электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.65

Обзор и сравнительный анализ систем управления нереляционными базами данных

С.И. Лисин, *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение

Научный руководитель: Горин С.В., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечени
irudakov@bmstu.ru

«Большие данные» и нереляционные базы данных

Большие Данные (англ. Big Data) в информационных технологиях — серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения человеко-читаемых результатов, эффективных в условиях непрерывного прироста данных, распределеных по многочисленным узлам вычислительной сети. [10]

Широкий спектр методов и технологий был разработан и адаптирован для объединения, управления, анализа и визуализации больших данных. Эти методы и технологии, взяты из нескольких областей, включая статистику, информатику, прикладную математику и экономику. Некоторые методы и технологии были разработаны в сферах, работающих с гораздо меньшими объемами однотипных данных, но эти методы и технологии были успешно адаптированы для применения к очень большим наборам более разнообразных данных. Другие были разработаны совсем недавно, специально для извлечения информации из больших данных. [2]

В последнее время большую популярность набирают нереляционные базы данных(NoSQL) как для хранения, так и для обработки больших данных. В информационных технологиях термин NoSQL обозначает ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других. [2]

Следующие характеристики часто связаны с NoSQL базами данных:

– NoSQL базы данных разрабатывались для управления большими объемами данных, которым не обязательно следовать фиксированной схеме;

– они не могут дать полную ACID (атомарность, согласованность, изолированность, долговечность) гарантию;

– они имеют распределенную отказоустойчивую архитектуру, могут легко масштабироваться за счет добавления новых серверов и отказ одного сервера не повлияет на работу. Этот тип баз данных обычно масштабируем по горизонтали и используется для управления большими объемами данных, когда производительность в режиме реального времени является более важным, чем консистенция данных.

NoSQL базы данных часто оптимизированы для операций поиска и добавления нового элемента. Они могут быть полезны при работе с огромным количеством данных, которые не требуют описания своей структуры. Для NoSQL систем важна способность хранить и извлекать большие объемы данных, а не отношения между элементами. Это особенно полезно для анализа статистических или растущих в режиме реального времени элементов (таких как сообщения в Twitter'e)[9].

В данной работе основное внимание уделяется наиболее популярным документо-ориентированным нереляционным базам данных, таким как CouchDB, MongoDB, CassandraDB, даётся их обзор и сравнение производительности для операций вставки, выборки, удаления объектов. Также будет произведено сравнение данных нереляционных БД с реляционной БД PostgreSQL.

CouchDB

CouchDB — документо-ориентированная система управления базами данных, не требующая описания схемы данных.

В отличие от реляционных баз данных, CouchDB не хранит данные и отношения в таблице. Вместо этого, каждая база данных представляет собой набор независимых документов. Каждый документ сохраняет свои собственные данные и автономные схемы. Приложение может получить доступ к нескольким базам данных. Документ метаданных содержит информацию о ревизиях, что позволяет определить любые различия, которые, возможно, произошли в то время, как некоторые базы данных были отключены. [7]

Особенности CouchDB:

– целостность базы данных обеспечивается исключительно на уровне отдельных записей (но не на уровне связей между ними);

– связи между таблицами или записями принципиально не поддерживаются, соответственно операция объединения (JOIN) между таблицами не определена;

– одновременно может быть запущено несколько потоков для чтения базы данных и только один — для записи;

– представления хранятся непосредственно в той же БД в которой и сами записи и их индексы обновляются непрерывно;

– распределение вычислений на несколько узлов не поддерживается.

Преимущества CouchDB [1]:

– отсутствие блокировок и, как следствие, высокая доступность;

– инкрементальный MapReduce;

– хранение всех данных в JSON документах;

– REST интерфейс – все операции, включая вставку данных, в CouchDB осуществляются через HTTP;

– множественная репликация – можно использовать неограниченное количество устройств, создавая особые топологии репликации.

