Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем
Кафедра информационно-аналитических систем

Суманеев Артем Павлович

Анализ производительности реляционных и NoSQL СУБД

Магистерская диссертация

Научный руководитель:
д. ф.-м. н., профессор Новиков Б. А.

Рецензент:
Инженер, Открытое акционерное общество «Ланит-Терком» Чередник К. Е.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Sub-Department of Analytical Information Systems

Artem Sumaneev

Performance analysis of relational and NoSQL DBMS

Master's Thesis

Scientific supervisor:
professor Boris Novikov

Reviewer:
Engineer Kirill Cherednik

Saint-Petersburg
2017

Оглавление

Введение

В различных областях деятельности накоплено огромное количество данных, что ведет к усилению требований к их обработке и хранению, в частности к производительности систем управления данными (СУБД). Данная проблема особенно актуальна для данных, требующих глубокого анализа. В связи с этой ситуацией появляются новые подходы к построению таких систем, которые должны преодолеть недостатки существующих.

На сегодняшний день существуют два наиболее распространенных типа систем управления данными: реляционные и NoSQL СУБД, различные во многих аспектах работы. Такие кардинальные отличия в вопросах, как надежность, гибкость, согласованность данных и масштабируемость, требуют тщательного анализа различных моментов функционирования систем, в особенности производительности. Однако существующие исследования на эту тему не в полной мере охватывают вопрос производительности двух подходов, ограничиваясь сравнением операций, предоставляемых NoSQL системами. В то же время обширный спектр операций, который реализует реляционная СУБД, требует вычислений на стороне пользователя, при работе с NoSQL системой, что может привести к значительным различиям в производительности. Цель данной работы - провести исследование производительности операций СУБД этих двух типов систем.

В рамках работы было проведено сравнение реляционной и NoSQL систем управления данными на примере PostgreSQL, Apach Cassandra и Amazon DynamoDB. Основным предметом исследования является производительность операций этих систем. Результаты о производительности каждой из них были получены с помощью системы тестирования Yahoo Cloud System Bechmark(YCSB), адаптированной для нужд исследования.

Итогом работы стали данные о производительности СУБД PostgreSQL, Cassandra и DynamoDB, полученные с использованием системы тестирования YCSB. Система была расширена для выполнения тестирования расширенного набора операций над схемой данных, содержащей связи между таблицами. На основании полученных данных о производительности операций были сделаны выводы о работоспособности исследуемых СУБД.

1. Введение в предметную область

Проблемы, связанные с хранением и обработкой информации всегда имели место в связи с потребностями индустрии, бизнеса и научных исследований. Базы данных были специально разработаны для организованного доступа к данным. Они начались с навигационных баз данных, основанные на связных списках, далее была разработана реляционная модель данных, а вместе с ней реляционные БД, после - объектно-ориентированные, и, наконец, NoSQL подход.

В настоящее время двумя наиболее используемыми типами баз данных являются реляционные и NoSQL БД . Не смотря на то, что базы данных NoSQL являются относительно свежими, по сравнению с другими, они стали достаточно популярным благодаря своим способностям быстро обрабатывать несвязные и неструктурированные данные. Широкое разнообразие баз данных создает трудности для разработчиков в выборе подходящей системы. Сравнение этих двух типов баз данных необходимо - это позволит выявить их сильные и слабые стороны, а также применимость в решении той или иной задачи. Базы данных играю важную роль в приложениях, и неправильный выбор может иметь катастрофические последствия в дальнейшем, так как сложно сменить БД в процессе работы системы, тем более на другой тип.

Есть множество работ по сравнению как NoSQL баз данных между собой, так и с реляционными СУБД. В [6] рассматриваются способы поддержки таких характерных особенностей нескольких РСУБД и NoSQL, как управление параллелизмом, хранение данных, репликации и транзакционные механизмы. Авторы [13] показывают несколько классификаций систем NoSQL, сравнивают аспекты реализации и производительности нескольких NoSQL баз данных и СУБД MySQL. [11] рассматривает несколько NoSQL баз данных и общие для все NoSQL достоинства и недостатки по сравнению с РСУБД. В [1] показываются результаты тестирования производительности различных NoSQL баз данных.

В рамках курсовой работы рассматривается сравнение производительности нескольких баз данных двух типов. Главной задачей было провести анализ производительности NoSQL-системы на сложных операциях, встроенных в РСУБД (таких как JOIN и агрегирующие функции), относительно реляционной системы управления данными. Зачастую производители NoSQL систем предоставляют лишь самый базовый набор операций над БД, оставляя пользователю реализацию более сложной обработки данных, что может существенно снизить скорость выполнения. В то же время в РСУБ аналогичные результаты можно получить одним или несколькими запросами к системе, выполнение которых оптимизировано разработчиками системы. Основным результатом работы является анализ производительности каждой из типов систем в различных аспектах

функционирования.

В работе представлено сравнение производительности реляционной базы данных и NoSQL-системы при работе с нетривиальной схемой данных, содержащей несколько связанных внешними ключами таблиц. Основным результатом является анализ производительности СУБД, данные о которой получены с помощью тестирующей системы, в различных аспектах функционирования.

Работа организована следующим образом. В Главе 1 дается общее представление предметной области, краткий обзор концепции реляционных СУБД и NoSQL баз данных, в частности PostgreSQL, Apache Cassandra и Amazon DynamoDB. Далее представлен способ тестирования с помощью Yahoo Cloud System Benchmark. В Главе 2 рассматриваются схема данных, метод тестирования и сравнение результатов тестирования исследуемых СУБД. В Заключении подводится итог проведенной работы и предлагаются направления дальнейшего анализа.

1.1. Реляционные базы данных

Базы данных определяются как коллекция хранимых данных, используемая приложениями [7, стр. 11]. Хотя, используя термин «база данных», мы, зачастую, подразумеваем систему целиком, сам термин определяет только коллекции и данные. Система, управляющая данными, транзакциями и другими использования БД, называется Система Управления Данными (СУБД). Далее последует описание двух типов баз данных, сравниваемых в данной работе.

Реляционные базы данных основаны на реляционной модели и теории множеств, в которой все данные представляются как n-арное отношение, которое, в свою очередь, представляется как подмножество n-арного декартова произведения n множеств. Отношение (таблица) состоит из множества кортежей (записей), атрибуты которых соответствуют столбцам. Такая модель данных очень точная и хорошо структурирована.

Реляционная база данных гарантирует высокую надежность транзакций благодаря полной поддержке четырех свойств ACID:

- Атомарность: если какая-либо часть транзакции не выполняется, то не выполняется транзакция целиком.
- Согласованность: если база данных находилась в согласованном состоянии до выполнения транзакции, то после выполнения она также будет находиться в согласованном состоянии.
- Изолированность: множество транзакций, выполняемых одновременно, не влияют на ход работы друг друга. Иными словами, параллельные транзакции должны быть сериализуемы.

- Долговечность: изменения, совершенные зафиксированной транзакцией, будут оставаться в системе не смотря на какие-либо сбои.

Из множества реляционных систем управления данными, в работе предлагается анализ производительности РСУБД PostgreSQL.

PostgreSQL [12] является объектно-реляционной системой управления базами данных. Система разрабатывается более 15 лет и имеет проверенную архитектуру, которая заработала хорошую репутацию надежности, целостности данных и точности. Также она целиком поддерживает ACID и стандарт ANSI-SQL:2008.

PostgreSQL содержит в себе нетривиальные возможности, такие как управление конкурентным доступом с помощью многоверсионности (MVCC), возврат к состоянию в определенный момент времени (Point-in-time recoverty), пространство таблиц, асинхронные репликации, внутренние транзакции (точки сохранения), резервное копирование во время исполнения, сложные планировщик и оптимизатор запросов и другие. Все эти функции позволяют PostgreSQL быть хорошей альтернативой NoSQL-системам в плане масштабируемости, сохраняя возможность сложных глубоких аналитических запросов посредством SQL.

1.2. NoSQL базы данных

В последние годы было разработано больше число новых систем [4, 10, 2], предоставляющих хорошее горизонтальное масштабирование для простых операций чтения/записи над базами данных, распределенных на множестве серверов. В отличии от них, традиционные базы данных дают меньше возможности к масштабированию.

