О. Д. Гобрик

# СРАВНИТЕЛЬНЫЙ ОБЗОР ТЕХНОЛОГИЙ АНАЛИЗА ДАННЫХ В ПАРАЛЛЕЛЬНЫХ СИСТЕМАХ

## Введение

Несоответствие между чрезмерным ростом объемов данных и тенденциями улучшения скорости обработки и доступа к памяти требует параллельной обработки, применяемой для обработки больших объемов данных. Как объемы данных, так и скорость обработки находятся на экспоненциально растущих траекториях, но первый вырос гораздо быстрее, чем второй. Отсюда следует, что параллельная обработка необходима для преодоления разрыва. В дополнение к обеспечению более высокой способности анализа. В статье представлен сравнительный анализ параллельной СУБД и MapReduce.

## I. Параллельная СУБД

Традиционный подход к анализу данных заключается в загрузке данных в базу данных и использование языка запросов для выполнения анализа. Параллельные СУБД были разработаны для повышения производительности систем баз данных. Она может быть определена как СУБД, которая работает на нескольких узлах. Они поддерживают стандартные реляционные таблицы, обычно разделенных на несколько узлов, и используют язык структурированных запросов (SQL).

В параллельной СУБД существует несколько различных типов архитектур:

- совместное использование памяти: несколько процессоров совместно используют пространство памяти и доступ к дискам;
- Общий диск: несколько процессоров имеют общий доступ к одному и тому же дисковому пространству, но каждый имеет свою собственную память;
- Ничего не делиться: каждый узел имеет свою память и дисковое пространство.

Важным моментом является то, что архитектуры без общего доступа перемещают только вопросы и ответы между узлами, в то время как две другие архитектуры будут перемещать данные через сеть присоединения. Основное преимущество нескольких процессоров без разделения ресурсов состоит в том, что они могут быть масштабированы до сотен и, возможно, тысяч процессоров, которые не мешают друг другу. Альтернативой параллельной СУБД, MapReduce, описана в следующем разделе.

## II. MapReduce

MapReduce - это метод обработки больших объемов распределенных данных, позволяющий использовать кластеры без общего доступа. В распределенных архитектурах кластер без общего доступа - это кластер, в котором узлы кластера не разделяют ни общее дисковое пространство, ни общий ЦП, ни общую память [1].

Работа MapReduce основана на разбиении обработки данных на две фазы: фазу отображения (map) и фазу свертки (reduce). Каждая фаза использует в качестве входных и выходных данных пары «ключ-значение». Сперва данные анализируются и некоторые вычисления завершаются. Эти задачи называются «Map» задачами. Затем данные перераспределяются по всем узлам кластера. Далее, второй набор задач выполняется параллельно каждым узлом в разделе данных, которые он получает. Эти задачи называются задачами «Reduce», принимая результаты задач карты и объединяя их вместе.

MapReduce позволяет пользователю сосредоточиться на реализации логики, необходимой для решения их конкретной проблемы. Для этого пользователь должен написать программу, которая интегрируется с библиотекой MapReduce.

При выполнении сложных алгоритмов MapReduce общий подход заключается в добавлении дополнительных циклов MapReduce, а не в попытке решить все элементы за один раз. Точная реализация MapReduce доступна только для Google. Тем не менее, парадигма была реализована и другими компаниями. Самыми популярными в коммерческом плане является Hadoop и Spark.

MapReduce имеет несколько выдающихся преимуществ [2]:

- Отказоустойчивость. Отказы являются общими для компьютерного кластера с тысячами узлов. MapReduce может справляться с мелкими сбоями, минимизировать объем потерянной работы и не требует перезапуска задания с нуля.
- Гибкость: входные данные MapReduce могут иметь любой формат вместо конкретной схемы.
- Независимость: MapReduce также зависит от системы хранения. Поддерживаемые системы хранения включают файлы, хранящиеся в распределенной файловой системе, результаты запросов к базе данных, данные, хранящиеся в Bigtable, и структурированные входные файлы.
- Масштабируемость: MapReduce может масштабироваться до тысяч процессоров.

