СРЕДСТВА ОБРАБОТКИ БОЛЬШИХ ДАННЫХ

MEANS OF BIG DATA PROCESSING

D. Chaikovsky

Summary. Cloud computing has served as an impetus for the development of parallel processing of Big Data. How to take advantage of the huge amount of data in order to obtain additional information and knowledge to optimize business activities and improve the quality of goods and services is an urgent problem. The article provides an analysis of modern methods and software for efficient parallel processing of large amounts of data.

Keywords: Big Data, parallel programming, MapReduce, Apache Hadoop, Apache Spark, MPI.

Чайковский Дмитрий Станиславович
К.ф.-м.н., доцент, Саратовская государственная юридическая академия
chaikovskyds@gmail.com

Аннотация. Облачные вычисления послужили импульсом для развития параллельной обработки Больших Данных. Как воспользоваться колоссальным объемом данных с целью получения дополнительной информации и знаний для оптимизации коммерческой деятельности и повышения качества товаров и услуг является актуальной проблемой. В статье приводится анализ современных методов и программных средств эффективной параллельной обработки большого объема данных.

Ключевые слова: Большие Данные, параллельное программирование, MapReduce, Apache Hadoop, Apache Spark, MPI.

Введение

Стремительно развивающиеся технологии Больших Данных [1] нуждаются в новых методах и инструментах обработки данных. Для работы с Большими Данными необходимы программы, которые обеспечивают взаимодействия физических и виртуальных машин для обработки данных за разумный промежуток времени.

В настоящее время разрабатываются программные средства, которые организуют эффективную работу множества вычислительных машин, в котором каждая получает для обработки свою часть данных, после чего результаты вычислений со всех вычислительных машин объединяются воедино. Обработка локальных данных происходит намного быстрее тех, которые доставляются из сети, поэтому вопрос о распределении данных в кластере и способе объединения машин в кластер является важным вопросом при работе с Big Data.

Насколько огромны Big Data, настолько и широко их использование. Известным примером является анализ данных пользователей социальных сетей. Чтобы узнать больше о человеке изучается пользовательский контент, интересы пользователя используются поисковым системами, с целью подбора рекламы, релевантной интересам пользователя, а также предоставления исчерпывающих ответов на поисковые запросы. В настоящее время существует два крупнейших источника данных — транзакционные данные, в которых содержатся курсы валют, личные данные клиентов банка, включая истории покупок; и данные датчиков, большая часть которых состоит из *Internet of Things (IoT)* или Интернета вещей.

Одним из наиболее известных методов преобразования необработанных данных в полезную информацию является *MapReduce*, представленным компанией *Google*. *MapReduce* — метод распределенных вычислений, на основе параллельной обработки большого количества данных, вплоть до нескольких петабайт, на компьютерных кластерах. *MapReduce* состоит из двух частей: функции *Map*, которая выполняет сортировку и фильтрацию данных, помещает их в категории, для последующего анализа и функции *Reduce*, которая предоставляет сводку этих данных путем их объединения.

Несмотря на то, что часто при упоминании данного метода, ссылаются на исследования в *Google*, в настоящее время *MapReduce* — общее обозначение модели, которая используется во многих технологиях.

Средства анализа больших данных

Самым распространённым и наиболее мощным инструментом для анализа больших данных является *Apache Hadoop*. *Apache Hadoop* представляет собой свободно распространяемую платформу для хранения и обработки данных [2] в больших масштабах. *Hadoop* может работать на стандартном оборудовании, что упрощает его использование с имеющимся центром обработки данных, коме того, возможно проведение анализа данных в облаке. *Hadoop* состоит из четырех основных частей:

1. *The Hadoop Distributed File System (HDFS)* [3] — распределенная файловая система, предназначенная для очень высокой пропускной способности;
2. *YARN*-платформа для управления ресурсами *Hadoop* и планирования программ, которые будут выполняться в инфраструктуре *Hadoop*;

3. *MapReduce*, как было описано выше, модель для обработки больших данных;
4. Общий набор библиотек для других модулей.

