ВВЕДЕНИЕ

MapReduce – модель программирования и платформа для пакетной обработки больших объемов данных, разработанная и используемая внутри компании Google для широкого круга приложений. Модель MapeReduce отличается простотой и удобством использования, скрывая от пользователя детали организации вычислений в ненадежной распределенной среде. Пользователю достаточно описать процедуру обработки данных в виде двух функций – map и reduce, после чего система автоматически распределяет вычисления по кластеру из большого количества машин, обрабатывает отказы машин, балансирует нагрузку и координирует взаимодействия между машинами для эффективного использования сетевых и дисковых ресурсов.

Впервые описание MapReduce было опубликовано в работе. За последние четыре года внутри Google было разработано более 10 тысяч программ для MapReduce. В среднем, каждый день на кластерах Google выполняется около тысячи MapReduce-заданий, обрабатывающих вместе более 20 петабайтов данных.

Используемая в Google реализация MapReduce является закрытой технологией, однако существует общедоступная реализация Apache Hadoop.

1 Назначение и области применения

MapReduce можно по праву назвать главной технологией Big Data, т.к. она изначально ориентирована на параллельные вычисления в распределенных кластерах. Суть MapReduce состоит в разделении информационного массива на части, параллельной обработки каждой части на отдельном узле и финального объединения всех результатов.

Программы, использующие MapReduce, автоматически распараллеливаются и исполняются на распределенных узлах кластера, при этом исполнительная система сама заботится о деталях реализации (разбиение входных данных на части, разделение задач по узлам кластера, обработка сбоев и сообщение между распределенными компьютерами). Благодаря этому программисты могут легко и эффективно использовать ресурсы распределённых Big Data систем.

Технология практически универсальна: она может использоваться для индексации веб-контента, подсчета слов в большом файле, счётчиков частоты обращений к заданному адресу, вычисления объём всех веб-страниц с каждого URL-адреса конкретного хост-узла, создания списка всех адресов с необходимыми данными и прочих задач обработки огромных массивов распределенной информации. Также к областям применения MapReduce относится распределённый поиск и сортировка данных, обращение графа веб-ссылок, обработка статистики логов сети, построение инвертированных индексов, кластеризация документов, машинное обучение и статистический машинный перевод. Также MapReduce адаптирована под многопроцессорные системы, добровольные вычислительные, динамические облачные и мобильные среды.

2 История развития главной технологии big data

Авторами этой вычислительной модели считаются сотрудники Google Джеффри Дин (Jeffrey Dean) и Санджай Гемават (Sanjay Ghemawat), взявшие за основу две процедуры функционального программирования: map, применяющая нужную функцию к каждому элементу списка, и reduce, объединяющая результаты работы map [3]. В процессе вычисления множество входных пар ключ/значение преобразуется в множество выходных пар ключ/значение.

Изначально название MapReduce было запатентовано корпорацией Google, но по мере развития технологий Big Data стало общим понятием мира больших данных. Сегодня множество различных коммерческих, так и свободных продуктов, использующих эту модель распределенных вычислений: Apache Hadoop, Apache CouchDB, MongoDB, MySpace Qizmt и прочие Big Data фреймворки и библиотеки, написанные на разных языках программирования. Среди других наиболее известных реализаций MapReduce стоит отметить следующие:

* + Greenplum — коммерческая реализация с поддержкой языков Python, Perl, SQL и пр.;
  + GridGain — бесплатная реализация с открытым исходным кодом на языке Java;
  + Phoenix — реализация на языке С с использованием разделяемой памяти;
  + MapReduce реализована в графических процессорах NVIDIA с использованием CUDA;
  + Qt Concurrent — упрощённая версия фреймворка, реализованная на C++, для распределения задачи между несколькими ядрами одного компьютера;
  + CouchDB использует MapReduce для определения представлений поверх распределённых документов;
  + Skynet — реализация с открытым исходным кодом на языке Ruby;
  + Disco — реализация от компании Nokia, ядро которой написано на языке Erlang, а приложения можно разрабатывать на Python;
  + Hive framework — надстройка с открытым исходным кодом от Facebook, позволяющая комбинировать подход MapReduce и доступ к данным на SQL-подобном языке;
  + Qizmt — реализация с открытым исходным кодом от MySpace, написанная на C#;
  + DryadLINQ — реализация от Microsoft Research на основе PLINQ и Dryad.

