UrsaJ: HTTP File Storage

Кузьма Деретюк
kuzma.deretuke@gmail.com

Введение: план

- Какую проблему решаем
- Файловое хранилище
- Зачем это вам
- Демо
- Вопросы

Какую проблему решаем?

Как веб приложение работает с файлами?

- Загружает/отдаёт через сервер приложений
- Хранит в файловой системе приложения
- Все операции вне транзакций

Хранить файлы в приложении

- Велосипед для управления файлами
- I/O нагрузка на сервер приложений
- Решение плохо масштабируется
- Сложно сохранить консистентность при резервном копировании / восстановлении

Хранить данные в базе данных

- Стандартный драйвер и язык SQL запросов
- Нагрузка на сервер базы данных
- Решение гибко масштабируется
- Гарантированная консистентность

Хранить файлы в файловом хранилище

- Удобное API для управления файлами
- I/O нагрузка на сервер файлового хранилища
- Решение гибко масштабируется
- Гарантированная консистентность

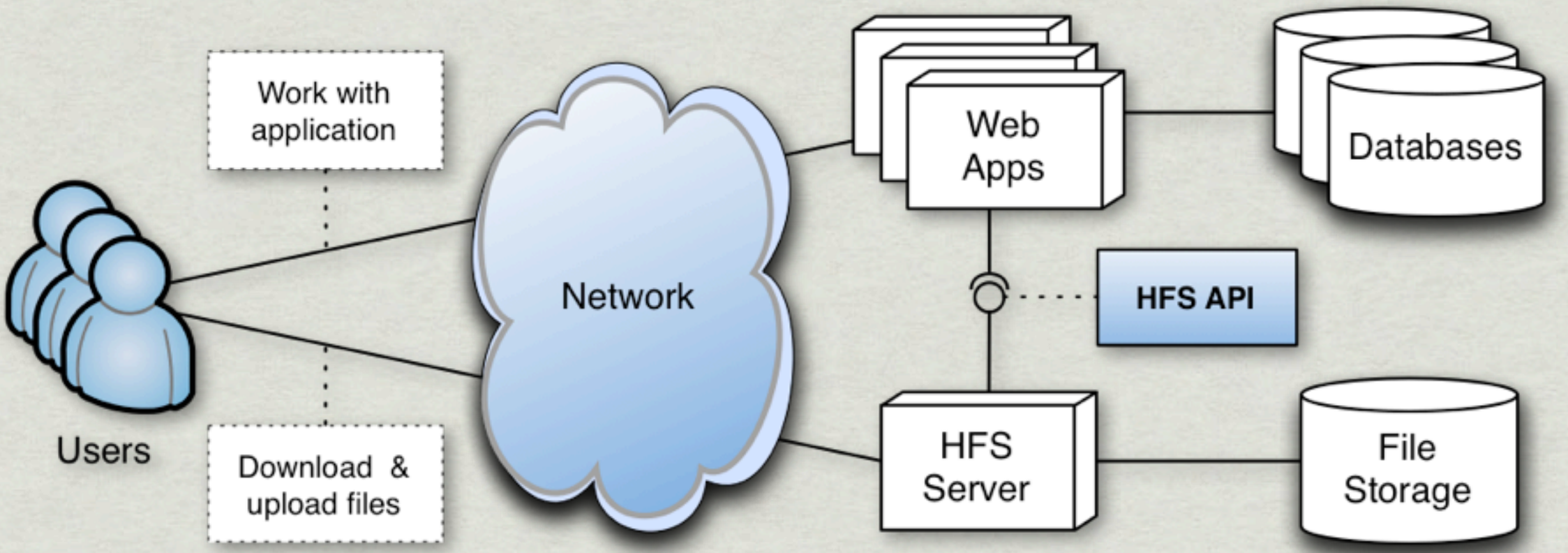
Файловое хранилище

UrsaJ: HFS

Файловое хранилище UrsaJ: HFS

- Обеспечивает загрузку и публикацию файлов
- Упрощает разработку веб приложений
- Работает в вашей инфраструктуре

Инфраструктура



Коммуникация: токены

- Токен - упакованное, подписанное сообщение

token = Base64.encode(Cipher.sign(message))

- Обмен токенами - через агент пользователя

Консистентность систем

- Файл появляется, когда сохранён в файловом хранилище: ID + Info + Expiration => Token
- Агент пользователя передает на сервер приложений токен загруженного файла
- Ваше приложение регистрирует загруженный файл в базе данных: Token => Upload Info
- Сборщик мусора проверяет актуальность хранимых файлов

Сборщик мусора

- **Коллекция файлов упорядочена по ID**
- GC опрашивает сервер приложений о живых файлах в заданном диапазоне
- GC обходит все файлы хранилища в заданном диапазоне
- Перемещает в корзину мертвые файлы

Client API

```
interface HfsClient {

    String createUploadUri();

    String createPublishUri(UUID fileId, String fileName);

    HfsUploadInfo parseUploadInfo(String token);

    HfsGcRequest parseGcRequest(String token);

}
```

Конфигурация файлового хранилища

```
<bean class="com.ursaj.hfs.store.HfsDefaultStore">

    <property name="hosts" value="localhost, 127.0.0.1, 192.168.1.1"/>

    <property name="homeDir" value="webapps/ROOT"/>

    <property name="privateKey" value="s3cret"/>

    <property name="cleanupUri" value="http://app.uri/cleanup?token="/>

</bean>
```

Зачем это вам

Разработчику

- Не надо заботиться о сохранении файлов: файлы уже загружены, сохранены и доставлены вам в виде ID + Meta
- Не надо заботиться об удалении файлов: просто сотрите запись о нём в базе данных
- Не надо заботиться о доставке файлов: опубликуйте ссылку на файл по его ID

Администратору

- Гибкость: один сервер может обслуживать несколько хранилищ
- Масштабируемость: несколько серверов могут обслуживать одно хранилище
- Простота: модель консистентности упрощает создание и восстановление резервных копий

Менеджеру

- Разработка приложений становится проще, а значит быстрее и дешевле
- IT инфраструктура легче балансируется за счёт более узкой специализации приложений

Демо

Вопросы

Контакты

- UrsaJ: HTTP File Storage
  http://ursaj.com
- Кузьма Деретюк
  kuzma.deretuke@gmail.com