ПРАКТИЧЕСКАЯ РАБОТА 10. ШАРДИНГ В MONGODB. РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ. MAPREDUCE НА НЕСКОЛЬКИХ СЕРВЕРАХ

10.1 Цель и содержание

Цель работы: изучить модель шардинга MongoDB.

Задачи работы: научиться создавать и конфигурировать шард-сервера.

10.2 Шардинг в MongoDB

Шардинг (партицирование) – разделение данных на несколько серверов (узлов) с сохранением определенного порядка.[19]

Шардинг используется для выравнивания нагрузки на запрос записей. При использовании шардинга данные исходной коллекции распределяются по нескольким серверам с использованием определенного шард-ключа. Шардинг позволяет создавать коллекции неограниченного размера, содержащие огромное количество записей, что важно для работы с большими данными.

Для каждой коллекции, которая распределена по шардам, определяется ключ шардинга. Пространство ключей для ключа шардинга делится на непрерывные диапазоны ключей (чанки), которые размещены по соответствующим шардам.

Схематически архитектура шардинга MongoDB представлена на рисунке 1.

Клиентское приложение – это приложение, которое подключается к MongoDB и использует его как хранилище данных. MongoDB может одновременно работать с несколькими клиентами.

Сервер маршрутизации (процесс «mongos.exe») выполняет функции доступа к кластеру. «Mongos.exe» ведет себя подобно «mongod.exe», обеспечивая прозрачное взаимодействие с хранилищем данных. Клиент, взаимодействуя с «mongos.exe» не имеет представления о том, что находится за ним: локальная база данных или распределенное хранилище. «Mongos.exe» хранит сведения о том, какие документы хранятся, в каких узлах (шардах).

