ИСПОЛЬЗОВАНИЕ MAPREDUCE И HDFS ДЛЯ ХРАНЕНИЯ БОЛЬШИХ ДАННЫХ

Полежаев П.Н., Порохненко Ю.С.
Оренбургский государственный университет

В настоящее время существует проблема хранения большого объёма данных, которые могут быть необходимы для анализа сетевого трафика, поступают с различных физических экспериментальных установок или используются для обучения и эксплуатации нейронных сетей. Для решения этой проблемы существуют модель MapReduce и файловая система Hadoop Distributed File System (HDFS).

MapReduce – это программная среда для удобного написания приложений, которые обрабатывают огромное количество данных в параллельном режиме на больших кластерах (тысячи узлов) надежным, отказоустойчивым способом [1]. MapReduce разбивает входные данные на независимые фрагменты, которые обрабатываются полностью параллельно. Работа MapReduce состоит из двух шагов, называемых map и reduce, как и основополагающие функции фреймворка. На первом шаге происходит предварительная обработка данных, один из узлов (master node) разделяет все входные данные и передаёт их другим узлам (worker node). На них происходит создание наборов «ключ-значение». На втором шаге происходит обработка полученных блоков данных, объединяются все промежуточные значения, связанные одним ключом, после чего рабочие узлы возвращают обработанные блоки главному узлу, на их основе формируется решение поставленной задачи. При этом все вызовы функций на разных узлах проходят независимо, и функции выполняются параллельно. Фреймворк заботится о планировании задач, контролирует их и повторно выполняет неудавшиеся задачи.

MapReduce состоит из одного главного планировщика JobTracker и одного подчиненного TaskTracker, по одному на каждый узел кластера. Ведущий отвечает за планирование задач на подчиненных устройствах, мониторинг и повторное выполнение неудачных задач. Планировщики на узлах выполняют задачи по указанию мастера.

Как правило, все вычисления и хранение данных происходят на одном узле, то есть фреймворк MapReduce и распределенная файловая система работают на одном и том же узле. Такая конфигурация позволяет инфраструктуре эффективно планировать задачи на узлах, где уже присутствуют данные, что приводит к очень высокой совокупной пропускной способности в кластере.

Стандартный подход к работе с объемными данными – это использование базы данных и большого количества дисков. Главная проблема таких носителей заключается в том, что со временем время поиска улучшается медленнее, чем скорость передачи. Поиск — это процесс перемещения головы диска в определенное место на диске для чтения или записи данных. Он характеризует ла-

тентность операции с диском, тогда как скорость передачи соответствует пропускной способности диска. Если в шаблоне доступа к данным преобладает поиск, потребуется больше времени для чтения или записи больших частей набора данных, чем для потоковой передачи.

С другой стороны, для обновления небольшой доли записей в базе данных чаще всего используется B-Tree (структура данных, используемая в реляционных базах данных, которая ограничена скоростью, с которой он может выполнять поиск). B-Tree менее эффективен, чем MapReduce, который использует сортировку слиянием для пересоздания базы данных, когда необходимо обновить большой объём данных. Во многих отношениях MapReduce можно рассматривать как дополнение к системе управления реляционной базой данных (РСУБД). Фреймворк хорошо подходит для решения задач, в рамках которых необходимо анализировать весь набор данных в пакетном режиме. СУРБД применяется для точечных запросов или обновлений, где набор данных проиндексирован для обеспечения небольшого времени поиска и обновления. MapReduce подходит для приложений, в которых данные записываются один раз и читаются много раз, тогда как реляционная база данных применяется для наборов данных, которые постоянно обновляются.

В таблице 1 представлена сравнительная характеристика РСУБД и MapReduce.

Таблица 0 – Сравнительная характеристика РСУБД и MapReduce

Параметр	РСУБД	MapReduce
Размер данных	Гигабайты	Петабайты
Доступ	Интерактивный и пакетный	Пакетный
Обновления	Многоразовые чтение и запись	Одноразовая запись, многоразовое чтение
Операции	ACID	-
Структура	Основана на записи	Основана на чтении
Целостность	Высокая	Низкая

Работа с нейронными сетями подразумевает частое чтение данных и редкое обновление, поэтому MapReduce подходит для решения задачи распределения данных, предназначенных для обучения и эксплуатации нейронных сетей.