Множество новых систем определяются термином «NoSQL»-хранилища данных. Термин «NoSQL», что расшифровывается как «Not Only SQL» или «Not Relational», до конца не определен. Обычно такие системы удовлетворяют следующим признакам:

1. Наличие средств распределения нагрузки на множество серверов.
2. Возможность распределение данных на множество серверов.
3. Простой протокол вызова операций (в сравнении с SQL).
4. Более слабая модель параллелизма, чем ACID-транзакции.
5. Эффективное использование распределенных индексов и оперативной памяти для хранения данных.
6. Отсутствие фиксированной схемы данных.

NoSQL-системы обычно не удовлетворяют свойствам ACID-транзакций: допускается согласованность в конечном счете. Предлагается модель «BASE» (Basically Available, Soft state, Eventually consistent, согласованность в конечном счете) в противоположность ACID. Идея состоит в том, что, отказавшись от ограничений ACID, можно добиться гораздо лучшей производительности и масштабируемости. Большинство систем разнятся в степени отказа от ACID.

Сторонники NoSQL часто ссылаются на CAP-теорему, которая утверждает, что система может удовлетворять только двум из трех следующих свойств:

- Согласованность - данные всегда одинаковы на всех репликах,
- Доступность - данные всегда доступны пользователю,
- Устойчивость к разделению - система базы данных продолжает корректную работу не смотря на отказ сети или узлов.

В NoSQL системах обычно опускается согласованность, но допущения могут быть сложнее.

Зачастую модели данных в NoSQL-системах разбиваются на следующие категории [9]:

- Хранилища типа *«ключ-значение»* (*Key-value stores*): хранят значения и идентификатор для поиска, основанный на заданном ключе.
- *Документно-ориентированные* хранилища (*Document stores*): система хранит данные в форме документов, которые индексируются.
- Хранилища с *расширяемой записью* (*Extensible records stores*): записи в таких хранилищах могут быть распределены вертикально и горизонтально по узлам.

Данная работа сосредотачивается на NoSQL-системах типа «ключ-значение» , в частности для сравнения были выбрана Apache Cassandra и DynamoDB.

Модель данных и функциональность **Apache Cassandra** [3] имеет сходство с другими масштабируемыми хранилищами. Обновления и группировки столбцов кэшируются в оперативной памяти, после чего сбрасываются на диск. Присутствует поддержка распределения данных и механизмы репликации, автоматические обнаружения отказов узлов и восстановление. Однако модель параллелизма в Cassandra слабее, чем в других системах, в следствии отсутствия механизмов блокировки и асинхронного обновление реплик. Для обработки данных Cassandra поддерживает CQL - Cassandra Query Language, основанный на SQL.

Amazon DynamoDB [8] является продуктом компании Amazon, основа которого – облачное хранилище данных типа «ключ-значение», где клиент оплачивает трафик, а не объем данных. Дополнительно компания предоставляет возможность развернуть

базу данных локально. Схемой хранения данных в этой системе является запись, содержащая ключ и некоторый набор именованных аттрибутов любой размерности. Среди особенностей DynamoDB можно выделить автоматическое кэширование таблиц в оперативной памяти (Amazon DynamoDB Accelerator, DAX), масштабирование в облаке по заданной пропускной способности и возможность создания приложений-триггеров, реагирующих на изменения данных.

Важным фактором при сравнении производительности нескольких систем является выбор тестирующей системы. В данной работе в качестве такой системы был выбран Yahoo Cloud System Benchmark.

1.3. Yahoo Cloud System Benchmark

Yahoo Cloud System Benchmark (YCSB) [5] - фреймворк для тестирования производительности, созданный для сравнения системы PNUTS от Yahoo с другими системами. Основной целью YCSB является выявления сильных и слабых сторон СУБД. Для тестирования производительности системы был выбран именно этот инструмент в связи с прозрачностью тестирования (благодаря открытым исходным кодам) и возможностью расширения и переработки под собственные нужды.

Система тестирования содержит набор готовых загрузок (workloads), покрывающих основные аспекты функционирования (чтение, запись, сканирование) и поддерживает созданные пользователем загрузки. Важной особенностью фреймворка является расширяемость: генератор загрузок облегчает работу по созданию новых типов загрузок, дополнительно, пользователь может достаточно легко добавить новую систему баз данных к фреймворку.

Основной задачей данной работы является сравнение производительности СУБД на нетривиальной схеме данных.

1.3.1. Схема данных

Тестирующая система позволяет работать с простой схемой данных, содержащей одну таблицу с некоторым числом столбцов. Использование более сложной схемы затруднено тем, что способ работы с базой данных у NoSQL и реляционных систем сильно различаются. Также существуют различия в интерфейсе между самими NoSQL системами.

Так как в работе наложены ограничения на анализируемые операции и используемые СУБД, есть возможность переиспользовать систему для работы со сложной схемой данных, состоящей из нескольких таблиц. Подробнее схема данных и адаптация тестирующей системы представлены во второй главе.

1.3.2. Workloads

В YCSB загрузкой является набор операций (их общее число и процентное отношение), размер обрабатываемых данных, запрос на распределенность и т.д., что может быть использовано для симулирования определенных ситуаций и поведения системы.

Предлагаемые разработчиками загрузки используют одну таблицу с N полями, среди которых есть ключевое. Значения каждого поля - случайная строка длины L. Количество полей и их размер можно настраивать.

Операции над хранилищем выбираются случайно из следующих:

- Вставка: добавление новой записи.
- Обновление: обновление значения записи.
- Чтение: чтение случайной записи или всех.
- Сканирование: упорядоченное чтение случайного числа записей, начиная с некоторого ключевого значения.

Общее число операций и их процентное соотношение - настраиваемые параметры, от которых будет зависеть направление тестирования.

Случайность выбора записи задается одним из распределений: равномерное, распределение Парето (с предпочтением некоторых записей перед большинством остальных), позднее (подобно Парето, но наиболее вероятными записями становятся более поздние) и мультномиальное.

Основным ограничением YCSB в плане тестирования является использование строго заданной единственной таблицы, сужающим набор возможных операций до вышеперечисленных. Для того, чтобы протестировать производительность систем на более сложных операциях, присущих реляционным системам, необходимо расширить возможности системы к использованию сложной схемы в плане тестирования. Также необходимо адаптировать эту схему для работы NoSQL-системы с ней вследствие существенной разницы в предоставляемых операций - NoSQL системы зачастую предоставляют лишь набор простейших запросов, перекладывая более продвинутую обработку на пользователя системы.

1.4. Выполнение запросов и измерения

Помимо задания размеров исследуемых данных и выполняемых операций, система тестирования предоставляет различные механизмы управления процессом выполнения запросов. При запуске тестирования можно указать следующие параметры:

- Число потоков – количество «клиетских» потоков, выполняющих тестируемые операции. Большее число поток увеличивает нагрузку на исследуемую систему
- Число операций в секунду – система будет стараться достичь заданное число операций в секунду, распределенное среди потоков
- Интервал измерений среднего отклика системы – система с заданной периодичностью замеряет среднее время выполнения операции

В данной работе использовался следующие настройки измерений: 10 потоков, 100 операций в секунду, измерение среднего времени выполнения каждые 2000 секунд. Данные параметры позволяют исследовать производительность систем при заданной нагрузке во времени.

В следующей главе, будут представлены схема данных, используемая для тестирования производительности, способ адаптации этой схемы в системе тестирования YCSB и анализ полученных результатов.

2. Тестирование

Для получения данных о скорости выполнения системами различных операций использовалась система тестирования Yahoo Cloud System Benchmark. Исходно YCSB позволяет протестировать простейшие операции на простой схеме, состоящей из одной таблицы и набора столбцов. Задачей данной работы было сравнение производительности СУБД на более сложных операциях, требующих нетривиальную схему из нескольких связанных таблиц, что потребовало адаптации системы тестирования.

2.1. Используемая схема данных

Для исследования производительности дополнительных операций (например, JOIN), присущих реляционным СУБД, была использована схема данных, представленная на Рис. 1. В данной схеме присутствуют несколько таблиц, содержащих внешние ключи.