Другой инструмент, который в последнее время является весьма популярным — *Apache Spark*. Основное преимущество *Spark* заключается в том, что он хранит большую часть данных для обработки в оперативной памяти, что в отличие от жесткого диска, позволяет получить гораздо более быстрый результат при проведении анализа. В зависимости от вида операции, увеличение скорости обработки может составлять 100 и более раз. *Spark* может использовать распределенную файловую систему *Hadoop*, а также может работать с другими хранилищами данных, такими как *Apache Cassandra* или *OpenStack Swift*. *Spark* легко запустить на одном локальном компьютере, что упрощает тестирование и разработку.

Существует огромное множество решений с открытым исходным кодом для работы с большими данными, многие из которых специализированы для обеспечения оптимального функционала и производительности для определенной задачи или для конкретных аппаратных конфигураций. И по мере того, как Большие Данные продолжают расти в размерах и значимости, количество инструментов с открытым исходным кодом для работы с ними, безусловно, будет продолжать расти.

Технологии параллельного программирования

Большая часть данных поступает из журналов программного обеспечения, камер, микрофонов, *RFID*-считывателей, беспроводных сенсорных сетей и т.д. Эти устройства генерируют высокоскоростные данные, и их производительность постоянно растет. Хранение таких данных не требует больших финансовых затрат, а на их основе можно добывать ценную информацию.

Приведем некоторые примеры параллельного программного обеспечения для анализа данных. Программное обеспечение, написанное на *SQL*, работает по принципу параллельной обработки уже достаточно давно. Но лишь теперь, когда обработка больших данных стала реальностью, все больше программистов интересуются построением программ на параллельной модели, а так как программирование на *SQL* является достаточно трудоемким процессом, требующим определенного набора знаний, возникла потребность в создании альтернативного средства. На смену пришла платформа параллельного программирования [4] *MapReduce*, которая получила известность благодаря ее использованию в компаниях по предоставлению веб-поиска.

Чтобы понять концепцию параллельного программного обеспечения давайте посмотрим, чего уже достигла компьютерная индустрия. Наиболее успешной отраслью параллельных исследований являются параллельные базы данных. Вместо того чтобы требовать от программиста разбивать алгоритм на отдельные потоки, которые будут выполняться на отдельных ядрах, параллельные базы данных позволяют им «разрезать» входные таблицы данных на части и обрабатывать каждую часть через одну и ту же программу на отдельном процессоре. Такая модель «параллельного потока данных» делает программирование на параллельных машинах такой же простой, как программирование для одного процессора. Она работает на кластерах компьютеров в центре обработки данных по принципу *«shared-nothing»*, то есть без разделения оперативной и дисковой памяти между процессорами. Участвующие машины могут обмениваться данными с помощью простых потоков сообщений данных, без необходимости дорогостоящей общей оперативной памяти или дисковой инфраструктуры.

SQL предоставляет собой язык высокого уровня, который является более гибким и оптимизируемым, но менее знакомым многим программистам. *MapReduce* же требует от программистов писать обычный код на таких языках, как *C*, *Java*, *Python* и *Perl*. *MapReduce* позволяет записывать и считывать программы из обычных файлов файловой системы и не требует определения схемы базы данных. *MapReduce* является настолько убедительным средством параллельного программирования, что он используется для подготовки нового поколения программистов в области параллельных вычислений. Многие бакалавры компьютерных наук за рубежом теперь изучают *MapReduce* и индустрия в этой области охотно это поддерживает.

Средства обработки больших данных

По мере увеличения объема данных до экзабайт, для всё большего числа организаций самой насущной проблемой станет разработка способов извлечения данных и придания им смысла. Ведущим примером является *Google*, который использует *MapReduce* [5] для обработки более 20 петабайт данных в день.

