УДК 004.896(06)

А.С. Сульженко, Е.С. Сакало, И.А. Ревенчук

Харьковский национальный университет радиоэлектроники, Харьков

РАБОТА С БОЛЬШИМИ ОБЪЕМАМИ ДАННЫХ В ОБЛАКЕ С ПОМОЩЬЮ MAPREDUCE

В статье анализируется и рассматривается модель распределённых вычислений MapReduce, используемая для параллельных вычислений над большими наборами данных в компьютерных кластерах.

Ключевые слова: *Google MapReduce, ApacheeHadoop, Map, Reduce.*

Введение и постановка задачи

С каждым годом растут объемы обрабатываемых данных. По оценкам IDC, количество данных, хранимых в электронном виде, в 2011 году является 1.8 зеттабайт 1. Откуда берется эта цифра? Ситуацию проясняют следующие факты:

– FaceBook на своих серверах хранит примерно 10 млрд. картинок;

– фондовая биржа Нью-Йорка ежедневно генерирует около 1 Тб новых данных;

– в 2008 году Google обрабатывал ежемесячно 400 ПБ.

В то же время за последние 20 лет быстродействие винчестеров не увеличилось настолько, чтобы справляться с объемами данных: в 1990 году данные с винчестера объемом 1370 Мб и скоростью доступа 4,4 Мб/с можно было прочесть за 2,5 минуты, а у современных винчестеров скорость доступа около 100 Мб/с при среднем объеме в 1 Тб. Получается, что все данные с жесткого диска можно прочесть за 2,5 часа. Но если мы сможем обрабатывать 1 Тб параллельно, читая его, скажем, со ста машин одновременно, то на обработку данных нам потребуется около 2 минут.

Именно это и позволяет сделать технология MapReduce. В рамках этой технологии Google была предложена абстрактная модель, которая позволила выражать простые вычисления, в то же время, пряча сложные детали параллелизации: обработку ошибок, распределение данных, балансировку нагрузки.

Первая реализация этой модели была выполнена на основе распределенной файловой системы той же компании GFS (GoogleFileSystem). Эта реализация активно используется в программных продуктах самой Google, но является сугубо проприетарной и недоступна для использования вне Google.

Альтернативная, свободно доступная реализация HadoopMapReduce (с открытыми исходными текстами) была выполнена в проекте Hadoop сообщества Apache.

Она основана на использовании распределенной файловой системы HDFS (HadoopDistributedFileSystem), также разработанной в проекте Hadoop. Реальную популярность MapReduce принесла именно реализация Hadoop в силу своей доступности и открытости, а широкое использование Hadoop-MapReduce в различных исследовательских проектах приносит несомненную пользу этой системе, стимулируя разработчиков к ее постоянному совершенствованию.

В этой модели вычисления производятся над множествами входных пар "ключ-значение", и в результате каждого вычисления также производится некоторое множество результирующих пар "ключ-значение".

Для представления вычислений в среде MapReduce используются две основные функции: Map и Reduce. Обе функции явно кодируются разработчиками приложений в среде MapReduce.

Функция Map в цикле обрабатывает каждую пару из множества входных пар и производит множество промежуточных пар "ключ-значение". Среда MapReduce группирует все промежуточные значения с одним и тем же ключом I и передает их функции Reduce.

Функция Reduce получает значение ключа I и множество значений, связанных с этим ключом. В типичных ситуациях каждая группа обрабатывается (в цикле) таким образом, что в результате одного вызова функции образуется не более одного результирующего значения.

Реализации MapReduce от Google и Hadoop ориентированы на использование в кластерной распределенной среде со следующими основными характеристиками:

– узлы среды выполнения MR-приложений обычно представляют собой компьютеры общего назначения с операционной системой Linux;

– используется стандартное сетевое оборудование с адаптерами, рассчитанными на скорости передачи в 100 мегабит в секунду или 1 гигабит в секунду, но средняя пропускная способность существенно ниже;

– кластер состоит из сотен или тысяч машин, так что вполне вероятны отказы отдельных узлов;

– для хранения данных используются недорогие дисковые устройства, подключенные напрямую к отдельным машинам;

– для управления данными, хранящимися на этих дисках, используется распределенная файловая система;

– пользователи представляют свои задания в систему планирования; каждое задание состоит из некоторого набора задач, которые отображаются планировщиком на некоторый набор узлов кластера.

