УДК 004

ИССЛЕДОВАНИЕ И СРАВНЕНИЕ ВОЗМОЖНОСТЕЙ NoSQL СУБД Cassandra и РЕЛЯЦИОННОЙ СУБД PostgreSQL В ЗАДАЧЕ ПОИСКА DICOM ИЗОБРАЖЕНИЙ

Карпунов Г.А., Костюкова Н.С.
Донецкий Национальный Технический университет
кафедра прикладной математики и информатики
E-mail: sweet_misery@list.ru

Аннотация

Карпунов Г.А., Костюкова Н.С. Исследование и сравнение возможностей NoSQL СУБД Cassandra и реляционной СУБД PostgreSQL в задаче поиска DICOM изображений. *Рассмотрен подход хранения изображений в СУБД Cassandra и СУБД PostgreSQL. Определены преимущества и недостатки схемы данных для хранения PACS DICOM изображений в указанных СУБД Рассмотрены варианты быстродействия при разных схемах архитектуры NoSQL СУБД Cassandra.*

Общая постановка проблемы

Большинство современных медицинских учреждений имеют MRT сканнеры, которые получают данные в цифровом формате, преобразуя их в DICOM файлы. За несколько лет работы MRT сканнера накапливаются DICOM файлы исследований объемами в десятки терабайт.

В реляционных базах данных при хранении атрибутов DICOM файлов используется схема «ключ-значение»: каждый атрибут имеет уникальный ключ, по ключу совместно с идентификатором DICOM файла можно получить значение атрибута. Для выборки и отсеивания по нескольким атрибутам в реляционных базах данных необходимо применять стандартную реляционную операцию JOIN при каждом новом атрибуте. Стандарт DICOM подразумевает более 1500 атрибутов, больше половины которых может использоваться в качестве метаданных DICOM файлов. Соответственно, для каждого DICOM файла мы можем получить 1000 атрибутов в базе данных, и вынуждены использовать для отсеивания по набору, например, из 100 атрибутов, 100 операций JOIN. Это приводит к тому, что каждый запрос к небольшому PACS архиву выполняется более 20 секунд, что неприемлемо к системе с потенциально большой нагрузкой.

Необходимо найти альтернативу реляционным базам данным, которая существенно ускорит поиск таких файлов по их метаданным.

Исследования

Для исследования воспользуемся реляционной СУБД PostgreSQL и нереляционной СУБД Cassandra. Объектом исследования будет служить PACS архив разнородных изображений, которые имеют разный размер и, соответственно, разное число атрибутов. Ряд изображений тестовой выборки представляются действительными снимками MRT сканнера (5.7 Гб изображений, 46458 DICOM файлов).

Каждый DICOM файл может содержать различное число атрибутов метаданных. В исследовании будут участвовать все метаданные, которые хранятся в DICOM файле. В СУБД PostgreSQL будет использоваться операция JOIN. В системах NoSQL данные хранятся не по реляционной модели. Это подразумевает возможную избыточность данных. Тем не менее, за счет избыточности NoSQL СУБД работают в определенных классах задач значительно быстрее классических реляционных СУБД.

Для эксперимента с СУБД Cassandra будем использовать следующую архитектуру сети (рис. 1):

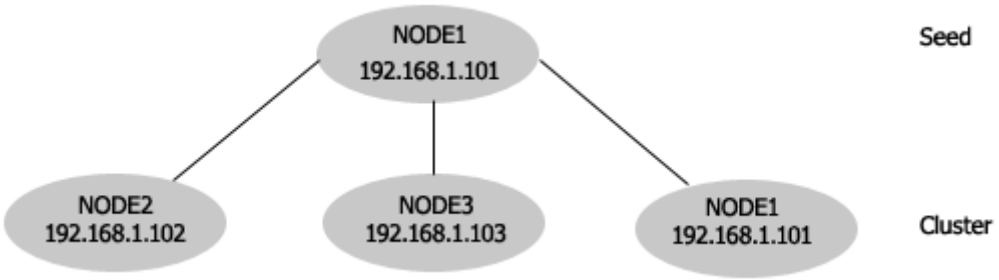

Рисунок 1 – Структура вычислительной сети для исследования

Узлы содержат следующие характеристики (табл. 1):