К недостаткам CouchDB в основном относят:

– сложность написания запросов на MapReduce и, как следствие, низкая скорость разработки приложения;

– долгое время вставки объектов.

MongoDB

MongoDB — документо-ориентированная система управления базами данных (СУБД), не требующая описания схемы таблиц.

За счёт минимизации семантики для работы с транзакциями в MongoDB появляется возможность решения целого ряда проблем, связанных с недостатком производительности, причём горизонтальное масштабирование становится проще. Используемая модель документов хранения данных (JSON/BSON) проще кодируется, проще управляется, а внутренняя группировка релевантных данных обеспечивает дополнительный выигрыш в быстродействии.[8]

Особенности MongoDB[3]:

– документо-ориентированное хранилище с JSON-подобной схемой данных;

– полная поддержка индексов;

– эффективное хранение двоичных данных больших объёмов;

– журналирование операций, модифицирующих данные в БД;

– поддержка масштабируемости: асинхронная репликация, набор реплик и шардинг;

– может работать в соответствии с парадигмой MapReduce.

К недостаткам MongoDB в основном относят:

– то что, MongoDB — продукт довольно молодой, и в нем встречаются багги;

– по-умолчанию максимальный размер объекта — 4 мегабайта;

– на 32-битных машинах, максимальный размер одной базы данных — ~2.5 гигабайта.

Cassandra

Apache Cassandra — распределённая система управления базами данных, относящаяся к классу noSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша.

Cassandra использует модель хранения данных на базе семейства столбцов (ColumnFamily), что отличается от систем, подобных memcachedb, которые хранят данные только в связке ключ/значение. Cassandra относится к категории хранилищ, повышенно устойчивых к сбоям: помещённые в БД данные автоматически реплицируются на несколько узлов распредёленной сети. При сбое узла, его функции на лету подхватываются другими узлами. Добавление новых узлов в кластер и обновление версии Cassandra производится на лету, без дополнительного ручного вмешательства и переконфигурации других узлов[6].

Особенности Cassandra:

– децентрализованность – каждый узел в кластере имеет одинаковое значение. Не существует единой точки отказа. Данные распределяются по кластеру (так что каждый узел содержит разные данные), но нет хозяина, таким образом, каждый узел может обслуживать любые запросы.

– Поддержка репликации и нескольких центров обработки данных репликации.

– Язык запросов CQL (Cassandra Query Language) - алтренатива SQL.

Сравнение производительности CouchDB, MongoDB, Cassandra, PostgreSQL

В данном эксперименте сравнивалась производительность основных операций, таких как вставка, удаление, выборка. В качестве объектов для хранения и обработки были взяты записи с фиксированным набором полей основных типов данных (целочисленный, с плавающей точкой, временной, строковый). Для сравнения,

эксперимент проводился также и на реляционной базе данных PostgreSQL, которая использовалась как хранилище типа ключ-значение с фиксированным набором полей. При проведении тестов использовался язык питон и библиотеки couchdb, pymongo, cql и psycopg2.

Операции вставки, удаления, выборки проводились при различной «наполненности» баз данных: 0, 1 000, 10 000, 100 000, 1 000 000, 2 000 000 и 3 000 000 строк. Т.к. БД CouchDB показала низкую скорость вставки элементов (на несколько порядков больше остальных), то её тестирование закончилось на отметке 10 000.

Вставка элементов

В ходе проведения эксперимента в базу данных вставлялось 1000 элементов и замерялась средняя скорость вставки.

Результаты представлены в таблице 1 и на рисунке 1.