[19] http://gliffer.ru/articles/nosql--sharding-mongodb-na-paltsah/

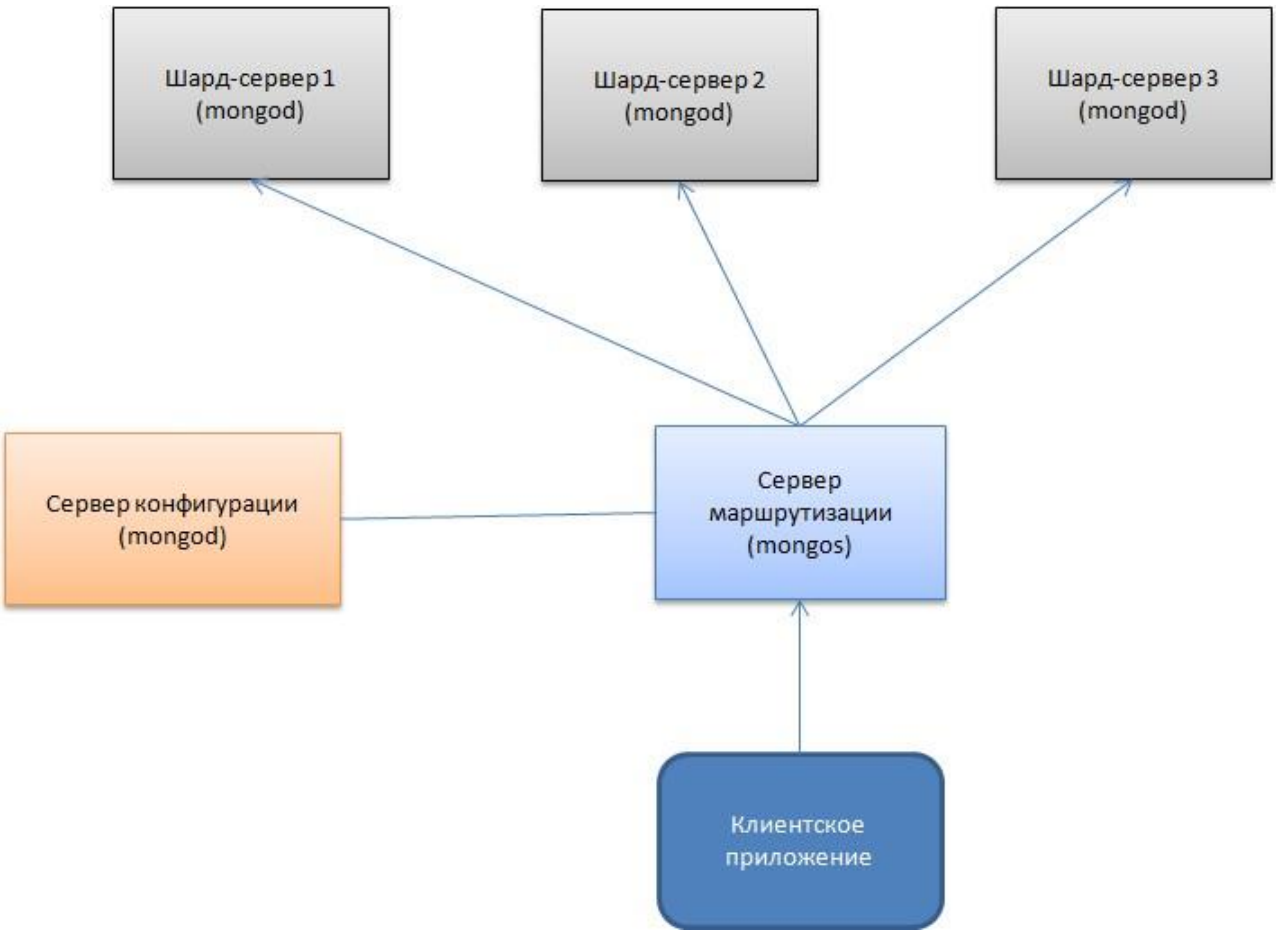

Рисунок 1 – Архитектура шардинга MongoDB

Шард-сервер состоит из шарда и процесса, управляющего шардом. Шард – это определенный набор данных. «Mongod.exe» – это процесс, управляющий данными шарда. Один шард может обслуживаться несколькими mongod-процессами. Также каждый шард может обслуживаться распределенными mongod-процессами (репликами).

Как было указано ранее, чанки – это непрерывные куски данных какой-либо коллекции. Каждый чанк характеризуется следующими параметрами:

- минимальный шард-ключ (MinKey);
- максимальный шард-ключ (MaxKey);
- имя шарда, где находится чанк.

Когда чанк становится большим и превышает максимальный размер чанка, он делится пополам (этот процесс называется балансировкой и выполняется автоматически). Когда совокупный размер чанков шарда больше доступного для шарда, чанк переносится на другой шард.

Сервер конфигурации является процессом (или процессами), который хранит метаданные о шардах (описание чанков).

Коротко процесс обработки запроса на выборку данных из распределенной коллекции выглядит следующим образом: клиентский процесс обращается к серверу маршрутизации. Сервер маршрутизации обращается к серверу конфигурации и выясняет, в каких чанках хранятся требуемые данные и на каких шардах расположены эти чанки. После того как сервер маршрутизации получил данные о шардах и чанках, он обращается к ним (процесс «mongod») и получает искомые данные. Далее сервер маршрутизации возвращает данные клиенту.

Если запросы на изменение/вставку/удаление содержат ключ шардинга, то сервер маршрутизации, основываясь на информации, полученной от сервера конфигурации, передаст запрос к основному серверу, содержащему чанк с данными, диапазон ключей которого покрывает ключ шардинга искомых документов.

В случае запроса без указания ключа шардинга (ключ для шардинга не является частью критерия выбора) сервер маршрутизации отправит запрос к каждому шарду. Каждый шард произведет локальный поиск, а результат будет собран на сервере маршрутизации и возвращен клиенту.

10.3 MapReduce над распределенной коллекцией

Операция MapReduce, также как и выборки данных, происходит прозрачно для клиента без указания того, с каким хранилищем в данный момент происходит работа: с распределенным или с локальным.

Распределенный MapReduce в MongoDB состоит из следующих шагов:

1. Клиент после определения функций map/reduce и исходных коллекций отправляет запрос к серверу маршрутизации.
2. С помощью сервера конфигурации определяется расположение искомых чанков.

3. Сервер маршрутизации передает запрос соответствующим шардам (или шарду, как и с обычными запросами шарды, вовлеченные в операцию, зависят от того, указан ли ключ шардинга в запросе).

4. База данных каждого шарда выполняет запрос и функцию map, которая выдает набор пар ключ/значение.

5. Вызывается функция reduce для результата, полученного на предыдущем шаге. Функция reduce также может быть вызвана на предыдущем шаге, если объем вывода функции map полностью заполняет буфер памяти. В этом случае вызов функции reduce частично уменьшит объем занимаемой памяти.

6. После окончания работы предыдущего шага сервер маршрутизации уведомляет шарды, которые будут хранить обработанные данные.

7. База данных шарда, хранящего результирующую коллекцию, запросит частично уменьшенные данные от шардов на основе своего диапазона ключей.

8. База данных шарда снова выполняет функцию reduce над уже частично уменьшенным результатом. Затем результат сохраняется в базу данных, расположенную на этом (-их) шарде (-ах).

9. Выполняется функция финализации, если она представлена пользователем.

10.4 Выбор ключа для шардинга

Выбор ключа для шардинга является важной темой. Шардинговый ключ необходимо выбирать таким образом, чтобы данные можно было равномерно распределить между шардами. Примером ключа шардинга могут быть: диапазон имен, номера штрих-кодов, и т.д. Общие рекомендации дать сложно, т.к. в каждом конкретном случае шардинговый ключ подбирается в зависимости от задачи.

Иногда, когда документы не содержат полей, которые могут быть использованы в качестве ключа для шардинга, в документы добавляет поле, которое содержит случайные данные и используется в качестве ключа для шардинга. Примером такого поля может служить хеш-сумма документа. Иногда в качестве ключа для шардинга используется поле «_id». Однако использование «_id» объекта по умолчанию в качестве ключа для шардинга является не очень хорошей идеей.