Hadoop HDFS – это распределенная файловая система, предназначенная для работы на кластерах из огромного количества узлов [2]. Она имеет много общего с существующими распределенными файловыми системами, однако отличия от них значительны. HDFS отличается высокой отказоустойчивостью и предназначена для развертывания на стороннем оборудовании. HDFS обеспечивает высокопроизводительный доступ к данным приложения и подходит для приложений с большими наборами данных. Первоначально HDFS была построена как инфраструктура для проекта веб-поиска Apache Nutch, сейчас является

частью проекта Apache Hadoop Core. Hadoop разработан в рамках идеи MapReduce и считается одной из главных составляющих технологии Big Data.

Экземпляр HDFS может состоять из сотен или тысяч серверных машин, каждая из которых хранит часть данных файловой системы. Тот факт, что существует огромное количество компонентов и каждый из них имеет нетривиальную вероятность отказа, означает, что какой-то компонент HDFS всегда нефункционален, поэтому обнаружение ошибок и быстрое автоматическое восстановление являются основной архитектурной целью HDFS.

Приложения, работающие на HDFS, нуждаются в потоковом доступе к своим наборам данных. Они не являются приложениями общего назначения, которые обычно запускаются на обычных файловых системах. HDFS больше подходит для пакетной обработки, основное внимание в фреймфорке уделяется высокой пропускной способности доступа к данным. Набор стандартов для обеспечения совместимости операционных систем и переносимости прикладных программ на уровне исходного кода (POSIX) накладывает множество жестких требований, которые не нужны для приложений, предназначенных для HDFS, поэтому некоторые требования были смягчены для увеличения пропускной способности данных и обеспечения потокового доступа к данным файловой системы.

Приложения, использующие HDFS, работают с большими наборами данных. Типичный размер файла HDFS – от 100 Гб до нескольких петабайт. Таким образом, HDFS настроен на поддержку больших файлов и должен обеспечивать высокую совокупную пропускную способность, масштабироваться до сотни узлов в одном кластере, а также поддерживать десятки миллионов файлов. Вычисление, которое выполняет приложение, намного эффективнее, если оно выполняется вблизи данных, с которыми оно работает. Это особенно актуально, когда размер набора данных огромен, т.к. минимизируется перегрузка сети и увеличивается общая пропускная способность системы. В связи с этим лучше переносить вычисления на узел, где находятся данные, а не перемещать данные туда, где выполняется приложение. HDFS подходит приложений, которые находятся максимально близко к узлу, на котором находятся данные. Hadoop разработан таким образом, чтобы быть легко переносимым с одной платформы на другую. Это облегчает широкое внедрение HDFS в качестве платформы для большого набора приложений.

Hadoop не является первой распределенной системой для хранения и анализа данных, но он обладает некоторыми особенностями, которые отличает её от других похожих систем. С развитием HDFS различия между реляционными базами данных и системами Hadoop уменьшаются. Реляционные базы данных начали включать некоторые идеи из Hadoop, системы HDFS, такие как Hive, становятся более интерактивными и обладают такими функциями, как индексы и транзакции, которые делают их более похожими на традиционные РСУБД.

Основным различием между Hadoop и РСУБД является структура наборов данных, с которыми они работают. Структурированные данные, которые

имеют определенный формат, например, документы XML или таблицы базы данных, которые соответствуют определенной схеме — это область РСУБД. Менее структурированные, например, электронная таблица, в которой структура представляет собой сетку ячеек, которые могут содержать любую форму данных, как обычный текст, так и данные изображения, – это область HDFS. Hadoop хорошо работает с неструктурированными или полуструктурированными данными, поскольку он предназначен для интерпретации данных во время обработки. Это обеспечивает гибкость и позволяет избежать дорогостоящей фазы загрузки данных в РСУБД, поскольку в Hadoop это всего лишь копия файла.