Таблица 1

Время вставки объектов в БД (в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
cassandra	0,000752	0,000315	0,00087	0,000864	0,000487	0,00093	0,000387
mongodb	0,000115	0,000195	0,000099	0,000097	0,000102	0,000097	0,000094
postgresql	0,000237	0,000225	0,000184	0,000162	0,000209	0,000163	0,000188
couchdb	0,082746	0,082403	0,082367				

Из результатов видно, что быстрее всего вставка объектов идёт в БД mongodb. СУБД postgresql показало меньшее время, что, скорее всего, обусловленно проверкой, не нарушает ли целостность вставляемое значение, а также накладными расходами, связааными с транзакициями при выполнении запроса. У СУБД cassandra поколоночный формат хранения данных, т.е. в памяти последовательно хранятся не кортежи, а атриуты объектов, что в свою очередь приводит к дополнительным накладным расходам при размещении объекта в памяти. СУБД couchdb написана на языке erlang, что не лучшим образом сказывается на её производительности. Данная СУБД хранит все записи в виде строк в JSON формате, что также замядляет работу с данными, т.к. операции работы со строками весьма затратны. Хотя данные в СУБД Mongodb также представлены в JSON формате, но для хранения и обработки используется формат BSON (Binary JSON), который является бинарной фомой представления простых типов данных.

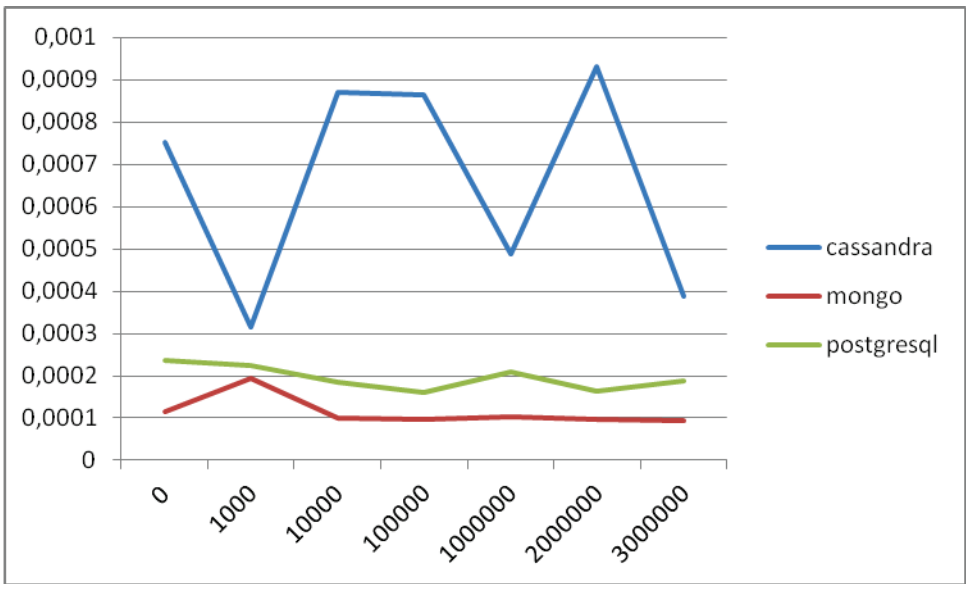

Рис. 1. Время вставки элементов в зависимости от наполненности БД

Простая выборка объектов

В рамках данного теста выбираются все записи, значение выбранного параметра которых больше некоторого значения. Данный запрос реализуется в рассматриваемых СУБД по-разному. В CouchDB для этого используются MapReduce функции. MongoDB обладает встроенными операторами, позволяющими наложить на данные простые фильтры. В Cassandra и PostgreSQL данный запрос осуществляется посредством языков запросов: CQL и SQL соответственно.

Результаты представлены в таблице 2 и на рисунке 2.