Таблица 1 – Характеристики узлов сети для исследования

Узел	Процессор	Оперативная память	Скорость HDD
NODE1	Core 2 QUAD Q6600 2.4 Ггц	4096 Мб	7500 об/мин
NODE2	Intel Pentium Dual Core 1.3 Ггц	4096 Мб	7500 об/мин
NODE3	Intel Pentium 3 1 Ггц	256 Мб	7500 об/мин

Сеть между узлами поддерживается с использованием маршрутизатора Cisco Linksys WAG120N, настроенного на скорость передачи данных 100 Мбит/с.

Можно обнаружить, что узел, который распространяет информацию о кластере (NODE1) включен в кластер и работает с данными аналогично NODE2 и NODE3. Получаем кластер, состоящий из 3 узлов (рис. 2).

```
[default@unknown] describe cluster;
Cluster Information:
   Snitch: org.apache.cassandra.locator.SimpleSnitch
   Partitioner: org.apache.cassandra.dht.RandomPartitioner
   Schema versions:
        020b6e96-4f01-11e0-b5a1-bd9004a36f78: [192.168.1.103, 192.168.1.102, 192.168.1.101]
```

Рисунок 2 – Информация о кластере

Вследствие того, что реляционные базы данных являются трудно масштабируемыми по горизонтали, сложно масштабировать на имеющуюся сеть СУБД PostgreSQL как минимум с сохранением производительности, которая, наиболее вероятно, в рассматриваемом примере будет ниже. Для эксперимента с СУБД PostgreSQL используется только NODE1 – самый мощный из представленных узлов.

Для добавления данных в базы данных PostgreSQL и Cassandra по выборке из 46458 DICOM файлов сгенерируем файлы с запросом. Для этого извлекаем все атрибуты из каждого файла выборки. Получаем более 47 млн пар вида «ключ-значение».

После занесения данных в схему СУБД PostgreSQL объем базы данных составил 72,65Мб. Данные заносились частями по транзакциям, суммарное время занесения составило 124 минуты. Схема данных представлена на рисунке 3:

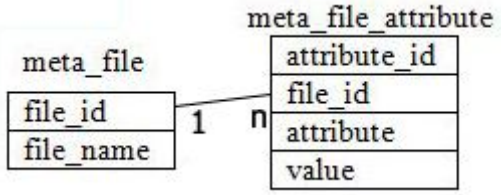

Рисунок 3 – Схема данных для исследования в PostgreSQL

При занесении данных в схему СУБД Cassandra используется внешний файл, загружаемый с помощью утилиты cassandra_cli. При загрузке схемы на каждой машине, входящей в кластер, формируется лог транзакции. Большинство NoSQL систем, в отличие от реляционных баз данных, организуют хранилище на основе хеш-таблиц. При этом данные прозрачно масштабируются на множество узлов, входящих в кластер. Вследствие этих условий поддерживать внутренний механизм транзакций в NoSQL становится сложно. При добавлении данных формируется внешний файл транзакции, который сохраняет состояние выполнения большого запроса. Этот файл по мере выполнения запроса обновляется на всех узлах, входящих в кластер. Каждый узел получает только ту часть добавляемых данных с транзакции, которая будет храниться у него в файловой системе. Таким образом, происходит распределение данных между узлами еще в процессе занесения записей.

Алгоритм записи в схему Cassandra:

- создание пространства ключей dicom;
- загрузка данных в формате dicom[‘file_name’][‘attribute’] = ‘value’

Запись заняла 49 минут, что примерно в 2 раза быстрее, чем при записи с использованием PostgreSQL. При этом суммарный объем данных, который заняла база данных, на трех узлах составил 93,76Мб. Стоит отметить, что на машине, которая выполняла запрос, сохранилось 79,02Мб данных. Можно обнаружить, что на двух других узлах объем данных выходит одинаковый - 7,36Мб.