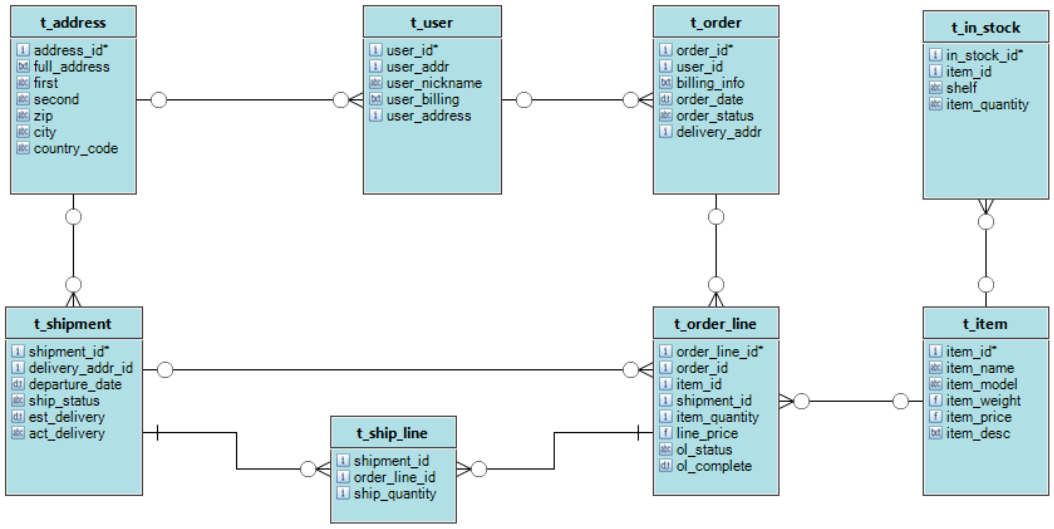


Рис. 1: Схема данных

Однако, исходная реализация системы тестирования не поддерживает работу с несколькими таблицами одновременно, также у нее ограничен набор выполняемых над таблицей операций. Система YCSB была адаптирована для нужд данной работы.

2.2. Адаптация Yahoo Cloud System Benchmark

К существующим в системе операциям было необходимо добавить две дополнительные: JOIN (1) и GROUP BY (2) для агрегатных функций. Реализация данного требования осложнено тем, что интерфейс взаимодействия с базой данных у различных СУБД разная. При этом в реляционных системах требуемые данные

можно получить с помощью одних только запросов к СУБД, а NoSQL-системы перекладывают окончательную обработку данных на пользователя, оставляя за собой лишь базовые операции над базой данных.

Пример 1: Оперция JOIN

```
SELECT *
FROM t_user
JOIN t_address
ON t_user_id = t_address_id
```

Пример 2: Оперция GROUP BY

```
SELECT COUNT(*)
FROM t_user
GROUP BY user_address
```

Для этого, во-первых, системе было необходимо добавить данные о внешних ключах и размерах каждой из используемых таблиц. Это было реализовано загрузкой информации о таблицах во время исполнения, что позволяет сохранить некоторую гибкость в работе системе.

В РСУБД схема использовалась как она есть, для NoSQL-систем схема была сведена к нескольким несвязным между собой таблицам, содержащим все поля, кроме вспомогательных (например, которые являются внешним ключом), из объеденных таблиц. Например, две таблицы *t_user* и *t_address* были преобразованы в одну таблицу, содержащую данные о пользователе и адресе. Основные операции чтения, записи и обновления над исходной таблицей теперь выполняются над той частью преобразованной таблицы, которая соответствует исходной. При загрузке сгенерированных данных в базы данных обоих систем соблюдалось ограничение, наложенное внешними ключами и размерами исходных таблиц.

Дополнительные операции JOIN и GROUP BY с применением агрегатной функции были реализованы для РСУБД Postgresql в виде единственного запроса со случайным выбором ключа, по которому будет вестись поиск записи. NoSQL-системы Cassandra и DynamoDB не поддерживают такие операции, для получения данных, аналогичных данным по запросу к реляционной БД, была реализована последующая обработка «сырых» данных, полученных простой операцией чтения, на стороне пользователя. Операция JOIN является простым чтением частей таблицы, соответствующих запрошенным, так как само объединение записей было произведено при создании таблицы. Для операции GROUP BY с агрегатной функцией COUNT производится подсчет уникальных записей в части таблицы, соответствующей исходно запрошенной.

Таким образом были реализованы дополнительные операции над реляционной и

NoSQL базами данных, производительность которых может быть измерена средствами системы тестирования. При этом соответствующие операции в двух типах систем могу быть сравнены, так как данные, полученные применением этих операций, являются одинаковыми.

После реализации требуемых операций в системе тестирования YCSB было проведено тестирование двух СУБД - реляционной Postgresql и NoSQL Cassandra - и проведен сравнительный анализ производительности этих систем.

2.3. Результаты тестирования

Тестирование проводилось на следующей конфигурации: одна машина (4х ядерный 3.4GHz Intel Core i5 процессор, 4GB RAM, диск размера 150 GB) для запуска одной из систем. Над базами данных выполнялись следующие операции: вставка записи, чтение записи, обновление записи, чтение записи из объединенных таблиц (JOIN) и выполнение группировки записей агрегатной функции COUNT (GROUP BY). Операции выполнялись над базами данных следующих размеров - малым, средним и большим.

Для тестирования операций использовались объемы данных, представленные в Таблице 1:

Объем	Число записей				Суммарный объем
	t_user	t_shipment	t_in_stock	t_address	
малый	200	200	1k	50	7 МБ
средний	20k	20k	100k	5k	700 МБ
большой	200k	200k	1M	50k	7 ГБ

Таблица 1: Объемы данных

Для замеров производительности над системой выполнялось 100 операций в секунду при общем объеме в 1000 операций. Измерялась производительность следующих операций:

1. Вставка (load) - вставка записи в базу данных. Результаты представлены для таблицы *t_shipment*
2. Обновление (update) - перезапись единичной полей существующей записи. Результаты представлены для таблицы *t_in_stock*
3. Чтение (read) - чтение единичной записи. Результаты представлены для таблицы *t_user*
4. Соединение (join) - чтение единичной записи из соединения по внешнему ключу двух таблиц. Результаты представлены для таблицы *t_user* и *t_address*

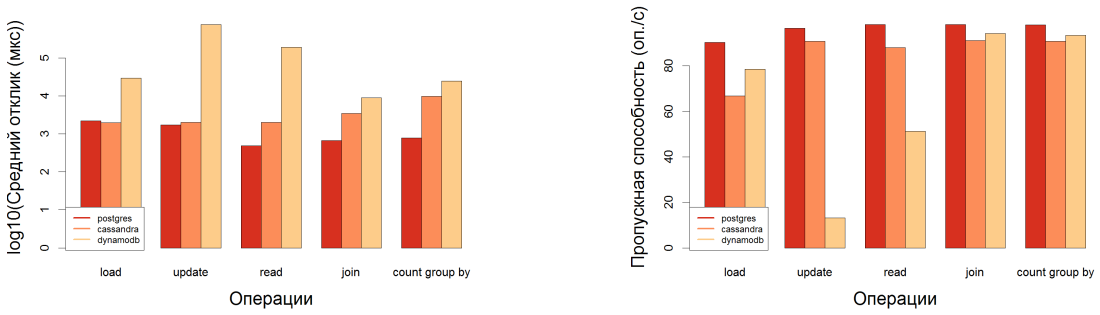
5. Группировка (count group by) - группировка записей в таблице по ключу с подсчетом числа вхождений уникальной записи. Результаты представлены для таблицы t_user

Для анализа производительностей измерялось время отклика систем на запрос - время между началом запроса и получением ответа. Сравнивались два вида показателей - средний отклик по всем выполненным операциям и детализированный за каждые 2 секунды работы.

2.3.1. Среднее время отклика СУБД

На графике 2 представлены измерения среднего времени выполнения операций (2a) и пропускной способности систем (заданное значение равно 100). Значения вертикальной оси соответствуют десятичному логарифму среднего отклика и числу операций в секунду соответственно. Таким же образом описываются графики 3 и 4 для среднего и большого объемов.

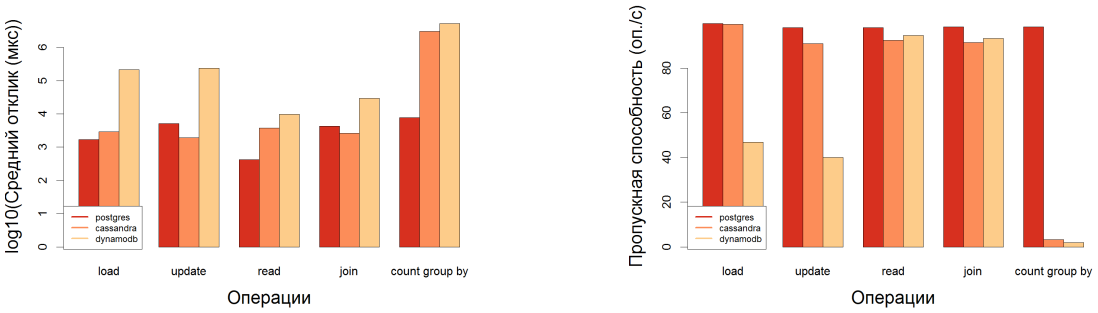
В дополнение к графическому анализу для сравнения результатов тестирования выборки средних откликов в зависимости от времени были исследованы с помощью непараметрического критерия однородности Манна-Уитни. Результаты применения критерия подтвердили сделанные выводы.