Таблица 2

Время простой выборки объектов (в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
cassandra	0,047501	0,077999	0,387682	0,833015	0,920113	0,912034	0,917035
mongo	0,004375	0,00428	0,006966	0,050004	0,558619	1,064267	1,617517
postgresql	0,011873	0,017049	0,066076	0,505179	5,321742	10,626587	15,98035
couchdb	0,461459	0,439767	2,06743				

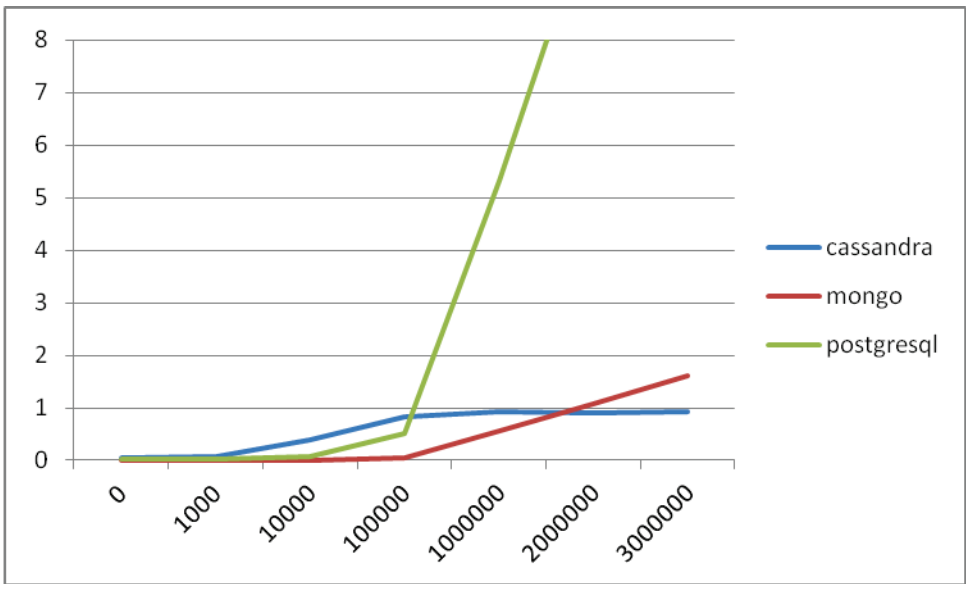

Рис. 2. Время простой выборки элементов в зависимости от наполненности БД

Из результатов видно, что при росте объёма данных с операцией выборки лучше всего справляется СУБД cassandra, что объясняется поколоночным форматом зранения данных, который, по словам исследователей из «Bloor research» лучше всего подходит для операций выборки над большими объёмами данных.

Агрегация данных

В рамках данного теста находится среднее значение некоторого поля, а также максимальное значение другого поля. Данный запрос в нереляционных СУБД выполняется посредством MapReduce функцией. В СУБД Cassandra есть поддержка MapReduce, но на уровне библиотек язык Java, а не на уровне сервера базы данных. Поэтому в данном эксперименте тесты с Cassandra не проводились[4].

Результаты нахождения среднего значения представлены в таблице 3 и на рисунке 3, нахождения максимального значения – в таблице 4 и на рисунке 4.

Таблица 3

Время нахождения среднего значения (в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
mongo	0,02547	0,044817	0,212003	1,905642	18,993331	37,344862	55,99169
postgresql	0,002604	0,001927	0,007669	0,064599	0,672475	1,351388	2,031556
couchdb	0,118381	0,156069	0,48995				

Из результатов видно, что лучше всего с задачей агрегации данных справляется СУБД postgresql. Это связано с тем, что couchdb и mongodb используют парадигму

MapReduce, которая заключается в том, что над каждым объектом в базе данных выполняется некоторая программа (Map-функция), результат которой поступает на вход другой программе (Reduce-функции). В реляционных же базах данных при агрегации выполняются некоторые операции, на вход которых подаются сразу все необходимые данные, что ускоряет нахождение результата, но предоставляет меньшие возможности для распараллеливания запроса, по сравнению с парадигмой MapReduce.