Apache Hadoop используется для анализа огромных объемов данных без необходимости приобретения дорогостоящего фирменного оборудования или программного обеспечения. Однако следует понимать, что использование *Hadoop* не предоставит компаниям конкретные рекомендации по принятию более взвешенных решений. Мощь масштабируемой инфраструктуры должна быть дополнена инструментами интеллектуального анализа данных и машинного обучения, визуализа-

цией результатов и более простыми способами отслеживания и анализа результатов за короткий промежуток времени. Кроме того, существует целая область аналитики в реальном времени, которая выходит за рамки пакета *Hadoop*.

В настоящее время происходит развитие функциональных языков *Pig*, *Sawzall*, *Microsoft Dryad*. Несмотря на то, что эти системы отличаются по интерфейсу, модели программирования, которые они предоставляют, имеют схожие цели, которые заключаются в самостоятельном решении таких задач параллельного программирования, как отказоустойчивость и оптимизация выполнения. Данный класс программ дает возможность разработчикам продолжать писать последовательные программы.

Платформа обработки распределяет программу между доступными узлами и выполняет каждый экземпляр программы на соответствующем фрагменте данных. Многое при анализе больших данных можно реализовать методом *SPMD* (единая программа, множество данных). В *SPMD* используются различные методы параллелизации, такие как *MPI* (*Message Passing Interface — интерфейс передачи сообщений*), *MapReduce*, или технологии потока операций, каждый из которых отличаются производительностью [6] и удобством использования.

Простота программирования в *MapReduce* является главным фактором, при выборе именного этого средства. Однако ограничения производительности архитектуры *MapReduce* делают её использование неэффективной для некоторых приложений (например, для алгоритмов машинного обучения). Более общий подход к потоку операций и потоку данных представляет большой интерес, поэтому данные средства будут рассмотрены в последующих разделах.

Среды выполнения для больших данных

Языки высокого уровня, позволяющие реализовывать параллельное программирование, были основой для исследований в области компьютерных наук, однако в последнее время наблюдается развитие возможностей сред выполнения в этом направлении.

Несмотря на то, что разные платформы различаются правилами работы, существует много сходства между параллельными и распределенными средами вычисления, одно из которых — поддержка обмена сообщениями с различными свойствами.

В платформах на основе итераций результаты одного этапа вычисления многократно повторяются. Это типично для большинства алгоритмов, реализуемых с помощью *MPI*. В конвейерных платформах результаты одного этапа вычисления (например, операции *Map* или *Reduce*) пересылаются другому этапу. Это функциональный параллелизм, типичный для приложений потока операций. Важная особенность в моделях параллельного или распределенного программирования связана с тем, что сегодня как параллелизм в стиле *MPI*, так и распределенные модели (*Hadoop*, *Dryad*, *Web Service*, *Workflow*) реализуются с помощью обмена сообщениями. Это обусловлено тем, что обмен сообщениями позволяет избежать ошибок, возникающих при синхронизации потоков общей памяти. *MPI* является прекрасным примером среды выполнения, которая поддерживает различные характеристики приложения. MPI обеспечивает превосходную производительность и простоту программирования для *MapReduce*. Однако *MPI* не обладает отказоустойчивостью и гибкостью *Hadoop* или *Dryad*.

Модель программирования MAPREDUCE

Модель *MapReduce* (*MR*) была разработана для программирования параллельных приложений обработки большого количества данных. Она привлекательна для высокоскоростной параллельной обработки произвольных данных и сегодня рассматривается как важная модель программирования крупномасштабных приложений, работающих с параллельными данными, таких как веб-индексация, анализ данных и научные моделирования, благодаря своей простоте, через которую пользователи могут выразить относительно сложные распределенные программы. *MapReduce* разбивает вычисления на небольшие задачи, которые выполняются параллельно на нескольких машинах, и легко масштабируется до очень больших кластеров, состоящих из бюджетных компьютеров.

