ЛР6. MongoDB. Репликация. Сегментирование

1. Репликация

В MongoDB репликация настраивается путем создания набора реплик.

Набор реплик – это группа серверов с одним первичным узлом, который получает операции записи, и несколькими вторичными узлами, где хранятся копии данных первичного узла.

Если первичный узел дает сбой, вторичные узлы могут выбрать новый первичный узел из их числа. Если используется репликация, и сервер выходит из строя, можно получить доступ к данным с других серверов в наборе реплик. Если данные на сервере повреждены или недоступны, можно сделать новую копию данных с одного из других членов набора реплик.

Репликация в MongoDB основана на двух механизмах: журнале операций и тактовом сигнале. Журнал операций делает репликацию возможной, а с помощью тактовых сигналов ведется мониторинг состояния и активируется процедура обработки отказа.

Журнал операций представляет собой ограниченную коллекцию в базе local на каждом узле, в эту коллекцию записываются все изменения данных, необходимые для воспроизведения операции.

Механизм тактовых сигналов позволяет обеспечить отработку отказа и выбор нового первичного узла. По умолчанию каждый член набора реплик посылает тактовые сигналы всем остальным членам раз в две секунды. Это позволяет в целом отслеживать состояние системы. Если узел перестает отвечать и является первичным, то система автоматически выбирает новый первичный узел из вторичных с самым свежим состоянием.

Процесс настройки репликации в MongoDB состоит из следующих шагов:

1) Создание каталогов для серверов и инициализация серверов. Пример команды инициализации одного сервера:

```
$> mongod --replSet mdbRep --dbpath c:\data\rs1 --port 27017
```

2) Объединение серверов в один набор реплик. Для этого необходимо создать конфигурацию, в которой перечислены все члены, и отправить ее одному из созданных процессов mongod, который передаст эту конфигурации другим членам:

```
rsconf = {
    _id: "mdbRep",
    members: [
        { _id: 0, host: "localhost:27017" },
        { _id: 1, host: "localhost:27018" },
        { _id: 2, host: "localhost:27019" }
    ]
}
rs.initiate(rsconf)
```

3) Разрешения чтения данных из вторичных узлов с использованием команды `secondaryOk`:

```
rs.secondaryOk()
```

2. Сегментирование

Сегментирование (sharding) — подход, предполагающий разделение баз данных, отдельных её объектов или индексов поисковых систем на независимые сегменты, каждый из которых управляется отдельным экземпляром сервера базы данных, размещаемым, как правило, на отдельном вычислительном узле. Таким образом, если при секционировании разными сегментами таблицы управлял один сервер, то в сегментировании сегменты данных управляются разными серверами.

Сегментирование в MongoDB позволяет создавать кластер из множества серверов и распределять коллекцию по ним, поместив подмножество данных в каждый сегмент.

Одна из целей сегментирования – сделать так, чтобы для клиентского приложения кластер из нескольких сегментов выглядел как единый сервер. Чтобы скрыть эти детали от приложения, перед сегментами запускается один или несколько процессов маршрутизации под названием `mongos`. Процессы `mongos` хранят так называемое «оглавление таблиц», которое сообщает им, в каком сегменте какие данные содержатся. Приложения могут подключаться к этому маршрутизатору и отправлять запросы в обычном режиме. «Маршрутизатор», зная, какие данные на каком сегменте находятся, может пересылать запросы соответствующему сегменту или сегментам. При получении ответов, маршрутизатор собирает их и, если необходимо, объединяет и отправляет обратно в приложение. Таким образом, с точки зрения приложения, оно подключено к автономному серверу, а вся реализация распределенного хранения данных от него

скрыта. При этом каждый сегмент может быть также реплицирован по нескольким серверам, т.е. каждый сегмент может храниться не на одном автономном сервере, а в наборе реплик.

Таким образом, сегментированный кластер MongoDB состоит из следующих компонентов:

- Непосредственно сегмент, который содержит подмножество данных коллекции. Каждый сегмент можно развернуть как набор реплик.
- Процесс mongos, который действует как маршрутизатор запросов, предоставляя интерфейс для взаимодействия клиентского приложения с сегментированным кластером.
- Серверы конфигурации, которые хранят метаданные и параметры конфигурации для кластера.

Горизонтальное масштабирование подразумевает распределение набора данных и нагрузки по нескольким узлам СУБД. Для этого распределения данных по сегментам используется так называемый ключ сегментирования. Сущности, связанные одинаковым значением ключа сегментирования, группируются в набор данных по заданному ключу. Этот набор данных хранится в пределах одного физического сегмента, что существенно облегчает обработку данных. Таким образом, если известен ключ сегментирования некоторого объекта, то всегда можно ответить на вопросы: «где следует сохранить данные?» и «Где найти запрошенные данные?».

MongoDB использует ключ сегментирования для распределения документов коллекции по сегментам. Ключ состоит одного или нескольких полей документа. Чтобы сегментировать заполненную коллекцию, она должна иметь индекс, начинающийся с ключа сегментирования. При сегментировании пустой коллекции MongoDB создает поддерживающий индекс, если в коллекции еще нет соответствующего индекса для указанного ключа сегмента.

Выбор ключа сегментирования влияет на производительность, эффективность и масштабируемость сегментированного кластера. Кластер с наилучшим аппаратным обеспечением и инфраструктурой может оказаться узким местом из-за выбранного ключа сегментирования.

MongoDB поддерживает две стратегии сегментирования:

- Ranged — разделение данных на непрерывные диапазоны;
- Hashed — разделение данных на основе хеш-функции.