3 Как устроен mapreduce: принцип работы

Прежде всего, еще раз поясним смысл основополагающих функций вычислительной модели [2]:

* map принимает на вход список значений и некую функцию, которую затем применяет к каждому элементу списка и возвращает новый список;
* reduce (свёртка) — преобразует список к единственному атомарному значению при помощи заданной функции, которой на каждой итерации передаются новый элемент списка и промежуточный результат.

Для обработки данных в соответствии с вычислительной моделью MapReduce следует определить обе эти функции, указать имена входных и выходных файлов, а также параметры обработки.

Сама вычислительная модель состоит из 3-хшаговой комбинации вышеприведенных функций:

* Map – предварительная обработка входных данных в виде большого список значений. При этом главный узел кластера (master node) получает этот список, делит его на части и передает рабочим узлам (worker node). Далее каждый рабочий узел применяет функцию Map к локальным данным и записывает результат в формате «ключ-значение» во временное хранилище.
* Shuffle, когда рабочие узлы перераспределяют данные на основе ключей, ранее созданных функцией Map, таким образом, чтобы все данные одного ключа лежали на одном рабочем узле.
* Reduce – параллельная обработка каждым рабочим узлом каждой группы данных по порядку следования ключей и «склейка» результатов на master node. Главный узел получает промежуточные ответы от рабочих узлов и передаёт их на свободные узлы для выполнения следующего шага. Получившийся после прохождения всех необходимых шагов результат – это и есть решение исходной задачи.

На рисунке 1 представлен принцип работы MapReduce.

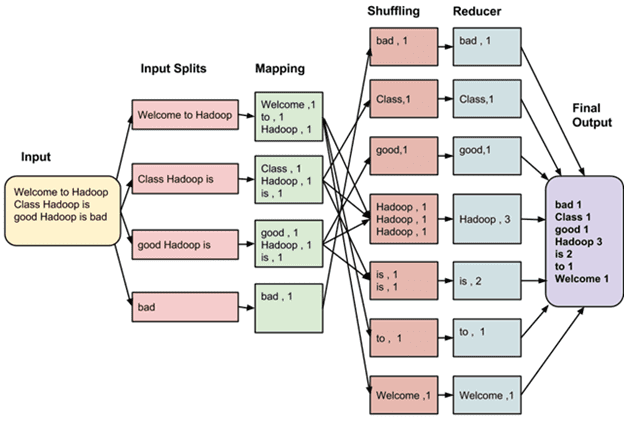


Рисунок 1 принцип работы MapReduce

Другое более детальное понимание разделения MapReduce на 5 шагов:

Подготовьте входные данные Map (): платформа MapReduce сначала определяет процессор Map, а затем назначает ему входные данные для обработки - пара ключ-значение K1 и предоставляет процессору все входные данные, связанные с ключом-значением;

* Запустите предоставленный пользователем код Map (): Map () запускается один раз для пары ключ-значение K1, чтобы сгенерировать выходные данные пары ключ-значение, заданной K2;
* Перемешайте вывод Map на процессоры Reduce: переместите ранее сгенерированные пары ключ-значение K2 в один и тот же рабочий узел в соответствии с тем, является ли «ключ» одинаковым;
* Запустите предоставленный пользователем код Reduce (): Reduce () пар ключ-значение K2 на каждом рабочем узле;
* Произведите окончательный вывод: платформа MapReduce собирает все выходные данные Reduce и сортирует их по K2 для получения окончательного результата для вывода.