Таблица 4

Время нахождения максимального значения (в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
Mongo	0,025221	0,045554	0,219112	1,976588	20,486356	38,551766	58,22219
postgresql	0,001017	0,000897	0,00369	0,028555	0,285282	0,576101	0,864792
couchdb	0,126301	0,160708	0,52951				

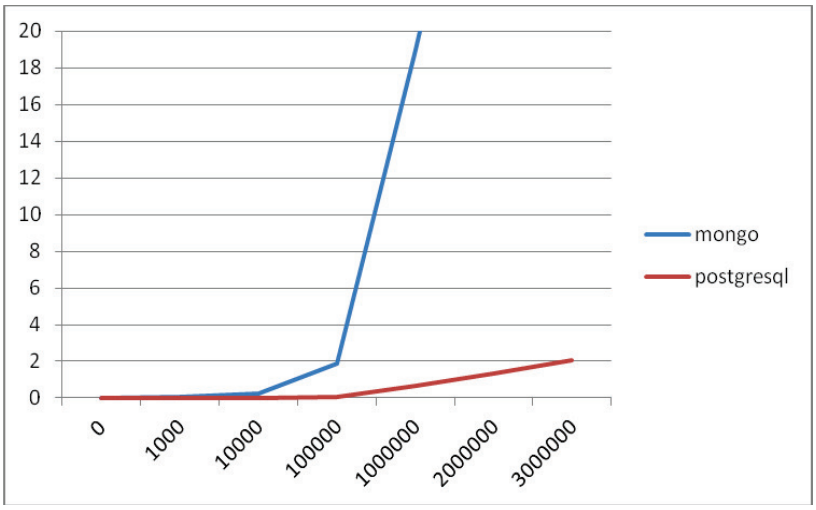

Рис. 3. Время нахождения среднего значения в зависимости от наполненности БД

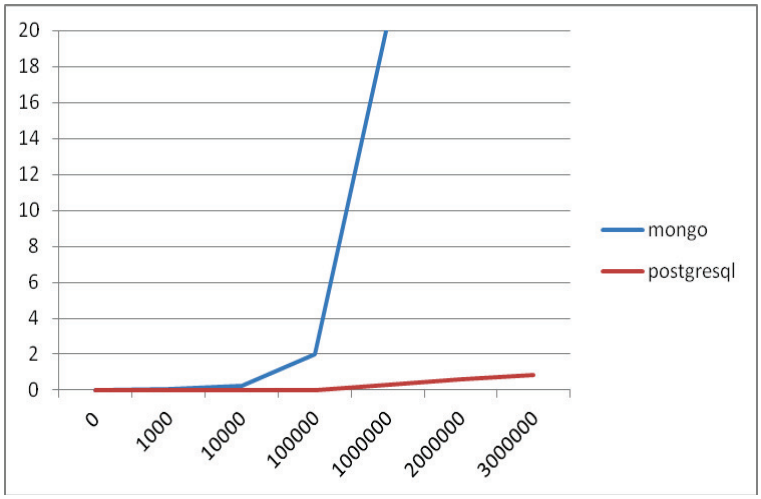

Рис. 4. Время нахождения максимального значения

Доступ к элементу по id

В рамках данного теста определяется среднее время доступа к элементу базы данных по его идентификатору. Результаты представлены в таблице 5 и на рисунке 5.

Таблица 5

Время доступа к элементу по id (в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
cassandra	0,001277	0,00135	0,00101	0,002057	0,006465	0,002325	0,002213
mongo	0,000457	0,002228	0,006869	0,05232	0,571589	1,098512	1,699814
postgresql	0,000437	0,000708	0,001662	0,009389	0,095481	0,19627	0,296314
couchdb	0,040234	0,040194	0,040514				

