, которая копирует входные данные. Далее на стадии Shuffle данные разделяются по ключам и подаются на вход функции Reduce в виде пары, где в качестве значения используется массив элементов Type и Value.

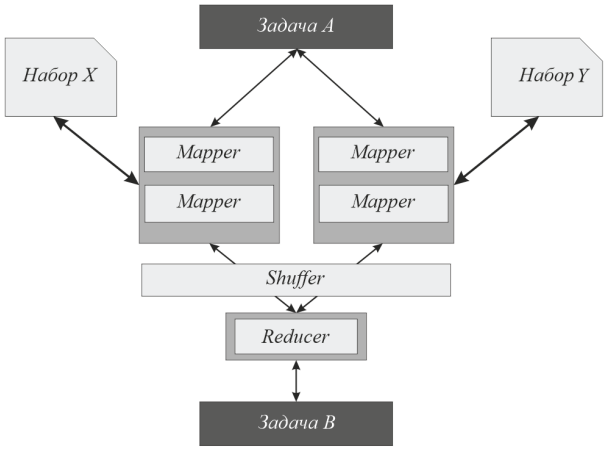

Рис. 4. Схема работы ReduceJoin

Модель MapReduce накладывает ряд ограничений на реализующее ее программное средство в связи с необходимостью автоматизировать распараллеливание, запуск и управление вычислениями на кластере. Кроме того, данная парадигма всегда требует полного сканирования данных, поэтому использование индексов здесь недопустимо. Это означает, что подход MapReduce плохо применим и требует дополнительных оптимизаций, когда необходимо получить ответ в режиме реального времени.

Эффективная реализация MapReduce невозможна без эффективной организации способа хранения данных на кластерной системе. Для этой цели используются распределенные файловые системы (DFS), обеспечивающие высокую производительность, масштабируемость, надежность и доступность данных. Эти файловые системы должны быть оптимизированы для хранения файлов большого размера, эффективного использования сетевых ресурсов и оптимизации под высокую агрегированную пропускную способность, нестандартный интерфейс файловой системы, а также ослабленную модель целостности данных, связанную с хранением слабоструктурированной или не структурированной информации [6].

Запуском MapReduce-задач на кластере должен управлять планировщик, который отслеживает состояние всех узлов и подбирает группу машин для выполнения задания. Вызовы функции Map распределяются между несколькими машинами путем автоматического разбиения входных данных, хранящихся в распределенной файловой системе, на несколько частей. Полученные порции данных могут обрабатываться параллельно различными машинами. Вызовы Reduce распределяются путем разбиения пространства промежуточных ключей на совокупность частей, определяемых с помощью заданной функции разбиения. Каждый из Reduce-процессов загружает со всех Map-процессов порции обработанных данных с соответствующими значениями промежуточных ключей, производит сортировку и объединение этих данных, после чего выполняет функцию Reduce. Результаты вычислений записываются в виде файлов в DFS [5].

Парадигма MapReduce используется многими компаниями, такими как: Google (служит для параллельных вычислений над очень большими, несколько петабайт, наборами данных в компьютерных кластерах), CouchDB (использует MapReduce для определения представлений поверх распределенных документов), MongoDB (позволяет применять MapReduce для параллельной обработки запросов на нескольких серверах), Apache Hadoop (фреймворк для разработки и выполнения распределенных программ), Nvidia (распараллеливание вычислений на видео-ядрах с использованием технологий CUDA), Яндекс (обработка и анализ данных Интернет-сайтов). Каждая компания имеет свои закрытые реализации моделей MapReduce, позволяющие выполнять задачи, написанные на языках Java, C++, C, Python, JavaScript, C#, Perl, Erlang, Ruby.

Заключение. В данной статье рассмотрена одна из основных моделей распределенных

вычислений MapReduce, применяемая для обработки и анализа больших массивов данных на кластерных системах. Выявлено, что отличительными особенностями рассматриваемой парадигмы являются ее высокая гибкость, простота и удобство использования. Она скрывает от пользователя детали организации вычислений на кластерной системе и открывает новые возможности по применению данных систем в научных и прикладных исследованиях для решения задач, связанных с обработкой изображений и текстовой информации, сортировкой данных, искусственным интеллектом.

В статье также была представлена классификация задач, решаемых с использованием исследуемой модели. В качестве классификаторов применялись стадии работы MapReduce-контейнера, а также учитывались возможные сценарии прикладного использования решаемых задач в распределенных компьютерных средах.

Литература

1. Евстигнеева И. Революция в аналитике. Как в эпоху Big Data улучшить ваш бизнес с помощью операционной аналитики? М.: Альпина Паблишер, 2016. 320 с.

2. Натан М. Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени. М.: Вильямс, 2015. 368 с.

3. Фрэнкс Б. Укрощение больших данных. Как извлекать знания из массивов информации с помощью глубокой аналитики? М: Манн, Иванов и Фербер, 2014. 352 с.

4. Сухобоков А. А. Влияние инструментария Big Data на развитие научных дисциплин, связанных с моделированием. М.: МГТУ им. Н. Э. Баумана, 2015. 51 с.

5. Dean J., Ghemawat S. MapReduce: Simplified data processing on large clusters // Proceedings of Operating Systems Design and Implementation (OSDI). 2004. P. 137–150.

6. Том У. Hadoop: Подробное руководство. СПб.: Питер, 2015. 670 с.

References

1. Evstigneeva I. *Revolyutsiya v analitike. Kak v epokhu Big Data uluchshit' vash biznes s pomoshch'yu operatsionnoy analitiki* [The revolution in Analytics. As in the era of Big Data to improve your business with operational Analytics?]. Moscow, Al'pina Pablisher Publ., 2016. 320 p.

2. Natan M. *Bol'shiye dannyye. Printsipy i praktika postroeniya masshtabiruemykh sistem obrabotki dannykh v real'nom vremeni* [Big data. Principles and practice of building highly scalable data processing systems in real time]. Moscow, Vil'yams Publ., 2015. 368 p.

3. Frenks B. *Ukroshcheniye bol'shikh dannykh. Kak izvlekat' znaniya iz massivov informatsii s pomoshch'yu glubokoy analitiki* [The taming of big data. How to extract knowledge from the massive amounts of information using deep Analytics?]. Moscow, Mann, Ivanov i Ferber Publ., 2014. 352 p.

4. Sukhobokov A. A. *Vliyaniye instrumentariya Big Data na razvitie nauchnykh distsiplin, svyazannykh s modelirovaniem* [The impact of Big Data tools for the development of scientific disciplines related to modeling]. Moscow, MGTU imeni N. E. Baumana Publ., 2015. 51 p.

5. Dean J., Ghemawat S. MapReduce: Simplified data processing on large clusters. *Proceedings of Operating Systems Design and Implementation (OSDI)*, 2004, pp. 137–150.

6. Tom U. *Hadoop: Podrobnoye rukovodstvo* [Hadoop: The Definitive Guide]. St. Petersburg, Piter Publ., 2015. 670 p.

Информация об авторе

Гладкий Максим Валерьевич – ассистент кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: MaksHladki@gmail.com

Information about the author

Gladkiy Maksim Valer'yevich – assistant lecturer, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: MaksHladki@gmail.com

Поступила 10.03.2016