Основная часть

Далее будет рассмотрена реализация модели MapReduce в распределенной среде.

1. Выполнение MR-приложения

Вызовы Map распределяются по нескольким узлам кластера путем разделения входных данных на M непересекающихся групп (split). Входные группы могут параллельно обрабатываться на разных машинах. Вызовы Reduce распределяются путем разделения пространства ключей промежуточных ключей на R частей с использованием некоторой функции разделения (например, функции хэширования). Число разделов R и функция разделения задаются пользователем.

Выполнение MR-программы происходит следующим образом. Сначала среда MapReduce расщепляет входной файл на M частей, размер которых может задаваться пользователем. Затем сразу в нескольких узлах кластера запускается основная программа MapReduce. Один из экземпляров этой программы играет специальную роль и называется распорядителем (master). Остальные экземпляры являются исполнителями (worker), которым распорядитель назначает работу. Распорядитель должен назначить исполнителям для выполнения M задач Map и R задач Reduce.

Исполнитель, которому назначена задача Map, читает содержимое соответствующей группы, разбирает пары "ключ-значение" входных данных и передает каждую пару в определенную пользователем функцию Map. Промежуточные пары "ключ-значение", производимые функцией Map, буферизуются в основной памяти. Периодически буферизованные пары, разделяемые на R областей на основе функции разделения, записываются в локальную дисковую память исполнителя. Координаты этих сохраненных на диске буферизованных пар отсылаются распорядителю, который, в свою очередь, передает эти координаты исполнителям задачи Reduce. I-й Reduce-исполнитель снабжается координатами всех i-х областей буферизованных пар, произведенных всеми M Map-исполнителями.

После получения этих координат от распорядителя исполнитель задачи Reduce с использованием механизма удаленных вызовов процедур переписывает данные с локальных дисков исполнителей задачи Map в свою память или на локальный диск (в зависимости от объема данных). После переписи всех промежуточных данных выполняется их сортировка по значениям промежуточного ключа для образования групп с одинаковым значением ключа. Если объем промежуточных данных слишком велик для выполнения сортировки в основной памяти, используется внешняя сортировка.

Далее Reduce-исполнитель организует цикл по отсортированным промежуточным данным и для каждого уникального значения ключа вызывает пользовательскую функцию Reduce с передачей ей в качестве аргумента значения ключа и соответствующее множество значений. Результирующие пары функции Reduce добавляются в окончательный результирующий файл данного Reduce-исполнителя. После завершения всех задач Map и Reduce распорядитель активизирует программу пользователя, вызывавшую MapReduce.

После успешного завершения выполнения задания MapReduce результаты размещаются в R файлах распределенной файловой системы (имена этих результирующих файлов задаются пользователем). Обычно не требуется объединять их в один файл, потому что часто полученные файлы используются в качестве входных для запуска следующего MR-задания или в каком-либо другом распределенном приложении, которое может получать входные данные из нескольких файлов.

2. Отказоустойчивость

Поскольку технология MapReduce предназначена для обработки громадных объемов данных с использованием сотен и тысяч машин, в ней обязательно должна присутствовать устойчивость к отказам отдельных машин.

Отказ исполнителя. Распорядитель периодически посылает каждому исполнителю контрольные сообщения. Если некоторый исполнитель не отвечает на такое сообщение в течение некоторого установленного времени, распорядитель считает его вышедшим из строя. В этом случае все задачи Map, уже выполненные и еще выполнявшиеся этим исполнителем, переводятся в свое исходное состояние, и можно заново планировать их выполнение другими исполнителями. Аналогично распорядитель поступает со всеми задачами Reduce, выполнявшимися отказавшим исполнителем к моменту отказа.

Завершившиеся задачи Map выполняются повторно по той причине, что их результирующие пары сохранялись на локальном диске отказавшего исполнителя и поэтому недоступны в других узлах. Завершившиеся задачи Reduce повторно выполнять не требуется, поскольку их результирующие пары сохраняются в глобальной распределенной файловой системе. Если некоторая задача Map выполня-

лась исполнителем A, а потом выполняется исполнителем B, то об этом факте оповещаются все исполнители, выполняющие задачи Reduce. Любая задача Reduce, которая не успела прочитать данные, произведенные исполнителем A, после этого будет читать данные от исполнителя B.