(a) Средний отклик

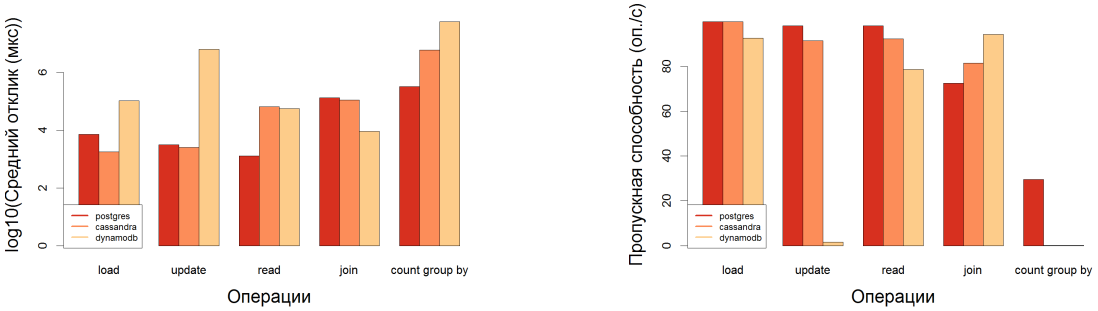
(b) Пропускная способность

Рис. 2: Результаты для малого объема



(a) Средний отклик (b) Пропускная способность

Рис. 3: Результаты для среднего объема



(a) Средний отклик (b) Пропускная способность

Рис. 4: Результаты для большого объема

2.3.2. Операция вставки записи

На графиках 5 и 6 представлен время отклика систем в зависимости от времени для операции вставки при среднем и большом объемах данных. Значения вертикальной оси соответствуют десятичному логарифму времени выполнения операции. Последующие подобные графики построены по тому же принципу.

Из графиков видно, что наибольшее время выполнения операции у DynamoDB при всех объемах данных. При среднем объеме данных Postgres показывает лучшее время выполнения, чем Cassandra. На большом объеме данных быстрее работает Cassandra.

Postgres и Cassandra удержтвают заданную пропускную способность при каждом объеме данных, DynamoDB – на большом.

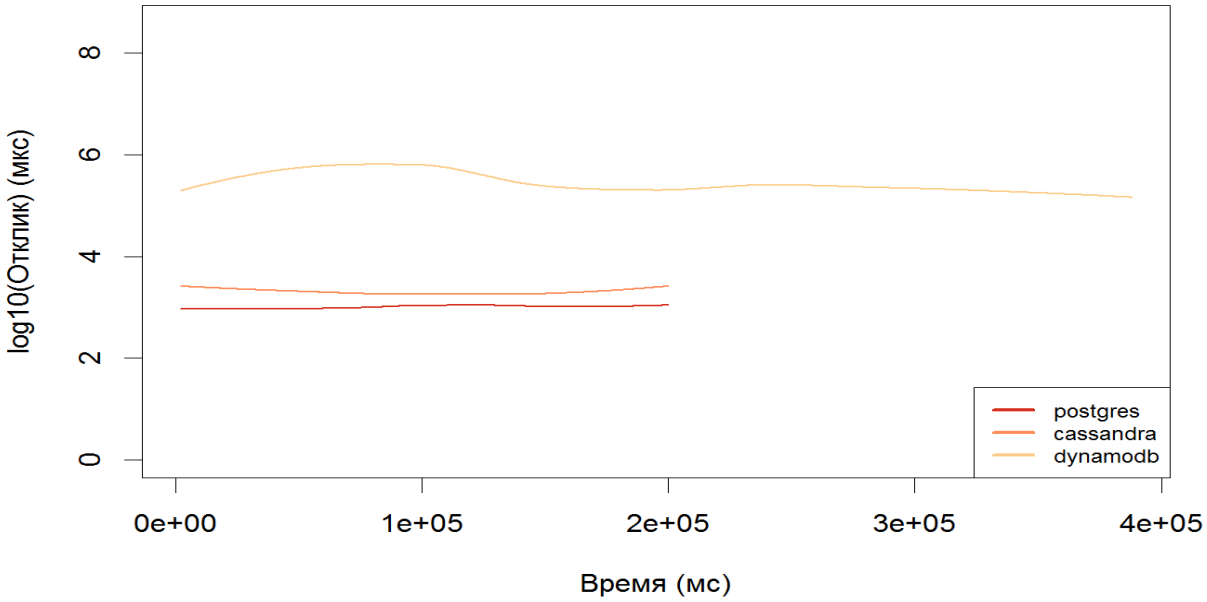


Рис. 5: Отклик операции load от времени при среднем объеме данных

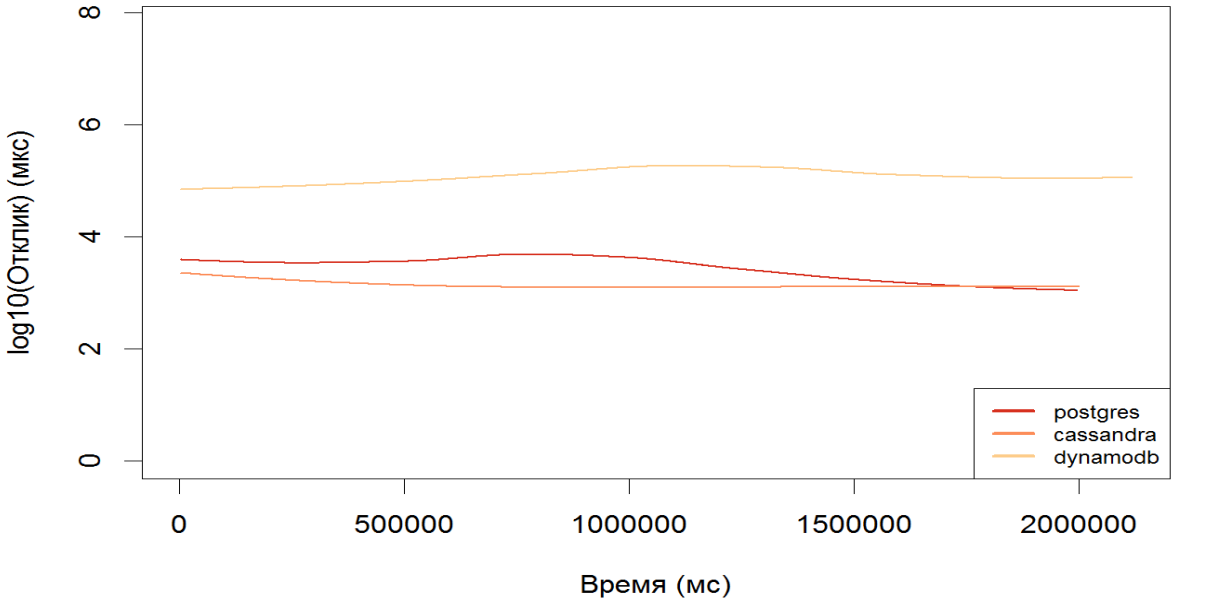


Рис. 6: Отклик операции load от времени при большом объеме данных

2.3.3. Операция чтения записи

Графики результатов для операции чтения записи представлены на рисунках 7 - 9. Из них следует, что с данной операцией Postgres справляется лучше NoSQL систем. DynamoDB значительно проигрывает в производительности Cassandra и показывает равный результат при большом объеме.

Postgres и Cassandra удержтвают заданную пропускную способность при каждом объеме данных, DynamoDB – на большом.

