Онлайн синхронизация данных в распределённой системе

В. В. Осокин, М. В. Шегай

Рассматривается задача синхронизации данных и кода между серверами, физически расположенными в различных местах. Реализована система репликации базы данных, настроена система контроля версий, а также система репликации содержимого каталогов.

Ключевые слова: система контроля версий, репликация баз данных, git, unison.

Введение

В связи с увеличением числа пользователей веб-сервисов возникает необходимость в распределении нагрузки между несколькими серверами. Ввиду того, что пользователи разнесены географически, для уменьшения задержки (latency) на обработку запросов пользователей, имеет смысл использовать сервер, наиболее приближенный к ним.

Для обеспечения одинакового пользовательского опыта, в независимости от того, какой сервер используется, нужно поддерживать в одинаковом состоянии данные и код.

Данные представлены записями в таблицах базы данных и файлами, используемыми веб-сервисом.

Кодом будем считать исходный код, обеспечивающий работу сервиса.

Веб-сервис использует СУБД MySQL 5 [1], php фреймворк Yii [2], javascript фреймворк AngularJS [3].

Для синхронизации данных и кода использовались система контроля версия Git [5], система синхронизации файлов unison [4] и средства репликации MySQL.

Постановка задачи и полученные результаты

Ставится задача настройки системы синхронизации данных и кода между двумя серверами, один из которых расположен в Санкт-Петербурге, а другой — в Ташкенте. Требуется обеспечить одинаковый пользовательский опыт при работе с обоими серверами.

В рамках поставленной задачи получены следующие результаты.

Настроена система синхронизации кода.

При внесении изменений в код разработчик может незамедлительно отправить изменения на оба сервера. Помимо этого система контроля версий позволяет вести учёт различных версий проекта, облегчает совместную разработку, позволяет отслеживать изменения кода, предоставляет возможность переходить к различным версиям проекта.

Настроена система репликации базы данных.

Пользователь получает одинаковые данные от веб-сервиса, независимо от того, какой сервер используется.

Репликация базы данных происходит в автоматическом режиме.

Была достигнута приемлемая скорость синхронизации: не более 350мс при одновременном использовании сервиса 53 пользователями (при предполагаемой пиковой нагрузке в 40–45 одновременно активных пользователей, что является хорошим показателем для веб-сервиса данного масштаба).

Настроена система синхронизации содержимого каталога, содержащего пользовательские файлы.

В независимости от того, какой сервер используется, пользователь имеет доступ к одинаковым файлам: изображениям, видео роликам и др.

Файлы синхронизируется в автоматическом режиме сразу после внесения изменений.

Синхронизация данных

Как уже упоминалось выше синхронизация данных состоит из двух частей: синхронизации таблиц базы данных и синхронизации файлов. Для синхронизации таблиц базы данных использовались

средства репликации MySQL, а для синхронизации файлов — приложение unison.

Репликация базы данных

При типичном сценарии работы пользователя с нашим веб-сервисом, более 90% запросов к базе данных являются запросами на чтение, и менее 10% приходится на запросы на изменение (создание, удаление, изменение записей).

В связи с этим схема синхронизации базы данных нашего веб-сервиса выглядит следующим образом:

1) запросы на чтение отправляются ближайшему серверу
2) запросы на изменение отправляются главному серверу, который в дальнейшем будем называть `master`
3) при поступлении запроса на изменение к `master` серверу, `master` сервер отправляет соответствующий запрос на изменение вторичному серверу, который в дальнейшем будем называть репликой или `slave`

Такая схема позволяет поддерживать таблицы в базах данных, находящихся на различных серверах, в одинаковом состоянии и обеспечить приемлемую скорость обработки запросов.

Рассмотрим пример настройки системы репликации базы данных MySQL.

Предположим, что IP-адрес `master` 192.168.1.1, реплики — 192.168.1.2. Требуется реплицировать базу данных `example`.

Настройки `master`

Откроем файл конфигурации MySQL `/etc/mysql/my.cnf`

Укажем в нём ID сервера, путь для бинарных логов и имя БД для репликации в разделе `[mysqld]`:

```
server-id = 1
log-bin = /var/lib/mysql/mysql-bin
replicate-do-db = example
```

Создадим пользователя `slave_user` с правами `replication slave`:

```
mysql@master> GRANT replication slave
      ON "example".*
      TO "slave\_user"@"192.168.1.1
      "IDENTIFIED BY "password";
```

Перезапустим MySQL:

```
sudo service mysqld restart
```

Настройки реплики

Откроем файл конфигурации `MySQL /etc/mysql/my.cnf`

Укажем в нём ID сервера, путь для бинарных логов, путь к relay-бинарным логам и имя БД для репликации в разделе `[mysqld]`:

```
server-id = 2
relay-log = /var/lib/mysql/mysql-relay-bin
relay-log-index = /var/lib/mysql/mysql-relay-bin.index
replicate-do-db = example
```

Перезапустим MySQL:

```
sudo service mysqld restart
```

Перенос данных

Для того чтобы перенести данные нужно заблокировать базу данных для изменений, для этого установим флажок `read_only` на `master`:

```
mysql@master> SET GLOBAL read_only = ON;
```

Введем команду:

```
mysql@master> SHOW MASTER STATUS;
```

Запомним значения `File` и `Position`

`File: mysql-bin.000001`

`Position: 101`

`File` — файл, из которого реплика будет реплицировать данные.