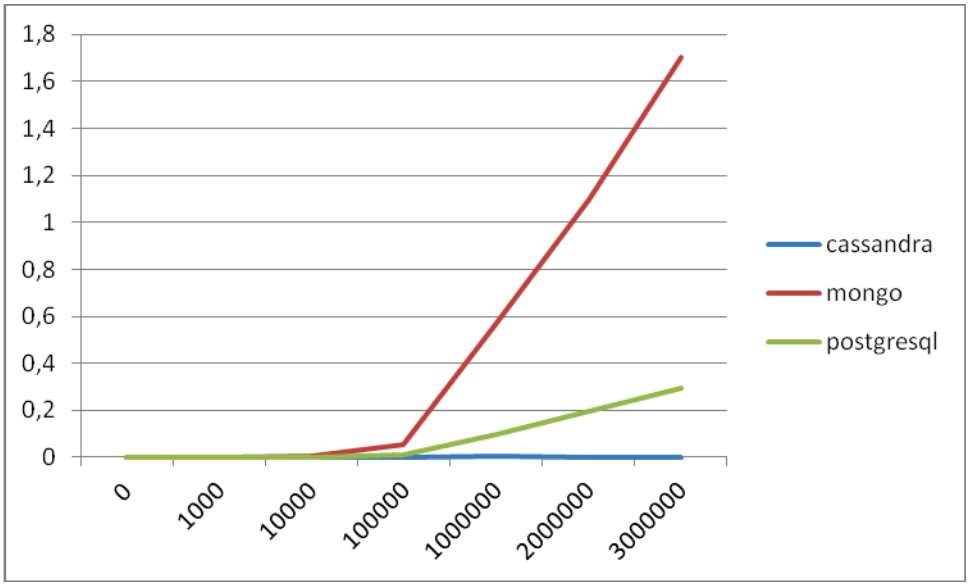

Рис. 5. Время доступа к элементу по id в зависимости от наполненности БД

Из результатов видно, что лучше всего с данной задачей справляется СУБД cassandra, что весьма странно, т.к. поколоночный формат хранения хорошо подходит для выбора больших объёмов данных, а не единичного объекта. Скорее всего, такие результаты обусловлены тем, что СУБД cassandra, так же как и СУБД postgresql, по умолчанию используют индексы на первичные ключи объектов.

Удаление элементов

В рамках данного теста определяется среднее время удаления одной записи из БД. Стоить отметить, что среди рассматриваемых СУБД удалить за один запрос несколько записей (например, удалить все записи, значение некоторого параметра которых меньше определённого значения) могут только СУБД MongoDB и PostgreSQL.

Результаты представлены в таблице 6 и на рисунке 6.

Таблица 6

Время удаления элемента из БД(в секундах)

БД	Количество записей в БД						
	0	1000	10000	100000	1000000	2000000	3000000
cassandra	0,0005	0,000616	0,000638	0,000878	0,001048	0,000891	0,000848
Mongo	0,000056	0,000066	0,000056	0,000057	0,000077	0,000068	0,000054
postgresql	0,000395	0,000706	0,00122	0,009146	0,095989	0,196652	0,296522
couchdb	0,044951	0,045411	0,048183				