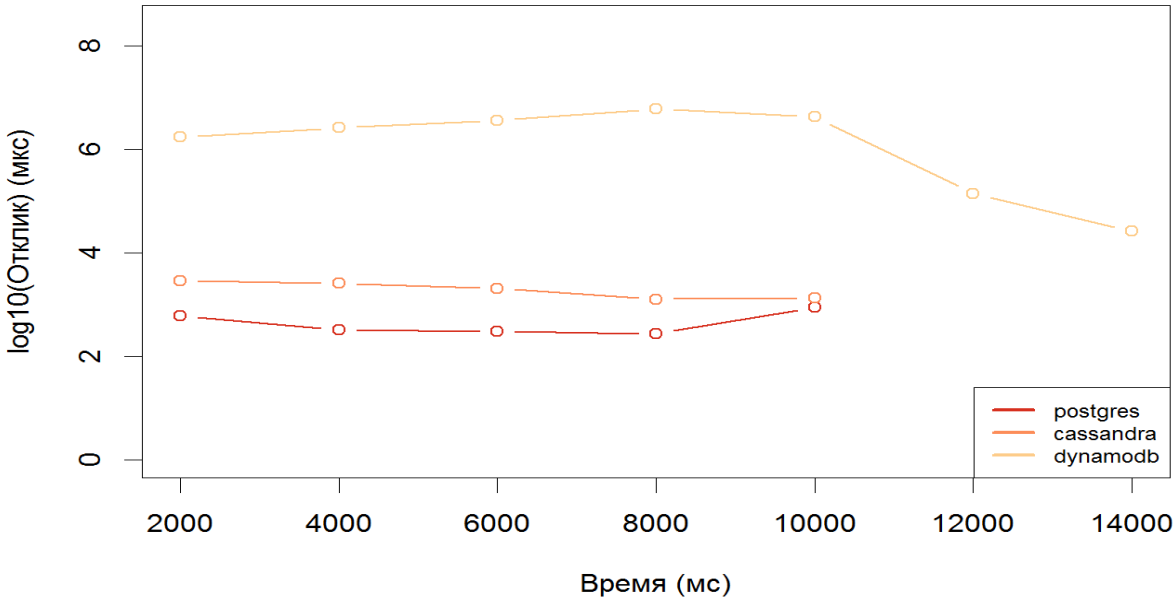


Рис. 7: Отклик операции read от времени при малом объеме данных

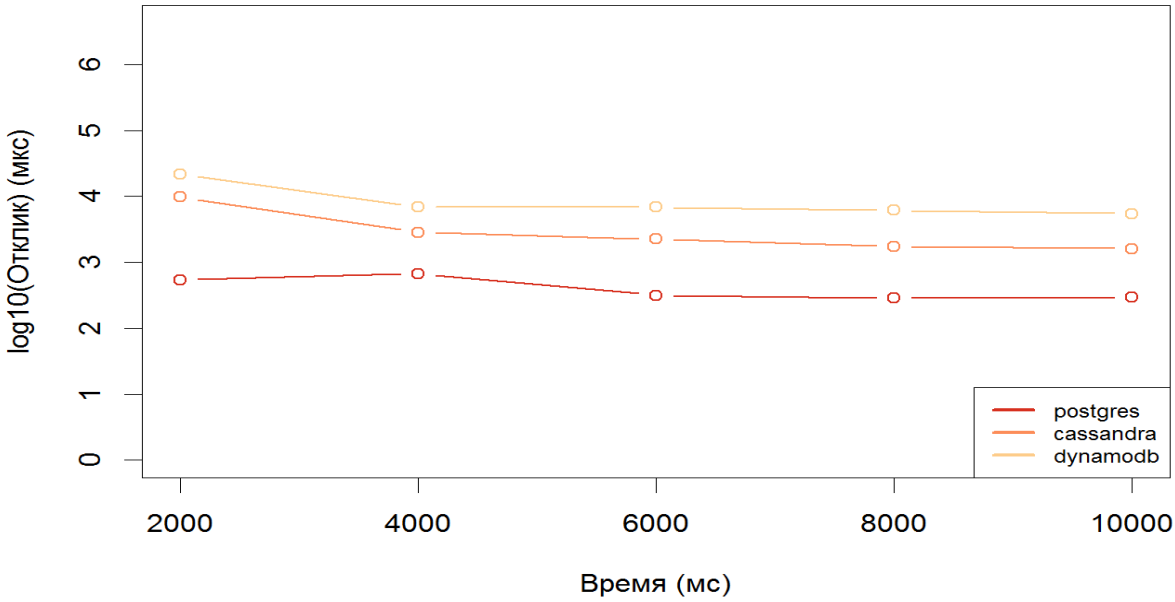


Рис. 8: Отклик операции read от времени при среднем объеме данных

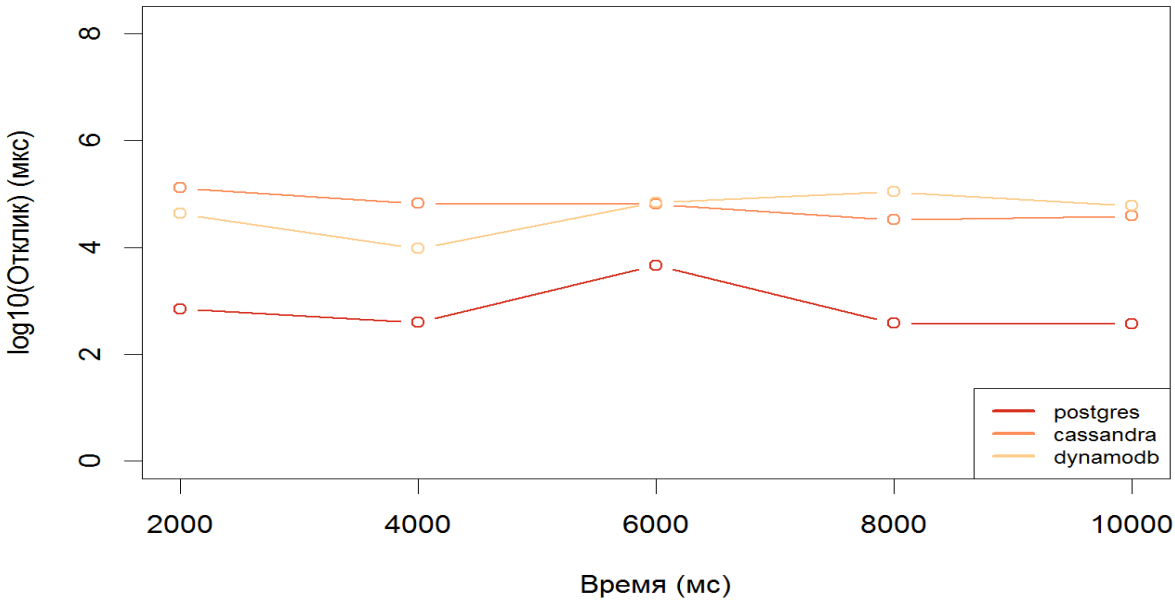


Рис. 9: Отклик операции read от времени при большом объеме данных

2.3.4. Операция обновления записи

Результаты производительности операции Update представлены на графиках 10 - 12. Согласно ним производительность DynamoDB ниже остальных систем. Из результатов применения критерия однородности следует, что производительности Postgres и Cassandra равны.

