**Репликация и шардирование в MongoDB**

**1. Введение в репликацию и шардирование**

Современные приложения требуют высокой доступности и отказоустойчивости данных. Репликация обеспечивает дублирование данных между несколькими серверами. Шардирование используется для горизонтального масштабирования больших объёмов данных.

**2. Репликация в MongoDB**

**Принципы репликации**

Репликация в MongoDB создаёт набор серверов (репликационныйнабор) для обеспечения:

Доступности данных: резервирование данных.

Отказоустойчивости: автоматическое переключение на резервный сервер при сбое.

**Компоненты репликационного набора**:

1. Первичная реплика (primary): принимает все операции записи.
2. Вторичные реплики (secondary): синхронизируют данные с первичной реплики.
3. Арбитр: сервер, не хранящий данных, но участвующий в голосовании при переключении ролей.

**Пример настройки репликационного набора**

Инициализация репликационного набора:

mongod --replSet "rs0" --port 27017 --dbpath /data/db1 --bind\_ip localhost

Конфигурация репликационного набора:

rs.initiate({

  \_id: "rs0",

  members: [

      { \_id: 0, host: "localhost:27017" },

      { \_id: 1, host: "localhost:27018" },

      { \_id: 2, host: "localhost:27019" }

  ]

});

Проверка статуса:

rs.status();

**2.3. Особенности репликации**

Автоматическое переключение (failover): при сбое primary другая реплика становится primary.

Чтение из вторичных реплик:

db.getMongo().setReadPref("secondary");

1. **Шардирование в MongoDB**

**Принципы шардирования**

Шардирование позволяет масштабировать данные горизонтально путём разделения их на несколько серверов (шардов). Каждый шард содержит часть данных коллекции.

Компоненты шардированной архитектуры:

Шарды: серверы, хранящие данные.

Mongos: маршрутизатор запросов.

Конфигурационные серверы: информация о структуре шардов.

**Настройка шардирования**

Запуск конфигурационного сервера:

mongod --configsvr --port 27019 --dbpath /data/configdb --replSet configReplSet

Инициализация репликационного набора конфигурационного сервера:

rs.initiate();

Запуск шардов:

mongod --shardsvr --port 27018 --dbpath /data/shard1

mongod --shardsvr --port 27017 --dbpath /data/shard2

Добавление шардов:

sh.addShard("shard1/localhost:27018");

sh.addShard("shard2/localhost:27017");

Включение шардирования для базы данных:

sh.enableSharding("myDatabase");

Настройка ключа шардирования:

db.myCollection.createIndex({ field: 1 });

sh.shardCollection("myDatabase.myCollection", { field: 1 });

**Особенности шардирования**

Данные распределяются по шардирующему ключу.

Балансировка: автоматическое перераспределение данных между шардами.

Масштабируемость: увеличение количества шардов по мере роста данных.

**4. Обеспечение отказоустойчивости и доступности данных**

Резервирование данных через репликацию.

Масштабирование запросов через шардирование.

Автоматический failover: система продолжает работу даже при сбое одного из серверов.

Мониторинг и управление:

* + mongostat: мониторинг производительности.
  + mongotop: просмотр активных операций.

**5. Примеры использования**

Репликация

Пример: резервирование базы данных интернет-магазина. Если primary сервер недоступен, все операции автоматически перенаправляются на новую primary.

Шардирование

Пример: обработка больших объёмов данных для геолокационных сервисов. Данные распределяются по регионам, что снижает нагрузку на сервер.

**6. Преимущества и недостатки**

Репликация

Преимущества:

* Высокая доступность.
* Минимальные задержки чтения при настройке чтения из secondary.

Недостатки:

* Увеличение объёма хранимых данных.

Шардирование

Преимущества:

* Горизонтальное масштабирование.
* Улучшение производительности для больших данных.

Недостатки:

* Сложность настройки и управления.
* Зависимость производительности от правильного выбора шардирующего ключа.