Рассмотрим варианты таких отличий в хранении данных. В реляционных базах данных широко применяется резервное копирование. В большинстве случаев резервное копирование делается только несколько раз в неделю. Если отказывает накопитель, может потеряться часть информации.

В NoSQL системах на данном этапе их развития для важных данных необходимо иметь значительную репликацию. При частом обращении к базе данных множества пользователей возникают сложные транзакции, которые иногда могут приводить к потерям данных даже в штатном режиме. В крупных компаниях, которые используют NoSQL системы (Google, Facebook и др.) создаются алгоритмы, которые, при частичной репликации на каждый узел, впоследствии могут собирать данные с разных узлов по одному и тому же ключу, анализировать их корректность и делать вывод об их достоверности.

При запуске Cassandra с узла, который не распространяет информацию о кластере, ему можно указывать параметр auto_bootstrap. С установленным параметром узел автоматически скопирует часть данных к себе с уже запущенных узлов. Это также способствует репликации: при отключении одного из узлов, данные, которые он содержал, можно восстановить из реплик, которые содержатся на других узлах. Тем не менее, при отключении значительного числа узлом часть данных может оказаться временно недоступной.

Еще одним недостатком реляционных баз данных является единственная точка входа. В небольших системах нет необходимости в другой архитектуре, потому что такая схема удобна для понимания и управления. Таким образом, если узел, на котором используется база данных, отказывает – отказывает вся система.

NoSQL СУБД предусматривают лучшую отказоустойчивость, так как существует несколько узлов, с которых можно получить данные. В приведенной схеме такой точкой является NODE1. В более сложных системах принято 5-30% узлов в кластере запускать в качестве машин, распространяющих информацию о кластере. Вероятность отказа такого числа узлов значительно ниже, соответственно система с такой архитектурой является более надежной.

После занесения данных в базы данных можно исследовать время выполнения запросов в каждой из них. Результаты выполнения различных запросов представлены в таблице 2.

Таблица 2 – Результаты выполнения запросов

Цель запроса	Структура выполнения		Время выполнения	
	Cassandra	PostgreSQL	Cassandra	PostgreSQL
Получить все атрибуты файла	Задание ключом имени файла, результат – все атрибуты	Использование JOIN для добавления каждого атрибута	215 мс	>10 мин
Получить файлы по 4 атрибутам	Задать параметр отсеивания	Использование JOIN для добавления каждого атрибута, четыре JOIN	220 мс	31сек
Получить все файлы, у которых заданный атрибут равен заданному значению	Задаем индекс по атрибуту, ищем по значению	Выбрать атрибут для каждого файла, один JOIN	510 мс	70 сек

Как видно из результатов, полученных при выполнении запросов, скорость NoSQL СУБД превосходит реляционную СУБД PostgreSQL в несколько раз.

В запросе получения всех атрибутов файла очевидно, что при использовании SQL придется использовать JOIN столько раз, сколько файл имеет атрибутов. Таким образом, результирующий запрос будет занимать несколько Кб и выполняться неприемлемо долго. В Cassandra при записи данных задавался ключ – имя файла, поэтому по ключу можно за доли секунды получить все атрибуты файла.

Запрос на получение файлов по 4 атрибутам является одним из самых часто встречающихся запросов. При поиске DICOM файлов параметрами поиска в большинстве случаев служат теги PatientID, StudyInstanceUID, SeriesInstanceUID, SopClassesUID. При использовании SQL нам потребуется 4 операции JOIN для таблицы meta_file_attribute. При использовании схемы Cassandra можно задать параметр, по которому отсеивать файлы. Извлечение будет происходить аналогично, записи, не удовлетворяющие критерию, будут отсеиваться.

В запросе на получение всех файлов, у которых заданный атрибут равен заданному значению, в схеме PostgreSQL будет использован лишь один JOIN для объединения таблиц meta_file и meta_file_attribute. В Cassandra такой запрос будет выполняться аналогично предыдущему случаю.