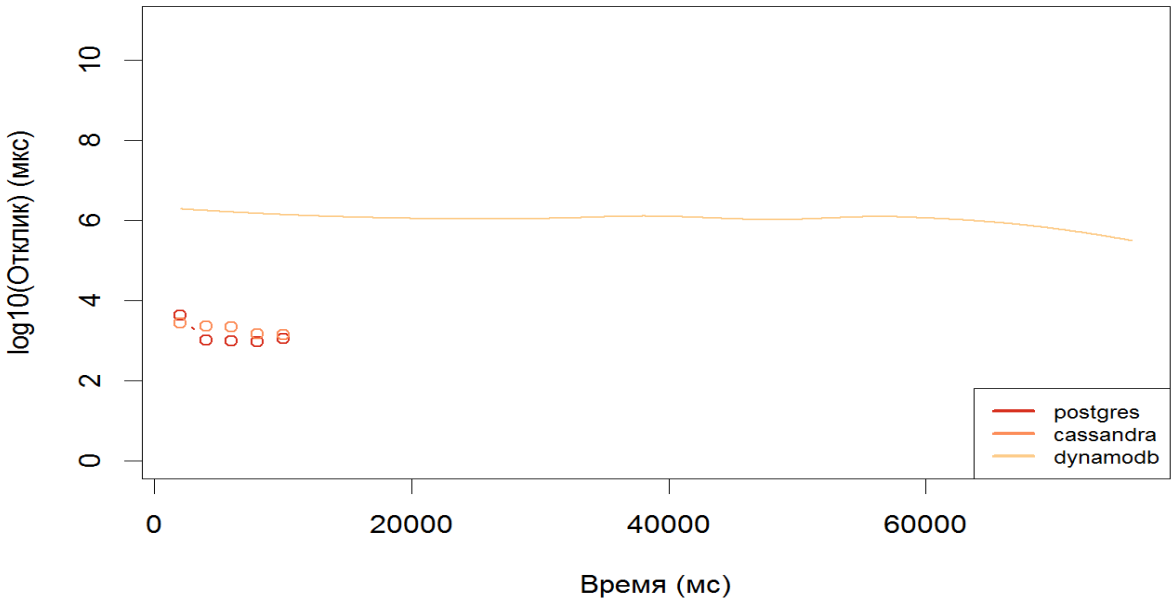


Рис. 10: Отклик операции update от времени при малом объеме данных

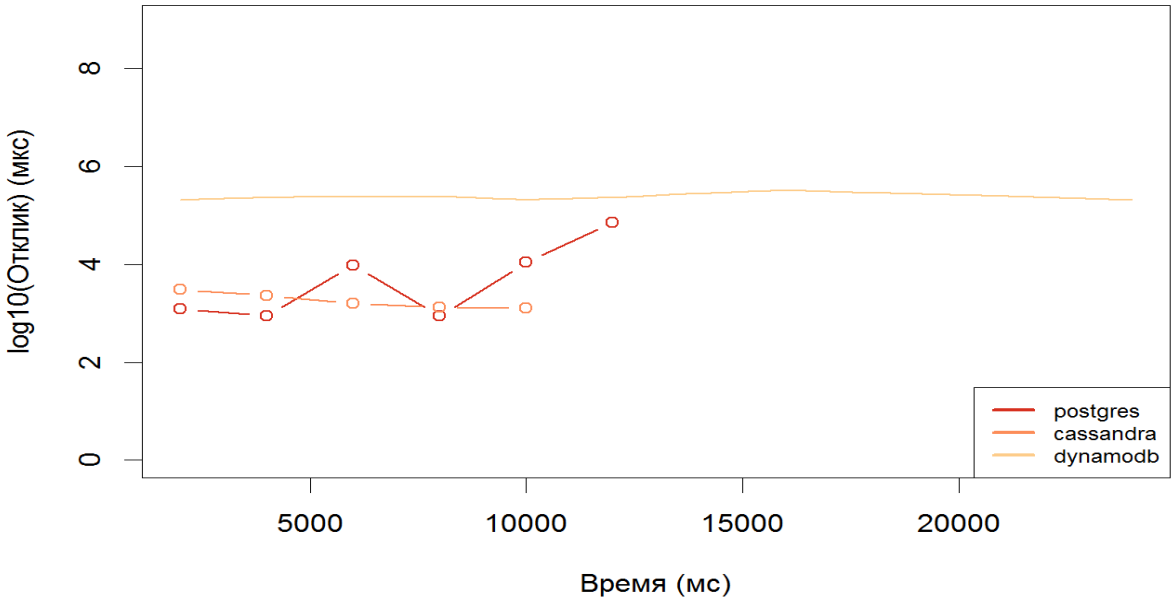


Рис. 11: Отклик операции update от времени при среднем объеме данных

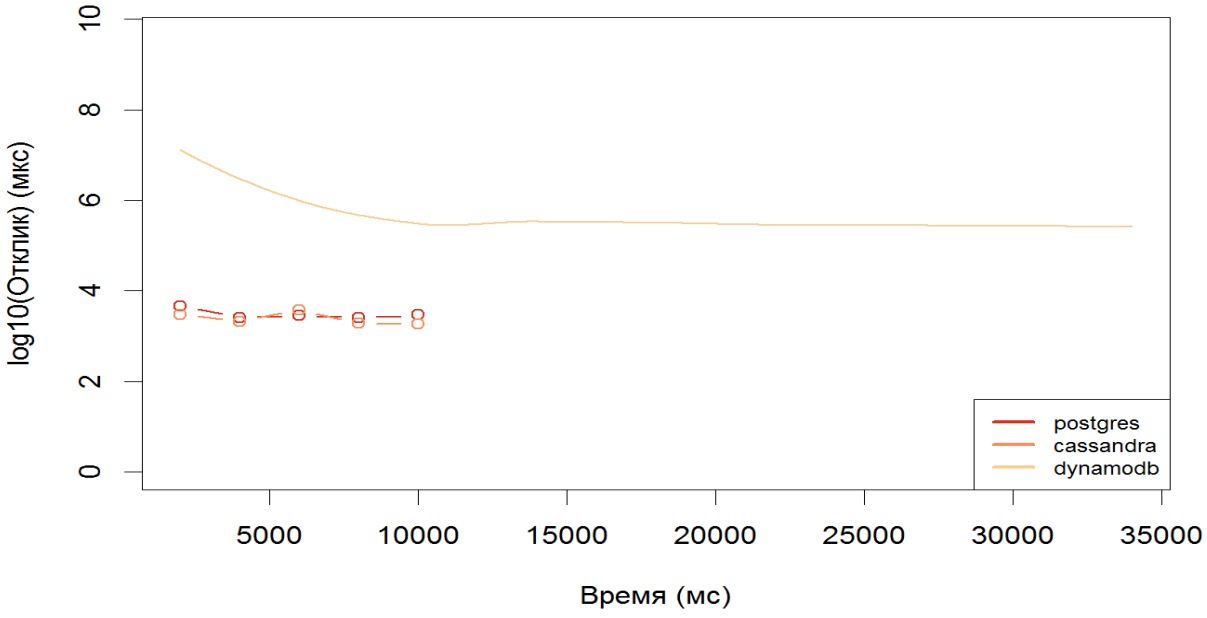


Рис. 12: Отклик операции update от времени при большом объеме данных

2.3.5. Операция соединения таблиц

Данные для операции Join представлены на рисунках 13 - 15. На малом объеме данных лучшие результаты производительности показывает Postgres, производительность Cassandra выше, чем у DynamoDB. При среднем объеме производительность Cassandra выше остальных систем, Postgres демонстрирует лучшую скорость выполнения запроса, чем DynamoDB. На большом объеме данных лучшие показатели производительности у DynamoDB, Postgres проигрывает Cassandra.

Пропускная способность Postgres и Cassandra снижаются с увеличением размера данных, DynamoDB удерживает этот показатель.