4 Преимущество и недостатки MapReduce

Ключевыми достоинствами MapReduce являются следующие:

* возможность распределенного выполнения операций предварительной обработки (map) и свертки (reduce) большого объема данных. При этом функции map работают независимо друг от друга и могут выполняться параллельно на разных узлах кластера. Отметим, что на практике количество одновременно исполняемых функций map ограничивается источником входных данных и числом используемых процессоров. Аналогичным образом множество узлов производят свертку (reduce) после того, как каждый из них обработал все результаты функции map с одним конкретным значением ключа.
* быстрота обработки больших объёмов данных за счет распределения операций по вышеописанному принципу. В частности, всего за пару часов MapReduce может отсортировать целый петабайт данных.
* отказоустойчивость и оперативное восстановления после сбоев: при отказе рабочего узла, производящего операцию map или reduce, его работа автоматически передается другому рабочему узлу в случае доступности входных данных для проводимой операции.

Прежде всего, отметим, что для первой версии фреймворка MapReduce, реализованного в Apache Hadoop v1.0, были характерны следующие ограничения:

* предел масштабируемости кластера Apache Hadoop: не более 4K вычислительных узлов и около 40K параллельных заданий;
* сильная связанностьфреймворка распределенных вычислений и клиентских библиотек, реализующих распределенный алгоритм;
* наличие единичных точек отказа и невозможность использования в средах с высокими требованиями к надежности;
* проблемы версионной совместимости: необходимость единовременного обновления всех вычислительных узлов кластера при обновлении платформы Hadoop (установке новой версии или пакета обновлений).

Эти ограничения были устранены в новой версии MapReduce 2.0, выпущенной в 2012 году, за счет изменений в менеджере ресурсов (ResourceManager) и планировщике-координаторе приложений ApplicationMaster, а также появления YARN (Yet Another Resource Negotiator). Этот программный фреймворк выполнения распределенных приложений предоставляет компоненты и API для разработки распределенных приложений различных типов, обеспечивая распределение ресурсов в ответ на запросы от выполняемых приложений и ответственность за отслеживанием статуса их выполнения.

В частности, ответственность по управлению ресурсами кластера лежит на ResourceManager, а по планированию/координации жизненного цикла приложений – на ApplicationMaster. При этом каждый вычислительный узел разделен на произвольное количество контейнеров Container, содержащих предопределенное количество ресурсов: CPU, RAM и т.д., за которыми наблюдает менеджер узла (NodeManager).

Тем не менее, эти нововведения не устранили недостатки MapReduce, обусловленные архитектурными особенностями этой вычислительной модели:

* недостаточно высокая производительность – классическая технология, в частности, реализованная в ядре Apache Hadoop, обрабатывает данные ациклично в пакетном режиме. При этом функции Reduce не запустятся до завершения всех процессов Map. Все операции проходят по циклу чтение-запись с жесткого диска, что влечет задержки (latency) в обработке информации.
* ограниченность применения – продолжая вышеотмеченный недостаток, высокие задержки распределенных вычислений, приемлемые в пакетном режиме обработки, не позволяют использовать классический MapReduce для потоковой обработки в режиме реального времени, повторяющихся запросов и итеративных алгоритмов на одном и том же датасете, как в задачах машинного обучения (Machine Learning). Для решения этой проблемы, свойственной Apache Hadoop, были созданы другие Big Data фреймворки, в частности, Apache Spark и Flink.

Таким образом, достоинства и недостатки MapReduce обусловливают специфику прикладного использования этой вычислительной модели. В частности, эта технология не применяется в чистом виде в потоковых Big Data системах, где требуется оперативно обрабатывать большие объемы непрерывно поступающей информации в режиме реального времени. На практике такое встречается в платформах Internet of Things. Однако, если требование быстрой обработки данных не является критичным и бизнес-приложению подходит пакетный режим работы с данными, как, например, в ETL-системах или индексировании веб-страниц, MapReduce справится с такими задачами на отлично.