Рассмотренные запросы являются одними из самых часто используемых при поиске DICOM файлов по метаданным в базе данных. Тем не менее, существует еще множество запросов, которые в данном примере не рассматриваются. Их удельный вес среди выше указанных запросов составляет менее 5%. Некоторые из них в архитектуре NoSQL могут работать значительно медленнее, чем в реляционных базах данных. Тем не менее, одной из ключевых концепций NoSQL систем является избыточность данных. В большинстве случаев можно, не убирая существующие данные, добавить новые записи таким образом, чтобы различные запросы проходили за минимальное время. Очевидно, что для этого требуется дополнительное пространство на накопителях.

Следует отдельно отметить, что в NoSQL СУБД запросы на обновление данных выполняются дольше, чем в реляционной СУБД. Это обуславливается необходимостью синхронизации всех распределенных хранилищ с обновленными данными. В примере с DICOM файлами очевидно, что атрибуты файла статичны и изменению в базе данных в дальнейшем подлежать не будут.

Выводы

NoSQL системы управления базами данных являются мощным инструментом для разработки больших распределенных систем. При правильном проектировании архитектуры системы и наличии достаточных аппаратных ресурсов, с использованием NoSQL можно добиться увеличения скорости работы с базой данных в десятки раз.

В то же время нереляционные СУБД подходят не для всех типов задач. В проблемах, при решении которых часто приходится обновлять данные в таблицах, а также со сложной логической структурой данных, проектирование правильной NoSQL архитектуры может занять значительное время, а в некоторых случаях вообще оказаться нецелесообразным.

Благодаря специфике хранения данных, NoSQL СУБД в отдельных видах запросов может показывать значительную производительность. Разработчики NoSQL систем рекомендуют при разработке приложений большинство промежуточных данных, которые представляют собой сгруппированные по каким-либо критериям исходные данные, но еще не полностью сформированные результаты, хранить в базе данных, тем самым обеспечивая повышение быстродействия в схожих запросах.

Также при рассмотре разработок с использование NoSQL СУБД необходимо учитывать простоту горизонтального масштабирования, и в тоже время ее прозрачность. В каждый момент времени легко разобраться, что происходит в системе и какие узлы участвуют в формировании ответов на запросы.

На сегодняшний день в системах PACS NoSQL еще не применяется. Это обуславливается относительной новизной технологии, а также тем, что на данном этапе развития NoSQL хранилища не всегда могут гарантировать сохранность всех данных, что может быть критично в медицинских системах.

Новые NoSQL проекты и решения предлагаются крупными компаниями, которые используют такие разработки, в первую очередь, в своих целях. Поэтому данные технологии будут активно развиваться с ростом числа пользователей Интернет. На сегодняшний день это перспективное направление исследований, так как оно может помочь многим разработчикам мира отойти от логичной и последовательной реляционной концепции и прийти к избыточности данных, нарушению целостности, но, вместе с тем, к значительному приросту производительности и скорости.

Литература

1. Clunie, David A.; DICOM Structured Reporting; PixelMed Publishing, Bangor, PA, 2001.
2. Revet, Bas; DICOM Cook Book for Implementations in Modalities; 1997; Philips Medical Systems.
3. International Telecommunication Union [Electronic resourse] / Интернет-ресурс. - Режим доступа : FTP: ftp://medical.nema.org/medical/dicom/2009/
4. International Telecommunication Union [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/URL: http://wiki.apache.org/cassandra/ - Cassandra Wiki
5. International Telecommunication Union [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/URL: http://wiki.postgresql.org/wiki/Main_Page - PostgreSQL Wiki
6. International Telecommunication Union [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/URL: http://jdevnotes.blogspot.com/2010/12/apache-cassandra.html - Java Dev Notes
7. International Telecommunication Union [Electronic resourse] / Интернет-ресурс. - Режим доступа : www/URL: http://wiki.apache.org/cassandra/CassandraCli - CassandraCLI - Cassandra Wiki.