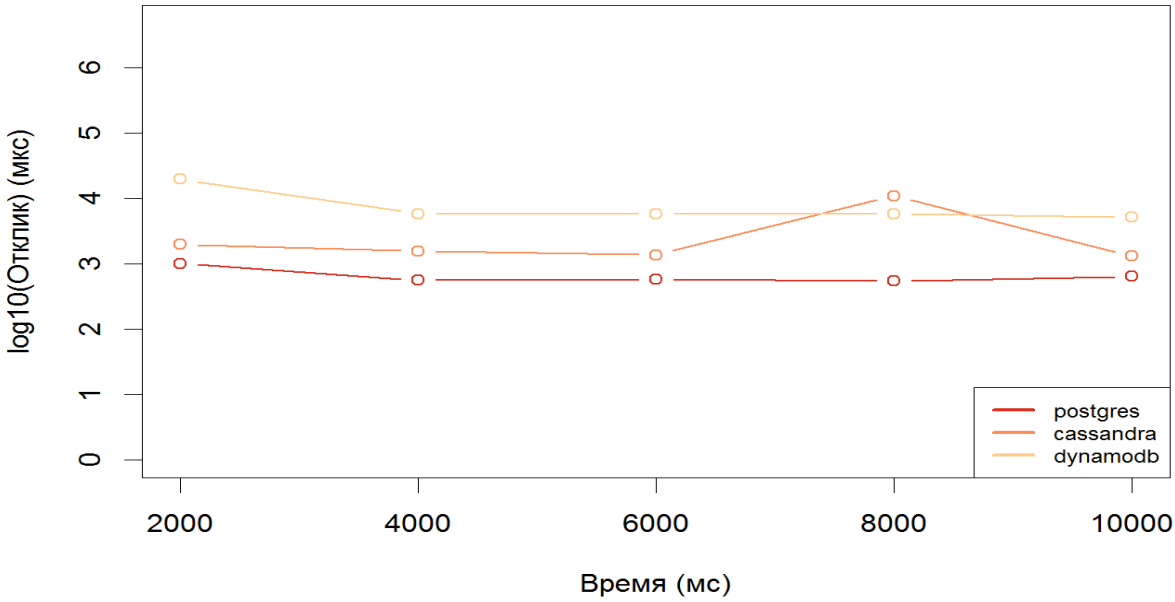


Рис. 13: Отклик операции join от времени при малом объеме данных

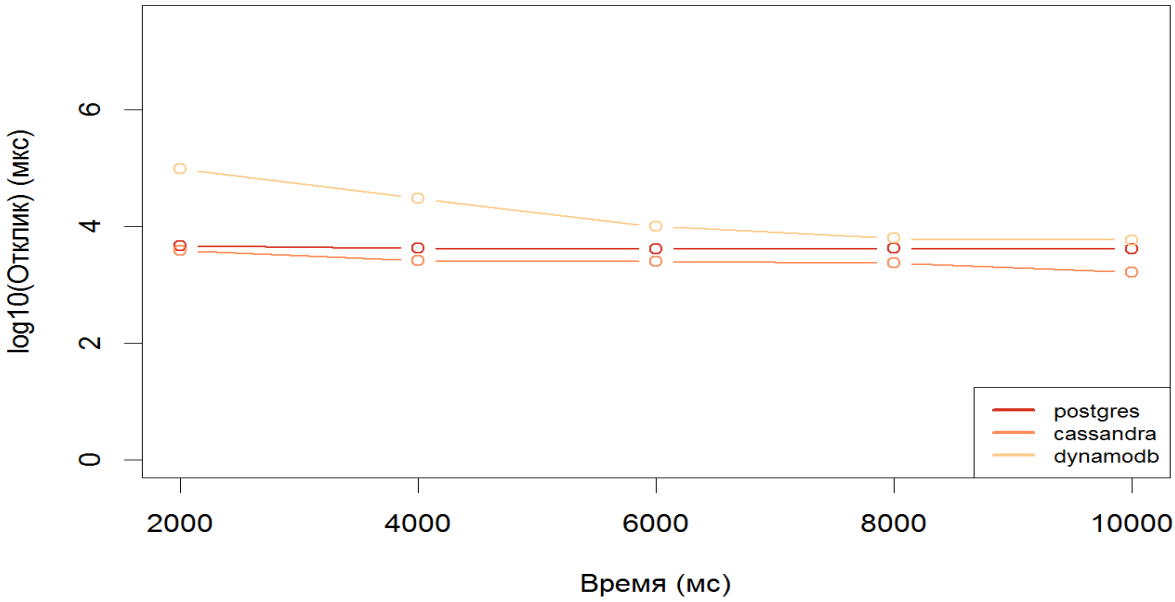


Рис. 14: Отклик операции join от времени при среднем объеме данных

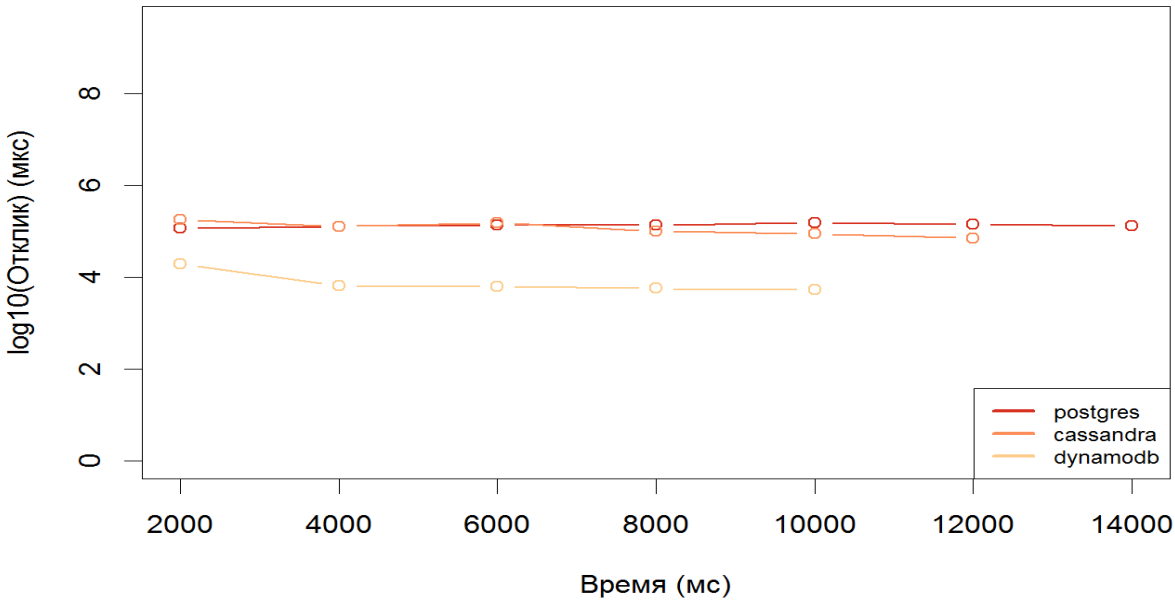


Рис. 15: Отклик операции join от времени при большом объеме данных

2.3.6. Операция подсчета с группировкой

Графики 16 - 15 представляют результаты тестирования операции Count Group By. В связи с тем, что данная операция выполняется Postgres на стороне сервера, а для NoSQL она реализована на стороне пользователя, Postgres демонстрирует лучшую производительность, чем NoSQL системы. В свою очередь, производительность Cassandra выше, чем производительность DynamoDB.

Пропускная способность Postgres сохраняется при малом и среднем объемах данных и снижается при большом объеме.