Отказ распорядителя. В реализациях MapReduce от Google и Hadoop какая-либо репликация распорядителя не производится. Считается, что поскольку распорядитель выполняется только в одном узле кластера, его отказ маловероятен, и если он случается, то аварийно завершается все выполнение MapReduce. Однако известно, что несложно организовать периодический сброс в распределенную файловую систему всего состояния распорядителя, чтобы в случае отказа можно было запустить его новый экземпляр в другом узле с данной контрольной точки.

Семантика при наличии отказов. Если обеспечиваемые пользователями функции Map и Reduce являются детерминированными (т.е. всегда выдают одни и те же результаты при одинаковых входных данных), то при их выполнении в среде распределенной реализации MapReduce при любых условиях обеспечивает тот же результат, как при последовательном выполнении всей программы при отсутствии каких-либо сбоев.

Это свойство обеспечивается за счет атомарности фиксации результатов задач Map и Reduce. Каждая выполняемая задача записывает свои результаты в частные временные файлы. Задача Reduce производит один такой файл, а задача Map – R файлов, по одной на каждую задачу Reduce. По завершении задачи Map исполнитель посылает распорядителю сообщение, в котором указываются имена R временных файлов. При получении такого сообщения распорядитель запоминает эти имена файлов в своих структурах данных. Повторные сообщения о завершении одной и той же задачи Map игнорируются.

При завершении задачи Reduce ее исполнитель атомарным образом переименовывает временный файл результатов в окончательный файл. Если одна и та же задача Reduce выполняется несколькими исполнителями, то для одного и того же окончательного файла будет выполнено несколько операций переименования. Если в используемой распределенной файловой системе операция переименования является атомарной, то в результате в файловой системе сохранятся результаты только какого-либо одного выполнения задачи Reduce.

3. Резервные задачи

Чаще всего к увеличению общего времени выполнения задания MapReduce приводит наличие "отстающих" ("straggler") – узлов кластера, в которых выполнение одной из последних задач Map или Reduce занимает необычно долгое время (например, из-за некритичной неисправности дискового устройства).

Для смягчения проблемы "отстающих" в MapReduce применяется следующий общий механизм. Когда задание близится к завершению, для всех еще не завершившихся задач назначаются дополнительные, резервные исполнители. Задача считается выполненной, когда завершается ее первичное или резервное выполнение. Этот механизм настраивается таким образом, чтобы потребление вычислительных ресурсов возрастало не более чем на несколько процентов. В результате удается существенно сократить время выполнения крупных MR-заданий.

4. Расширенные средства

В большинстве приложений MapReduce оказывается достаточно просто написать свои функции Map и Reduce. Однако в ряде случаев оказываются полезными некоторые расширения базовых функциональных возможностей.

5. Функция разделения

Пользователи MapReduce явно указывают требуемое число задач Reduce R (и соответствующих результирующих файлов). Данные распределяются между этими задачами с использованием некоторой функции разделения от значений промежуточного ключа. По умолчанию используется функция хэширования (например, "hash(key) mod R"). Однако пользователи MapReduce могут специфицировать и собственные функции разделения.

6. Гарантии упорядоченности

Внутри каждого раздела, передаваемого задаче Reduce, данные упорядочены по возрастанию значений промежуточного ключа. Это позволяет задачам Reduce производить отсортированные результирующие файлы.

7. Функция-комбинатор

В некоторых случаях в результатах задачи Map содержится значительное число повторяющихся значений промежуточного ключа, а определенная пользователем задача Reduce является коммутативной и ассоциативной. В таких случаях пользователь может определить дополнительную функцию-комбинатор (Combiner), выполняющую частичную агрегацию таких данных до их передачи по сети. Функция Combiner выполняется на той же машине, что и задача Map. Обычно для ее реализации используется тот же самый код, что и для реализации функции Reduce. Единственное различие между функциями Combiner и Reduce состоит в способе работы с их результирующими данными. Результаты функции Reduce записываются в окончательный файл результатов. Результаты же функции Combiner помещаются в промежуточные файлы, которые впоследствии пересылаются в задачи Reduce.