Еще одно понятие, используемое в MongoDB для сегментирования данных – чанки. MongoDB разделяет сегментированные данные на чанки, являющиеся единицами, которые MongoDB использует для перемещения данных. Каждый процесс mongo всегда должен знать, где найти документ, учитывая его ключ сегментирования. Теоретически MongoDB может отслеживать, в каком сегменте находится каждый документ, но в случае с коллекциями с миллионами или миллиардами документов это становится затратно. Поэтому MongoDB группирует документы в чанки по некоторому диапазону ключа сегментирования. Чанки всегда хранятся в одном сегменте, что позволяет MongoDB хранить небольшую таблицу связей чанк-сегмент.

Чтобы добиться равномерного распределения фрагментов по всем сегментам в кластере, в фоновом режиме запускается балансировщик для переноса чанков между сегментами.

Сегментирование по хешу основывается на вычислении хеша значения поля, входящего в ключ сегментирования. Каждому чанку назначается диапазон хеш-значений ключа сегментирования. MongoDB автоматически вычисляет хеши при анализе запросов с использованием хешированных индексов. Приложениям не нужно вычислять хеши.

Хотя диапазон ключей сегментирования может быть «близким», их хешированные значения вряд ли будут находиться в одном и том же чанке. Сегментирование на основе хешированных значений способствует более равномерному распределению данных, особенно в наборах данных, где ключ сегментирования изменяется монотонно.

Однако хешированное распределение означает, что запросы по диапазону значений с меньшей вероятностью будут нацелены на один сегмент, что приводит к большему количеству широковещательных операций в масштабе кластера.

Сегментирование по диапазонам включает разделение данных на диапазоны на основе значений ключа сегментирования, а не на основе хеша от этих значений. Каждому чанку назначается диапазон на основе значений ключа сегментирования.

Диапазон ключей сегментирования, значения которых «близки», с большей вероятностью будет находиться в одном и том же чанке. Это позволяет выполнять операции, содержащие условия поиска по ключу сегментирования, к конкретному сегменту или сегментам, которые содержат необходимые данные.

Эффективность сегментирования по диапазонам зависит от выбранного ключа сегмента. Плохо продуманные ключи сегментов могут привести к неравномерному распределению данных, что может свести на нет некоторые преимущества сегментирования или вызвать снижение производительности.

Настройка сегментирования в MongoDB состоит из следующих шагов:

1) Создание каталогов для серверов конфигурации и инициализация серверов. Помимо инициализации из командной строки, параметры запуска сервера можно указать в файле конфигурации. Для конфигурационного сервера указывается роль `configsrv`, например:

```
storage:
     dbPath: E:\mongo\cfg1\
net:
     port: 27018
     bindIp: 127.0.0.1  #<имя хоста | ip-адрес(а)>
sharding:
     clusterRole: configsvr
replication:
     replSetName: cfg_replica_set  #<имя набора реплик>
```

Для инициализации серверов с использованием файла конфигурации можно воспользоваться командой:

```
$> mongod --config <path-to-config-file>
```

2) Настройка набора реплик для конфигурационного сервера. Для этого необходимо подключиться к одному из экземпляров конфигурационного сервера с использованием утилиты `mongosh`:

```
$> mongosh --host <hostname> --port <port>
```

Далее необходимо выполнить инициализацию набора реплик с использованием метода `rs.initiate`, указав параметр `configsvr:true`:

```
rs.initiate({
    _id: "имя набора реплик",
    configsvr: true,
    members: […]
})
```

3) Создание каталогов для сервера сегмента и инициализация серверов. Для серверов данного типа в файле конфигурации необходимо установить параметр `clusterRole: shardsvr`.

4) Настройка набора реплик для сегмента. Выполняется аналогично настройке набора реплик для конфигурационного сервера (шаг 2) без указания параметра `configsvr`.

5) Создание процесса-маршрутизатора `mongos`. Для его запуска используется файл конфигурации, в котором указывается имя набора реплик сервера конфигурации и по крайней мере один из члена набора реплик сервера конфигурации:

```
net:
     port: 27027
     bindIp: 127.0.0.1  #<имя хоста | ip-адрес(а)>
sharding:
     configDB:
cfg_replica_set/127.0.0.1:27018,127.0.0.1:27019
```

```
$> mongod --config E:\mongo\mongos.cfg
```

6) Подключение к маршрутизатору:

```
$> mongosh --host 127.0.0.1 --port 27027
```

и добавление сегментов в кластер:

```
sh.addShard("shard1_replica_set/localhost:27021,localhost:27022,localhost:27023")
```

7) Сегментирование коллекции:

```
sh.shardCollection("<database>.<collection>", { <shard key field> : type } )
```

Для проверки статуса можно выполнить команду `sh.status`, которая отобразит информацию о настройках конфигурации, сегментах, активных экземплярах процесса-маршрутизатора, состоянии балансировщика нагрузки, а также сегментированных базах данных.

ЛР6. Вопросы для контроля

1. Репликация. Типы репликации.
2. Репликация с одним ведущий сервером.
3. Репликация с несколькими ведущими серверами.
4. Репликация без ведущих серверов.
5. Репликация в MongoDB.
6. Сегментирование. Основные понятия в MongoDB.
7. Компоненты кластера MongoDB.
8. Сегментирование в MongoDB с использованием хеш-функции.
9. Сегментирование в MongoDB по диапазонам.

ЛР6. Задание

1) Настроить сегментирование в MongoDB. Тестовая конфигурация кластера должна содержать:
   - процесс-маршрутизатор mongos;
   - конфигурационный сервер, запущенный на наборе из трех реплик;
   - двух сегментов, каждый из которых запущен на наборе из трех реплик.
2) Выполнить сегментирование двух коллекций:
   - для одной коллекции настроить сегментирование с использованием хеш-функции;
   - для другой – выполнить сегментирование по диапазонам.
3) Отобразить статус кластера, выполнить запросы получения данных к каждой из коллекций.