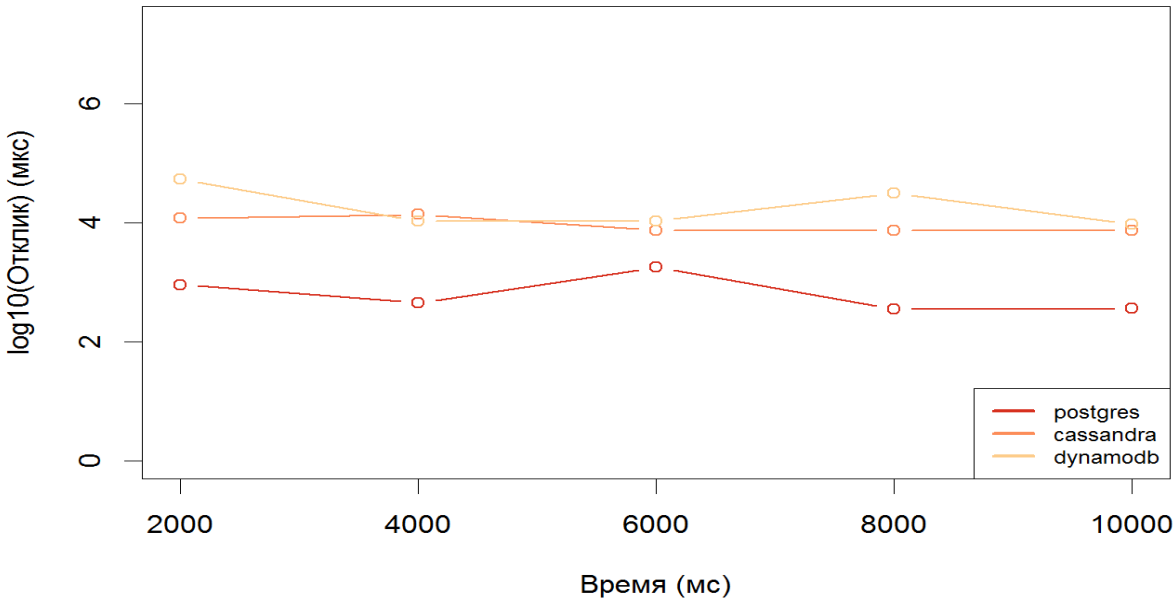


Рис. 16: Отклик операции count group by от времени при малом объеме данных

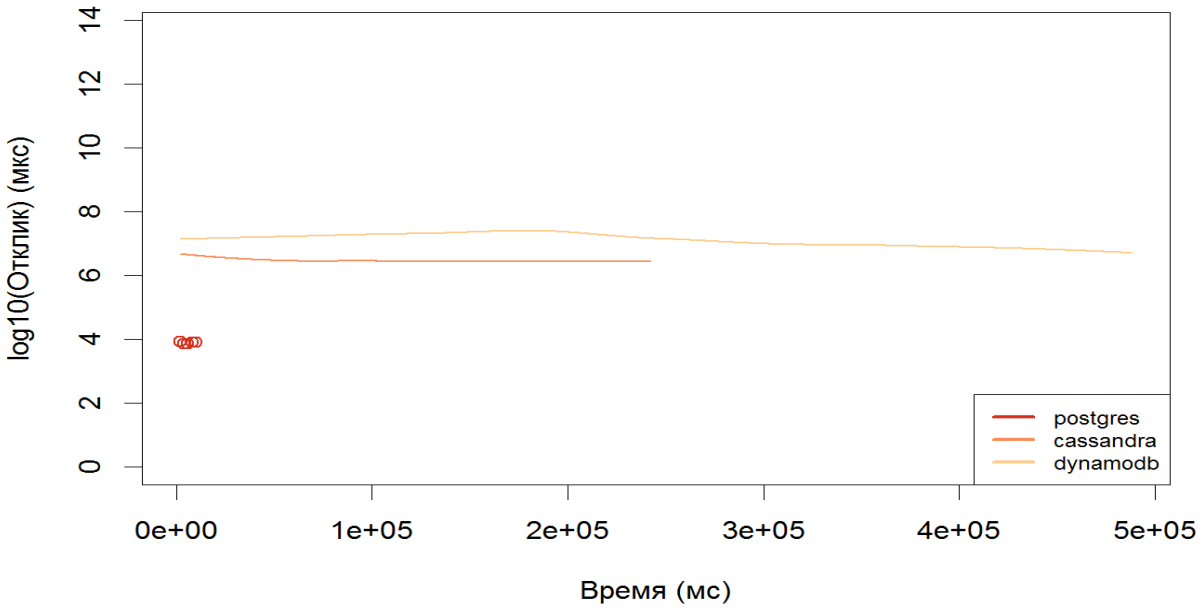


Рис. 17: Отклик операции count group by от времени при среднем объеме данных

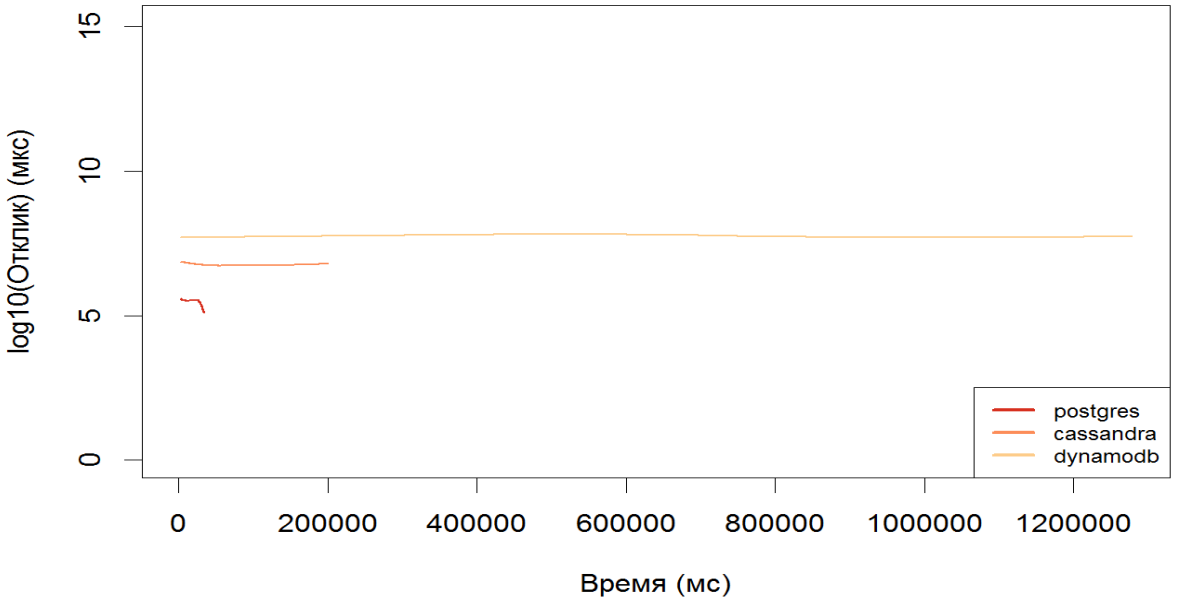


Рис. 18: Отклик операции count group by от времени при большом объеме данных

Из результатов тестирования производительности систем можно сделать вывод, что PostgreSQL выполняет простые операции чтения, вставки и обновления записи не хуже, а в некоторых случаях лучше, исследуемых NoSQL систем. Операция

чтения из соединенных таблиц выполняется в реляционной базе данных медленнее за счет объединения таблиц внутри NoSQL баз данных и выполнения для них простой операции чтения. Такое решение, в свою очередь, несет увеличение объема хранимых данных за счет избыточности при значительном различии ссылающейся и ссылаемой таблиц. Операция посчета сгруппированных записей выполняется Postgres быстрее, так как при работе с NoSQL основные вычисления оставались на стороне пользователя.

Заключение

В данной курсовой работе был произведен краткий анализ реляционных систем управления данными и NoSQL-систем, были приведены их основные отличия. Основной целью работы было сравнение производительности РСУБД и NoSQL-систем.

В ходе работы было проведено тестирование РСУБ Postgresql и NoSQL-хранилищ Cassandra и DynamoDB. Для тестирования производительности использовалась система Yahoo Cloud System Benchmark, модифицированная для работы с нетривиальной схемой данных, состоящей из нескольких связанных внешними ключами таблиц. Адаптация системы YCSB позволяет тестировать скорость выполнения сложных аналитических операций, изначально невозможных, при этом сохраняется гибкость в использовании множества различных СУБД и их настроек, объемов данных и гибкость моделирования сценариев использования СУБД, предоставляемых системой тестирования.

С помощью тестирующей системы YCSB, адаптированной для выполнения аналитических запросов, были получены данные о производительности представленных систем управления базами данных для набора различных запросов при описанной конфигурации. Полученные данные были представлены, и по ним был сделан вывод о производительности реляционной СУБД в сравнении с NoSQL системами.

Список литературы

[1] Abramova Veronika, Bernardino Jorge, Furtado Pedro. Which NoSQL Database? A Performance Overview. — 2014.

[2] Amazon DynamoDB. [Электронный ресурс] - Режим доступа: свободный. — `https://aws.amazon.com/documentation/dynamodb/`. — (дата обращения 1.12.2015).

[3] Apache Cassandra. [Электронный ресурс] - Режим доступа: свободный. — `http://cassandra.apache.org/`. — (дата обращения 1.12.2015).

[4] Apache HBase. [Электронный ресурс] - Режим доступа: свободный. — `https://hbase.apache.org/`. — (дата обращения 1.12.2015).

[5] Benchmarking cloud serving systems with YCSB / Brian F Cooper, Adam Silberstein, Erwin Tam et al. // Proceedings of the 1st ACM symposium on Cloud computing / ACM. — 2010. — P. 143–154.

[6] Cattell Rick. Scalable SQL and NoSQL data stores // ACM SIGMOD Record. — 2011. — Vol. 39, no. 4. — P. 12–27.

[7] Data CJ. An introduction to database systems, 8/E. — Addison-Wesley publ., 2004.

[8] Dynamo: amazon's highly available key-value store / Giuseppe DeCandia, Deniz Hastorun, Madan Jampani et al. // ACM SIGOPS Operating Systems Review / ACM. — Vol. 41. — 2007. — P. 205–220.

[9] He Changlin. Survey on NoSQL Database Technology // Journal of Applied Science and Engineering Innovation Vol. — 2015. — Vol. 2, no. 2.

[10] MongoDB. [Электронный ресурс] - Режим доступа: свободный. — `https://www.mongodb.org/`. — (дата обращения 1.12.2015).

[11] Nayak Ameya, Poriya Anil, Poojary Dikshay. Type of NOSQL Databases and its Comparison with Relational Databases // International Journal of Applied Information Systems. — 2013. — Vol. 5, no. 4. — P. 16–19.

[12] PostgreSQL. [Электронный ресурс] - Режим доступа: свободный. — `http://www.postgresql.org/`. — (дата обращения 1.12.2015).

[13] Tudorica Bogdan George, Bucur Cristian. A comparison between several NoSQL databases with comments and notes // Roedunet International Conference (RoEduNet), 2011 10th / IEEE. — 2011. — P. 1–5.