HDFS имеет архитектуру master/slave. Кластер HDFS состоит из одного Namenode, главного сервера, который управляет пространством имен файловой системы и регулирует доступ к файлам клиентами, и нескольких Datanodes, которые располагаются на каждом узле в кластере и управляют хранилищем, прикрепленным к узлам, на которых они работают. HDFS предоставляет пространство имен файловой системы и позволяет сохранять пользовательские данные в файлах. Внутри файл разбивается на один или несколько блоков, и эти блоки хранятся в наборе Datanodes. Namenode выполняет операции с пространством имен файловой системы, такие как открытие, закрытие и переименование файлов и каталогов. Он также определяет отображение блоков в Datanodes. Datanodes отвечают за обслуживание запросов на чтение и запись от клиентов файловой системы. Datanodes также выполняют создание, удаление и репликацию блока по команде Namenode. На рисунке 1 представлена архитектура HDFS.

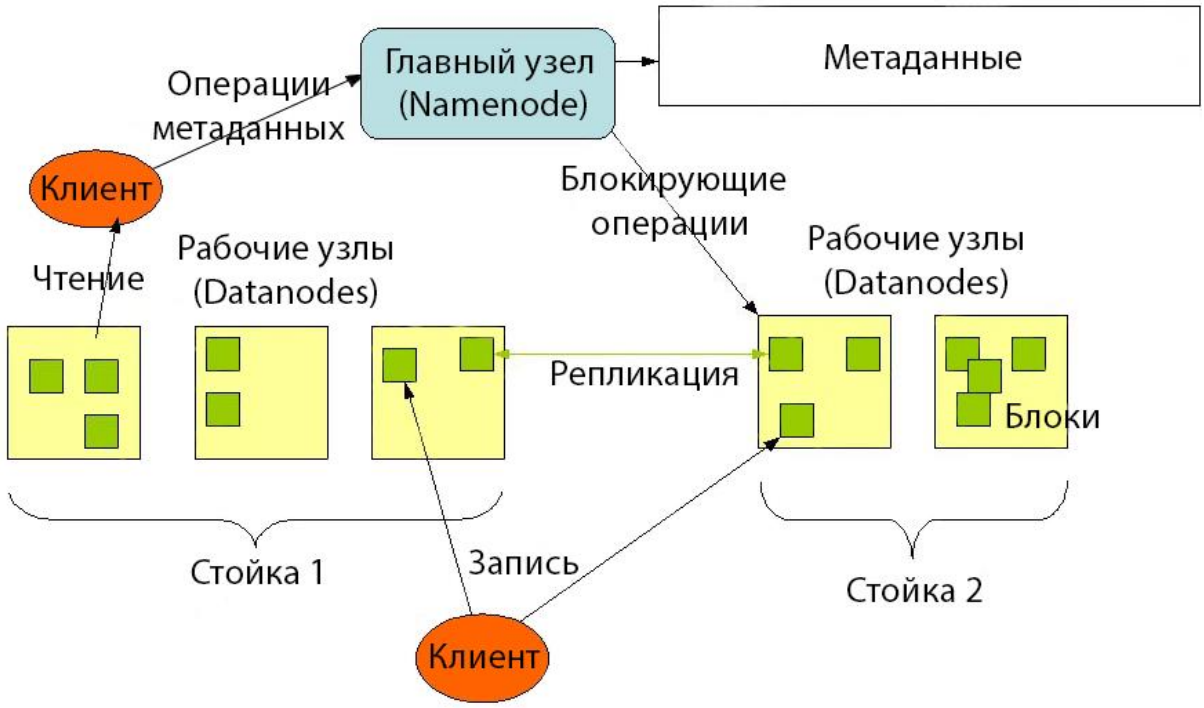

Рисунок 0 – Архитектура HDFS

Типичное развертывание HDFS состоит из выделенного компьютера, на котором работает только Namenode. Каждая из других машин в кластере запускает один экземпляр Datanode. Архитектура не исключает запуск нескольких Datanodes на одном компьютере, но в реальном развертывании такое происходит редко. Существование одного Namenode в кластере значительно упрощает архитектуру системы, т.к. он является арбитром и репозиторием для всех метаданных HDFS. Система разработана таким образом, чтобы пользовательские данные никогда не проходили через Namenode. Главный узел поддерживает пространство имен файловой системы. Любые изменения в пространстве имен файловой системы записываются в Namenode, помимо этого он хранит информацию о количестве копий файла (коэффициент репликации). Приложение может указывать количество реплик файла, которое должен поддерживать HDFS.

HDFS поддерживает традиционную иерархическую файловую организацию. Пользователь или приложение могут создавать каталоги и хранить файлы внутри этих каталогов. Иерархия пространства имен файловой системы похожа на большинство других существующих файловых систем; можно создавать и удалять файлы, перемещать файл из одного каталога в другой или переименовывать файл. В настоящий момент HDFS еще не реализует квоты пользователей или разграничение доступа и не поддерживает жесткие ссылки или программные ссылки, однако, архитектура HDFS не исключает возможности реализации этих функций.