MR-программа состоит только из двух функций, называемых *Map* и *Reduce*, для обработки пар данных ключ-значение. Входной набор данных хранится в наборе разделов распределенной файловой системы, развернутой на каждом узле кластера. Функция *Map* считывает набор «записей» из входного файла, выполняет фильтрацию или преобразования, а затем выводит набор промежуточных записей в виде новых пар ключ-значение. Функция *Map* создает выходные записи, а функция *split* разбивает записи на *R* непересекающихся сегментов, применяя функцию к ключу каждой выходной записи. Эта функция разделения обычно является хэш-функцией, хотя любой детерминированной функции будет достаточно. Каждый блок карты записывается на локальный диск узла обработки. Функция *Map* завер-

шает работу, создав R выходных файлов, по одному для каждого блока.

Имеется несколько экземпляров Map функции на разных узлах вычислительного кластера. Термин экземпляр используется для обозначения уникального выполняемого вызова функции Map или $Reduce$. Планировщик MR назначает каждому экземпляру Map отдельную часть входного файла для обработки. Если есть M таких различных частей входного файла, то есть R файлов на диске для каждой из задач M Map, в общей сложности MA-R файлов $F_{i,j}$, где $1 \leq i \leq M$, $1 \leq j \leq R$. причем, все экземпляры Map используют одну и ту же хэш-функцию; таким образом, все выходные записи с одинаковым хэш-значением хранятся в одном выходном файле.

Вторая фаза программы MR выполняет R экземпляров программы $Reduce$. Входные данные для каждого экземпляра $Reduce$ R_j состоят из файлов $F_{i,j}$, где $1 \leq j \leq M$. эти файлы передаются по сети с дисков узлов Map. Все выходные записи фазы Map с одинаковым хэш-значением используются одним и тем же экземпляром $Reduce$, независимо от того, какой экземпляр Map создал данные. Каждый экземпляр $Reduce$ обрабатывает или объединяет назначенные ему записи, а затем записывает записи в выходной файл (в распределенной файловой системе), который является частью конечного результата вычисления.

Набор входных данных существует в виде набора из одного или нескольких разделов распределенной файловой системы. Планировщик MR должен решить, сколько экземпляров Map запустить и как распределить их по доступным узлам. Кроме того, планировщик должен также принять решение о количестве и расположении узлов, на которых выполняются экземпляры $Reduce$. Центральный контроллер MR отвечает за координацию деятельности системы на каждом узле. Программа завершает выполнение, как только конечный результат запишется в виде новых файлов в распределенной файловой системе.

Ключевым преимуществом $MapReduce$ является автоматическая обработка сбоев. Программист освобождается от решения задач, связанных с отказоустойчивостью. Если узел аварийно завершает работу, $MapReduce$ автоматически перезапускает свои задачи на другом компьютере. Аналогично, если узел доступен, но работает некорректно, $MapReduce$ запускает спекулятивную копию на другом компьютере, чтобы быстрее завершить вычисление. Без этого механизма, известного как «спекулятивное выполнение», выполнение будет происходить намного медленнее. Представители $Google$ отмечают, что в их реализации спекулятивное исполнение может улучшить время отклика задания на 44%.

Проблемы

Технологии и промышленность в настоящее время претерпевают глубокие изменения: крупномасштабные, разнообразные наборы данных и потоки [7] (полученные из датчиков, интернета, транзакций) дают огромную возможность для принятия решений. Помимо огромного объема данных, Большие Данные будут поступать в различных форматах данных: данные датчиков, текст, аудио и видео. Скорость генерации данных постоянно увеличивается, что делает предмет скорости решающим для того, чтобы своевременно получить полезную информацию. Более того, данные поступающие из различных источников различаются по качеству и надежности, поэтому еще одним важным аспектом анализа данных является оценка достоверности результата анализа. Кроме того, визуализация и интерактивный анализ огромных, изменяющихся наборов данных по-прежнему представляют собой многочисленные проблемы для аналитиков данных.