8. Форматы входных и результирующих данных

В библиотеке MapReduce поддерживается возможность чтения входных данных в нескольких разных форматах. Например, в режиме "text" каждая строка трактуется как пара "ключ-значение", где ключ – это смещение до данной строки от начала файла, а значение – содержимое строки. В другом распространенном формате входные данные представляются в виде пар "ключ-значение", отсортированных по значениям ключа.

В каждой реализации формата входных данных известно, каким образом следует расщеплять данные на осмысленные части, которые обрабатываются отдельными задачами Map (например, данные формата "text" расщепляются только по границами строк).

Пользователи могут добавить к реализации собственные форматы входных данных, обеспечив новую реализацию интерфейса reader (в реализации Hadoop – RecordReader). Reader не обязательно должен читать данные из файла, можно легко определить reader, читающий данные из базы данных или из некоторой структуры в виртуальной памяти.

Аналогичным образом поддерживаются возможности генерации данных в разных форматах, и имеется простая возможность определения новых форматов результирующих данных.

Выводы

Модель программирования MapReduce жертвует гибкостью и универсальностью ради поддержки автоматического управления вычислениями в распределенной среде.

Круг приложений MapReduce ограничен параллельными по данным задачами, где не требуется организовывать сложное взаимодействие между процессами. Но, как показывает практика, таких приложений очень много, и тот уровень удобства, на который MapReduce поднимает программирование подобных вычислений, заслуживает внимания. Следующим шагом в данном направлении является технология MicrosoftDryad, более универсальная и мощная, чем MapReduce.

Несмотря на то, что оригинальные реализации технологий являются закрытыми разработками, благодаря opensource проектам активно развиваются их общедоступные аналоги.

Хочется отметить, что данные технологии, пришедшие из индустрии, начинают активно использоваться в академической среде. Как упоминалось во введении, стоящие перед исследователями вычислительные задачи часто имеют такие же требования, что и задачи Google. Представляется, что подобные технологии в ближайшее время станут неотъемлемой частью современных информационных систем, в которых все чаще возникает потребность в хранении и анализе больших объемов информации.

Список литературы

1. Chang. A distributed storage system for structured data / Chang, J. Dean. – Seattle, USA, 2006. – P. 205-218.

2. Dean J. MapReduce: Simplified data processing on large clusters / J. Dean, S. Ghemawat. – San Francisco, USA, 2004. – P. 137-150.

3. Barroso L.A. Websearchfor a planet: The Google cluster architecture / L.A. Barroso, J. Dean, U. UrsHölzle. – Seattle, USA, 2003. – P. 22-28.

4. HadoopMap/ReduceTutorial [Электронный ресурс] – Режим доступа к ресурсу: http://hadoop.apache.org/ common/docs/current/mapred_tutorial.html, свободный.

5. CloudEra [Электронный ресурс]. – Режим доступа к ресурсу: http://www.cloudera.com/, свободный.

Поступила в редколлегию 8.05.2012

Рецензент: д-р техн. наук, проф. Е.П. Путятин, Харьковский национальный университет радиоэлектроники, Харьков.

ОБРОБКА ВЕЛИКИХ ОБСЯГІВ ДАНИХ В ОБЛАЦІ ЗА ДОПОМОГОЮ MAPREDUCE

А.С. Сульженко, Є.С. Сакало, І.А. Ревенчук

У статті аналізується і розглядається модель розподілених обчислень MapReduce, використовувана для паралельних обчислень над великими наборами даних в комп'ютерних кластерах.

Ключові слова: *Google MapReduce, ApacheeHadoop, Map, Reduce.*

HANDLING LARGE AMOUNTS OF DATA IN THE CLOUD BY MAPREDUCE

A.S. Sulzhenko, E.S. Sakalo, I.A. Revenchuk

The given paper analyzes and discusses a model of distributed computing MapReduce, is used for parallel computations over large data sets in the computer clusters.

Keywords: *Google MapReduce, ApacheeHadoop, Map, Reduce.*