`Position` — позиция, начиная с которой реплика будет реплицировать данные.

Сделаем дамп базы данных `example`, и снимем блокировку:

```
mysqldump -u root -p --opt example > example.sql
mysql@master> SET GLOBAL read_only = OFF;
```

На реплике нужно создать соответствующую базу данных и заполнить её дампом базы данных, полученным с `master` сервера:

```
mysql@slave> CREATE DATABASE example;
mysql@slave> EXIT;
mysql -u root -p example < /path/to/example.sql
```

Запустим репликацию:

```
mysql@slave> CHANGE MASTER TO MASTER_HOST = "192.168.1.1 ",
      MASTER_USER = "slave_user",
      MASTER_PASSWORD = "password ",
      MASTER_LOG_FILE = "mysql-bin.000001",
      MASTER_LOG_POS = 101;
mysql@replica> start slave;
```

Значения `MASTER_LOG_FILE` и `MASTER_LOG_POS` — это значения `File` и `Position`, соответственно, взятые с `master`.

Синхронизация файлов

Как уже говорилось, для синхронизации файлов используется приложение `unison`.

Unison — кросс-платформенное приложение для двусторонней синхронизации файлов, которое позволяет синхронизировать две копии каталогов, расположенных на локальном компьютере или на двух разных серверах, обновляя каждую копию в зависимости от произведённых изменений. При этом среди серверов не выделяется главный. Так выглядит файл конфигурации unison:

```
root = /local/dir/to/sync
root = ssh://username@server.address//remote/dir/to/sync

# автоматически применять действия по-умолчанию (неконфликту-
# ющие)
auto=true
```

```
# пользовательский интерфейс не будет задавать
# никаких вопросов. Неконфликтующия изменения будут примене-
# ны, конфликтующие - пропущены

batch=true

# Unison будет использовать размер файла и время последнего
# изменения во время сканирования реплик для обнаружения
# изменений, вместо чтения содержимого каждого файла

fastcheck=true

# Синхронизировать атрибуты группы (group attributes)
group=true

# Синхронизировать владельца файла (owner)
owner=true
```

Для автоматического запуска синхронизации при изменении содержимого на обоих серверах устанавливается утилита `incrond` со следующей строчкой в файле конфигурации:

```
/path/to/sync/dir IN\_CLOSE_WRITE,IN_CREATE,IN_DELETE \
/usr/bin/unison &> /dev/null
```

Синхронизация кода

При внесении изменений в код разработчик должен иметь возможность легко отправлять свои изменения на рабочий сервер. Для этого в рабочем процессе используется система контроля версий `Git`.

В целом, `Git` обладает большими возможностями для облегчения совместной разработки, позволяет вести учёт версий и многое другое. Здесь же рассмотрим возможности `Git` для синхронизации кода между серверами и локальными компьютерами разработчиков.

Центральным объектом `Git` является репозиторий. Репозиторий `Git` представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описыва-

ющий расположение файлов и хранилище, содержащее собственно файлы.
Создадим центральный репозиторий на сервере, для этого нужно создать каталог для репозитория:

```
mkdir /path/to/repo.git
git init --bare
```

Далее разработчик должен скопировать (склонировать) себе этот репозиторий:

```
git clone ssh://server.address/path/to/repo.git
```

После этого разработчик может вносить локально изменения в каталоге скопированного репозитория.
Для отправления изменений на сервер создаётся `commit`:

```
git add <files_to_commit>
git commit -am "commit comment"
git push
```

Для получения изменений от сервера используется команда `git pull`. Для работы веб-сервиса так же создадим копию репозитория командой `git clone` и будем получать изменения командой `git pull`. Таким образом происходит синхронизация кода.

Заключение

Была решена задача синхронизации данных и кода между двумя серверами, один из которых расположен в Санкт-Петербурге, а другой — в Ташкенте. При этом был обеспечен одинаковый пользовательский опыт в независимости от того, с каким именно сервером работает пользователь, с сохранением приемлемой скорости работы.

Веб-сервис можно найти по адресам: http://dvinemnauku.ru и http://anzolis.com.

Список литературы

[1] http://www.mysql.com/

[2] http://www.yiiframework.com/
[3] http://angularjs.org/
[4] http://www.cis.upenn.edu/ bcpierce/unison/
[5] http://git-scm.com/
[6] Chacon S. Pro Git. — ISBN 978–1430218333.
[7] http://dev.mysql.com/doc/refman/5.0/en/replication.html
[8] http://habrahabr.ru/post/56702/