HDFS хранит каждый файл в виде последовательности блоков; все блоки в файле, кроме последнего блока, имеют одинаковый размер. Каждый блок реплицируется для отказоустойчивости, при этом размер блока и коэффициент репликации настраиваются для каждого файла. Коэффициент репликации указывается во время создания файла и может быть изменен позже. Файлы в HDFS записываются один раз и имеют строго одного автора. Namenode принимает все решения относительно репликации блоков. Он периодически получает от каждого узла в кластере список всех блоков и ошибок (Blockreport) и ответ (Heartbeat), получение которого подразумевает, что Datanode работает правильно.

Размещение реплик имеет решающее значение для надежности и производительности HDFS. Оптимизация размещения реплик отличает HDFS от большинства других распределенных файловых систем. Размещение реплик повышает надежность, доступность и пропускную способность сети. Чтобы минимизировать глобальное потребление полосы пропускания и латентность чтения, HDFS пытается удовлетворить запрос на чтение от ближайшей к читателю реплики. Если существует реплика на той же стойке, что и считывающий узел, то эта реплика предпочтительна для удовлетворения запроса на чтение. Если кластер HDFS охватывает несколько центров обработки данных, то ре-

плика, которая находится в локальном центре обработки данных, предпочтительнее любой удаленной реплики.

Каждый Datanode периодически отправляет сообщение Heartbeat в Namenode. Нахождение узлов в разных сетях может привести к тому, что поднабор Datanodes потеряет связь с Namenode. Namenode обнаруживает это из-за отсутствия сообщения Heartbeat и отмечает не ответивший узел как мертвый и не передает к нему никаких новых запросов на чтение и запись к ним. Любые данные, зарегистрированные в мертвом Datanode, больше не доступны для HDFS. Отключение узла может привести к тому, что коэффициент репликации некоторых блоков упадет ниже их заданного значения. Namenode постоянно отслеживает, какие блоки необходимо реплицировать, и при необходимости инициирует репликацию. Потребность в повторной репликации может возникнуть по многим причинам: узел может стать недоступным, реплика может стать поврежденной или коэффициент репликации файла должен быть увеличен.

Типичный размер блока, используемый HDFS, составляет 64 МБ. Таким образом, файл HDFS разделяется на фрагменты по 64 МБ каждый, и, если возможно, каждый фрагмент будет находиться на другом Datanode.

Когда файл удаляется пользователем или приложением, он не сразу удаляется из HDFS. Вместо этого HDFS сначала переименовывает его в файл в каталоге /trash. Файл может быть восстановлен быстро, пока он остается в этом каталоге. По истечении срока его жизни Namenode удаляет файл из пространства имен HDFS. Удаление файла приводит к освобождению блоков, связанных с файлом.