Для каждого конкретного случая подходит свой ключ. Можно выделить следующие типы ключей:

- Случайный. Ключ генерируется случайно, с использованием различных генераторов случайных чисел/последовательностей, либо различных хэш-сумм. Такой ключ отлично подходит для распределенного чтения и записи на всех шардах. Необходимо иметь ввиду, что для того, чтобы получить выгоду от использования шардинга при выполнении чтения данных, необходимо использовать свой ключ в запросах. В противном случае mongoDB будет опрашивать все шарды. При использовании ключа для шардинга в запросе, mongoDB может сразу обратиться к требуемому чанку на заранее известном шарде.
- Ключ в определенном диапазоне. К данным типам ключей относятся ключи, которые имеют заранее определенный диапазон. Например, для ведения журнала приложения может быть использован ключ, привязанный к типу записи и имеющий следующие возможные значения: «Ошибка», «Предупреждение», «Сообщение». Очевидно, что в таком случае в базе данных будет всего три чанка, и они могут достигать очень больших размеров. К данной группе ключей также относятся ключи, ориентированные на разбиение данных в соответствии с алфавитом, либо другим критерием, имеющим определенный диапазон.
- Составной ключ. Если необходимо использовать ключи, описанные в предыдущем пункте, и количество их значений ограничено, то для уменьшения размера чанков, можно использовать составной ключ. В приведенном выше примере, если Вы хотите использовать ключ с заранее определенными состояниями, Вы можете его сделать составным и добавить метку с датой создания. Теперь в случае превышения максимального размера чанка, mongoDB

может разделить его и вынести в отдельный чанк.

- Ключи, основанные на временных метках. При использовании подобного рода ключей в качестве ключа выступают какие-либо данные, привязанные ко времени. Например, дата создания документа. Однако следует иметь ввиду, что, так как в данном типе ключа отсутствует всякая случайность, а временные метки всегда возрастают, то Вы будете всегда производить запись в один чанк. Чанк может измениться, но все Ваши записи всегда будут производиться в одно и тоже место. Это может быть проблемой и узким местом приложения, если Вы имеете требовательные к ресурсам операции записи.[20]

10.5 Методика и порядок выполнения работы

Для выполнения работы необходимо разделиться на группы по три человека. Один студент (1) создает и настраивает сервер конфигурации, сервер маршрутизации и добавляет базу данных, второй (2) и третий (3) студенты подготавливают свой компьютер для использования в качестве шарда. После выполнения задания студенты меняются ролями.

Подготовка шард-сервера (2), (3):

1. Запустите «mongod» со следующими параметрами:

```
mongod --shardsvr <options>,
где options – другие опции настройки «mongod» (например: путь к каталогу с БД)
```

После успешного запуска «mongod» на ваших компьютерах заработают шард-сервера.

2. Узнайте IP-адрес Вашего компьютера. IP-адрес компьютера можно посмотреть в свойствах сетевого подключения, либо введя команду «ipconfig» в командной строке.

Создание сервера конфигурации (1). Сервера конфигурации и маршрутизации расположены на локальной машине, имеющей IP-адрес 127.0.0.1.

[20] http://www.quora.com/MongoDB/How-should-shard-keys-be-assigned-with-MongoDB

Для создания сервера конфигурации необходимо запустить «mongod» с параметром «--shardsvr»:

```
mongod --shardsvr
```

Создание сервера маршрутизации (1). Для создания сервера конфигурации введите в консоль:

```
mongos --configdb 127.0.0.1:<port>,
где port – порт, на котором запущен «mongos» (по умолчанию: 27018)
```

Конфигурирование кластера (1):

- Запустите процесс «mongo».
- Войдите под именем администратора.
- Узнайте у студентов, работающих в Вашей группе IP-адреса их компьютеров, предположим это: 192.168.1.2 и 192.18.1.3.
- Добавьте шарды в кластер:

```
db.runCommand( { addshard : "192.168.1.2:27018" } );
db.runCommand( { addshard : "192.168.1.2:27018" } );
```

- Сообщите серверу имя базы данных для шардинга:

```
> db.runCommand( { enablesharding : "pdb" } );
```

- Сообщите серверу имя коллекции для шардинга (в качестве ключа для шардинга выберите одно из полей БД и обоснуйте свой выбор; указывать следует свою базу данных, созданную в предыдущих работах):

```
> db.runCommand( { shardcollection : "pdb.phones", : {<key>:<keyValue>} })

где <key> - имя ключа для шардинга;
    <keyValue> - значение ключа для шардинга.
```

- Создайте map и reduce функции, основываясь на знаниях, полученных в ходе выполнения предыдущих работ.

- Выполните команду mapReduce над одной из коллекций. Результат работы выведите в консоль и сохраните в виде отдельной коллекции. Результатом работы MapReduce должен быть подсчет общего количества чего-либо, хранящегося в базе данных.