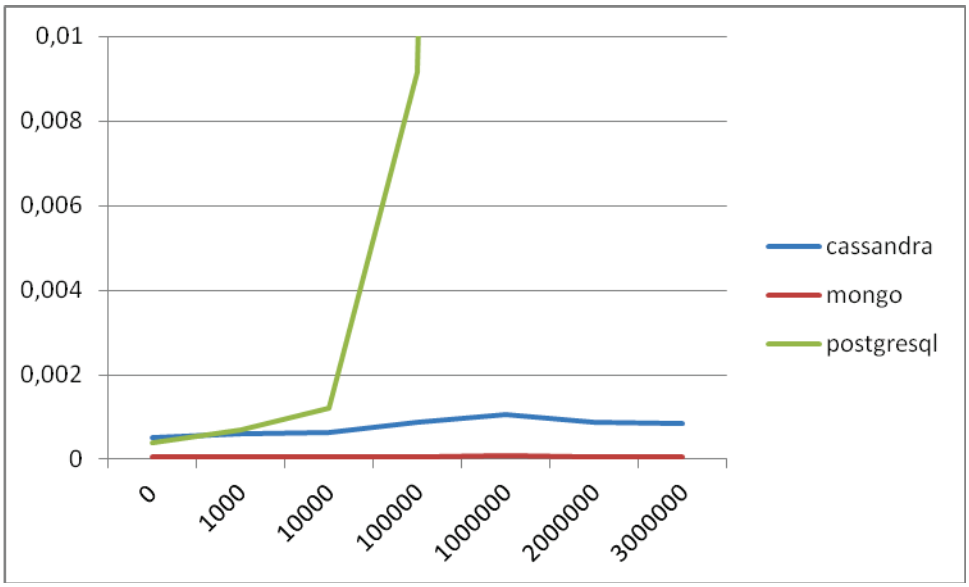

Рис. 6. Время удаления элемента из БД в зависимости от наполненности БД

Выводы

В результате проведённых экспериментов можно сделать вывод, что разные СУБД оптимизированы для работы с различными типами задач. Если требуется быстрая вставка/удаление объектов, то с этой стороны лучше всего себя показала БД mongodb. БД Cassandra организовано как поколоночное хранилище данных и оптимизировано для операций выборки. Если требуется некий анализ данных, такой как поиск максимальных и минимальных значений, вычисление суммы и среднего некоторых величин, то здесь лучше всего справляются реляционные хранилища данных.

Но данный эксперимент проводился на одном компьютере. Нереляционные хранилища данных разрабатывались для развёртывания на кластерах (из-за этого у них есть некоторые ограничения целостности и ограниченный функционал работы с отношениями между сущностями). Модель распределённых вычислений MapReduce

предназначена для выполнения на нескольких узлах кластера, что повышает скорость получения результата, но, как показали другие исследования, недостаточно сильно.

Группа известных специалистов по базам данных во главе с Майклом Стоунбрейкером и Дэвидом Девиттом провела ряд испытаний, результаты которых были приведены в статье «A Comparison of Approaches to Large-Scale Data Analysis». В проводимых ими экспериментах система Hadoop сравнивалась с параллельными реляционными СУБД Vertica и СУБД-X, которые по итогам продемонстрировали существенное превосходство в производительности над Hadoop при выполнении ряда тестовых задач анализа данных большого объема.

В принципе, данный эксперимент подтвердил слова Филиппа Ховарда, директора отдела исследований компании Bloor Research: «Подход, ориентированный на столбцы, обеспечит гораздо более высокую производительность при более низкой стоимости по сравнению с другими подходами к организации сред анализа, отчетности и хранения данных». [11]

Но это не значит, что стоит полностью отказаться от NoSQL-систем. Существуют определённые типы задач, когда эффективнее использовать реляционные СУБД, а когда NoSQL-системы.

Так, например, для построения различных отчётов, в которых требуется подсчитать сумму некоторых показателей, их максимальные, минимальные и средние величины, количество каких-либо сущностей (так называемый Business intelligence), то здесь лучше подходят реляционные СУБД.

В таблице 7 приведены характерные черты реляционных СУБД и NoSQL-систем, которые призваны помочь определиться с выбором подхода к хранению и анализу данных в различных задачах[5].

Таблица 7

Сравнение реляционных СУБД с NoSQL-системами

Реляционные базы данных	**NoSqL-системы**
Структурированные данные	Неструктурированные данные
ACID	Нет ACID
Строгая консистентность данных	Частичная консистентность данных
ETL	Нет ETL
Требуется время для получения результатов	Быстрое получение результатов
Зрелость, стабильность, эффективность	Гибкость

Список литературы

1. Foundation, Apache Software. Apache CouchDB. — 2012. http://couchdb.apache.org/.

2. James Manyika, Michael Chui. Big data: The next frontier for innovation, competition, and productivity / Michael Chui James Manyika // McKinsey Global Institute. — 2011.
3. Артемов, Сергей. Большие данные = большая проблема? / Сергей Артемов // JET INFO. — 2012. — июнь.
4. Говард, Филипп. SYBASE IQ 15.1 / Филипп Говард // Исследования Bloor Research. — 2009.
5. Черняк, Леонид. Большие Данные — новая теория и практика / Леонид Черняк // «Открытые системы» , № 10. — 2011.