Hadoop MapReduce – это фреймворк для программирования распределённых вычислений в рамках модели MapReduce. Разработчику приложения необходимо реализовать базовый обработчик. Фреймворк позволяет создавать задания как с базовыми обработчиками, так и с обработчиками, написанными без использования Java. Также, в состав дистрибутивов Hadoop входят реализации различных конкретных базовых обработчиков и свёрток, наиболее типично используемых в распределённой обработке. Модуль Hadoop MapReduce реализован поверх YARN.

Оценка производительности – это количественная основа любого исследования компьютерных систем. Для того, чтобы набор эталонов был уместным, его рабочие нагрузки должны представлять собой важные приложения целевой системы и быть достаточно разнообразным, чтобы демонстрировать диапазон поведения целевых приложений.

Для HDFS и MapReduce существует набор тестов HiBench, разработанный Intel для тестирования систем с помощью стресс-тестов [3]. HiBench состоит из набора программ Hadoop, которые помогают оценить структуру Hadoop с точки зрения скорости, пропускной способности, использования системных ресурсов и шаблонов доступа к данным. Он состоит из 10 различных рабочих нагрузок: Micro Benchmarks (сортировка, подсчёт количества слов, TeraSort, улучшенный DFSIO), веб-поиск (индексирование Nutch, PageRank), машинное

обучение (байесовская классификация, кластеризация методом K-средних) и аналитические запросы (Hive Join, Hive Aggregation).

Для тестирования кластера из 3 узлов, которые представляют собой виртуальные машины с ОС Ubuntu, расположенные в одной сети, были выбраны: подсчёт количества слов, индексирование Nutch и кластеризация методом K-средних. В кластере один узел является главным, остальные два узла – рабочие. В таблице 2 представлена информация об объёмах данных для каждого тестирования.

Таблица 2 – Информация об эксперименте

Тест	Подсчёт количества слов	Индексирование Nutch	Кластеризация методом K-средних
Объём данных	60 Гб	8.4 Гб	66 Гб

На рисунках 2-4 представлены графики использования центрального процессора, памяти и дискового ввода-вывода на временной шкале для каждой рабочей нагрузки.