Последние достижения в области компьютерных технологий и подходов обработки данных позволили создать новые инструменты и технологии для Больших Данных. Новые алгоритмы, методы прогнозирования и моделирования, новые подходы к сбору и интеграции данных, анализу и сжатию данных[8], расширенные технологии обработки и обмена данными и информацией, а также новые языки для автоматической оптимизации и распараллеливания сложных программ анализа данных необходимы для одновременного решения вопросов объема, скорости, разнообразия и достоверности анализа данных.

Достижения в области обработки информации, интеграции, обработки сигналов, машинного обучения, интеллектуального анализа данных, сжатия и визуализации откроют новые способы своевременного извлечения полезной, доступной и проверенной информации из огромных и разнообразных наборов данных. Подход «$NoSQL$» связан с целым рядом проблем. Интеграция больших данных различных типов и носителей и высокоскоростная аналитика достоверной информации является серьезной проблемой, не решаемой существующими системами управления данными. Аналитика больших данных должна иметь возможность принимать данные с различных типов носителей, в частности аудио и видеопотоков, при увеличении скорости, уже включив непрерывный анализ данных. Необходимо разработать масштабируемые, простые в использовании системы баз данных или анализа данных, а также новые алгоритмы и методы анализа, одновременно учитывающие различные требования.

Автоматическая оптимизация программ анализа данных, включающих итерации и сложные пользователь-

ские функции, для различных аппаратных платформ, таких как *SIMD* (single instruction, multiple data — одиночный поток команд, множественный поток данных), кластеров или большинства основных процессоров, а также сложных аппаратных архитектур, таких как NUMA (Non-Uniform Memory Access — «неравномерный доступ к памяти») по-прежнему содержит много проблем. К еще одной проблеме можно отнести развитие новых языков программирования и методов масштабируемой обработки и оптимизации сложных программ анализа данных.

Заключение

С появлением облачных вычислений, как перспективного и нового подхода для параллельной обработки данных, крупные компании стали интегрировать платформы для параллельной обработки данных в свой портфель продуктов, облегчая клиентам доступ к службам и упрощая развертывание своих программ.

Существует множество механизмов, которые генерируют Большие Данные в повседневной жизни, что является потенциальным источником увеличения доходов. Изобилие доступных данных в широком спектре областей поднимают новые проблемы и возможности в различных дисциплинах, начиная от науки, техники и технологий. Одна из основных проблем заключается в том, как воспользоваться беспрецедентным объемом данных, которые, как правило, носят разнородный характер, для получения дополнительной информации и знаний в целях повышения качества предлагаемых услуг.

В статье приведен анализ современных возможностей и существующих проблем эффективной параллельной обработки данных.

ЛИТЕРАТУРА

1. Чайковский Д. С. Перспективы развития технологии «Big data» в России. Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. Сборник статей VII Международной научно-практической конференции: в 4 частях. Пенза: МЦНС «Наука и Просвещение», 2017, часть 1, С. 199–201.
2. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communication of the ACM 51(1), 2008, pp. 107–113.
3. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Oper. Syst. Rev. 41(3), 2007, pp. 59–72.
4. Fox, G., Bae, S.H., Ekanayake, J., Qiu, X., Yuan, H.: Parallel data mining from multicore to cloudy grids. In: High Performance Computing Workshop, vol. 18, 2009, pp. 311–340.
5. Tudoran, R., Costan, A., Antoniu, G.: Mapiterativereduce: a framework for reduction-intensive data processing on azure clouds. In: Proceedings of Third International Workshop on MapReduce and Its Applications Date, 2012, pp. 9–16.
6. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving mapreduce performance in heterogeneous environments. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, 2008, pp. 29–42.
7. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: Distributed stream computing platform. In: 2010 IEEE International Conference on Data Mining Workshops (ICDMW), 2010, pp. 170–177.
8. Pike, R., Dorward, S., Griesemer, R., Quinlan, S.: Interpreting the data: parallel analysis with sawzall. Sci. Program. 13(4), 2005, 277–298.

Журнал «Современная наука: актуальные проблемы теории и практики»