## III. Сравнение между СУБД и MapReduce

До MapReduce параллельные СУБД использовались в качестве подходов для крупномасштабного анализа данных. По сути, все задачи MapReduce могут быть записаны как эквивалентные задачи СУБД. Но имея свои достоинства и недостатки, MapReduce дополняет СУБД, а не конкурирующую технологию. Целью СУБД является эффективность, а MapReduce - масштабируемость и отказоустойчивость. Обе системы явно улучшаются за счет привлечения силы партнера. Существенные различия между СУБД и MapReduce перечислены в таблице 1. Сравнение также выявляет некоторые противоречивые недостатки MapReduce. Далее описаны два основных.

Таблица 1 – Сравнение Параллельной СУБД и MapReduce

| | Параллельная СУБД | MapReduce |
|---|---|---|
| Схема | Есть | Не поддерживается |
| Индексация | Есть | Не поддерживается |
| Программирование (Язык) | Декларативный (SQL) | Императивные (C++, Python, ...) |
| Оптимизация | Компрессия, хранение столбцов | Не поддерживается |
| Предварительный разбор | Есть | Не поддерживается |
| Гибкость | Не поддерживается | Есть |
| Отказоустойчивость | Уровень транзакций | Уровень задач |
| Обновления | Чтение и запись многократно | Однократная запись, многократное чтение |
| Интеграция | Высокая | Низкая |
| Доступ | Интерактивный и пакетный | Пакетный |

– Неэффективность: Сравнивая результаты производительности, время загрузки MapReduce быстрее, чем СУБД, но медленнее во времени выполнения задач. Более длительное время выполнения MapReduce частично объясняется некоторыми специфическими проблемами реализации MapReduce, такими как стоимость запуска Hadoop [3]. Есть и некоторые модельные причины. Как показано в Таблице 1, СУБД выполняет синтаксический анализ во время загрузки и может также реорганизовать входные данные для определенных оптимизаций, в то время как MapReduce не будет изменять макет данных и должен выполнять весь анализ во время выполнения.

- Возможность повторного использования: MapReduce не использует схему или индексацию. Необходимо анализировать структуру входных файлов или реализовать индексацию для ускорения в программах «Map» и «Reduce». Кроме того, пользователи должны предоставлять реализации для простых и распространенных операций, таких как выбор и проекция. Однако СУБД имеет встроенную схему и индексацию, которая упрощает работу. СУБД также поддерживает множество операторов с более высоким уровнем абстракций.

## IV. Заключение

В соответствии с преимуществами MapReduce, ситуации, когда MapReduce предпочтительнее СУБД, являются:

- Наборы данных для однократного анализа. Эти наборы данных не стоят усилий по индексации и реорганизации в СУБД. Для сравнения, СУБД больше подходит для задач, требующих повторного анализа.
- Комплексный анализ. В некоторые ситуациях, когда функции «Map» слишком сложны, чтобы их можно было легко выразить в запросе SQL, такие как извлечение исходящих ссылок из коллекций документов HTML и агрегирование по целевому документу [4].
- Быстрый старт анализа. Реализации MapReduce просты в настройке, программировании и запуске.
- Ограниченная бюджетная задача. Большинство реализаций MapReduce являются открытыми, в то время как существуют редкие параллельные СУБД с открытым исходным кодом.

## V. Список литературы


1. https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
2. Уайт Т. Hadoop: Подробное руководство. — СПб.: Питер, 2013. — 672 с.
3. M. Stonebraker et al., “MapReduce and parallel DBMSs: Friends or foes?” Commun. ACM, vol. 53, no. 1, pp. 64–71, 2010
4. J. Dean and S. Ghemawat, “MapReduce: A flexible data processing tool,” Commun. ACM, vol. 53, no. 1, pp. 72–77, 2010



*Гобрик О.Д.*, магистрант ФИТиУ БГУИР, oleg.gobrik@gmail.com.

*Научный руководитель: Гуринович Алевтина Борисовна*, кандидат физ.-мат. наук, доцент, gurinovich@bsuir.by