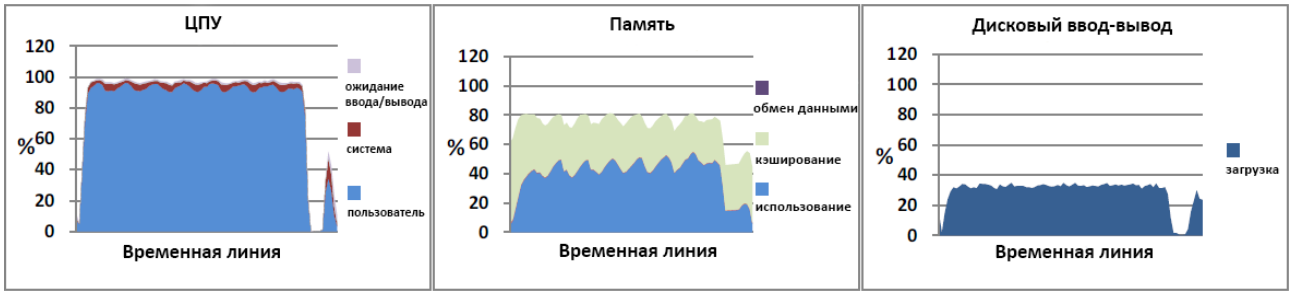

Рисунок 2 – Результаты для подсчёта количества слов

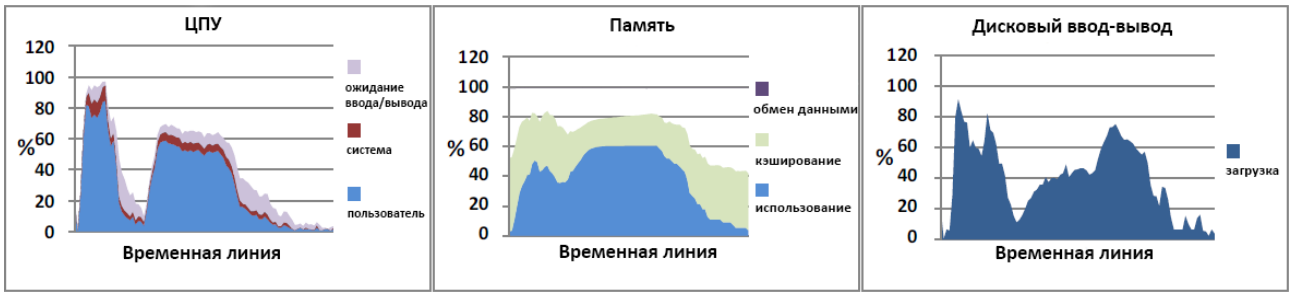

Рисунок 3 – Результаты для индексирования Nutch

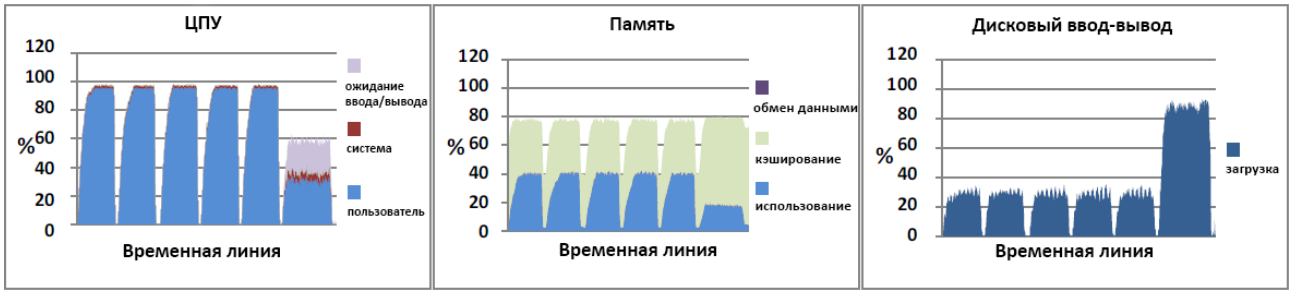

Рисунок 4 – Результаты для кластеризации методом K-средних

Рабочая нагрузка при подсчёте количества слов в основном связана с высокой загрузкой центрального процессора и незагруженным дисковым вводом-выводом, как показано на рисунке 2. Кроме того, ожидается, что поведение практически не изменится при тестировании на больших кластерах.

Как показано на рисунке 3, индексирование Nutch связано с высокой загрузкой процессора во время этапа Map, и с большим количеством операций ввода-вывода на диск (с примерно 60% использованием процессора).

Рабочая нагрузка кластеризации методом K-средних занимает большую часть времени на итерациях работы при вычислении центров кластеров. Как показано на рисунке 4, вычисление центроидов в методе K-средних в основном приводит к высокой нагрузке процессора. С другой стороны, задача кластеризации приводит к большому количеству операций ввода/вывода, в основном из-за записи результатов.

Таким образом, модель MapReduce и файловая система HDFS решают проблему хранения больших объёмов данных. Их можно эффективно применять для работы с данными, необходимыми для обучения и эксплуатации нейронных сетей.

Исследование выполнено при финансовой поддержке Правительства Оренбургской области и РФФИ (проекты №17-47-560046, №16-29-09639 и №18-07-01446), Президента Российской Федерации в рамках стипендии для молодых ученых и аспирантов (СП-2179.2015.5).

Список литературы

1. White, T. Hadoop: The definitive guide [Текст] // O'Reilly Media, Inc., 2012.- 756 с.

2. Borthakur, D. The hadoop distributed file system: Architecture and design [Текст] // Hadoop Project Website. – 2007. – Т. 11. – №. 2007. – С. 21.

3. Huang S. et al. Hibench: A representative and comprehensive hadoop benchmark suite [Текст] // Proc. ICDE Workshops. – 2010 – 325 с.