Опыт использования простой системы передачи сообщений

Роман Друзягин, Иван Фролков

Передача сообщений

“Очереди – это если очень высокая нагрузка”

Нет, не только

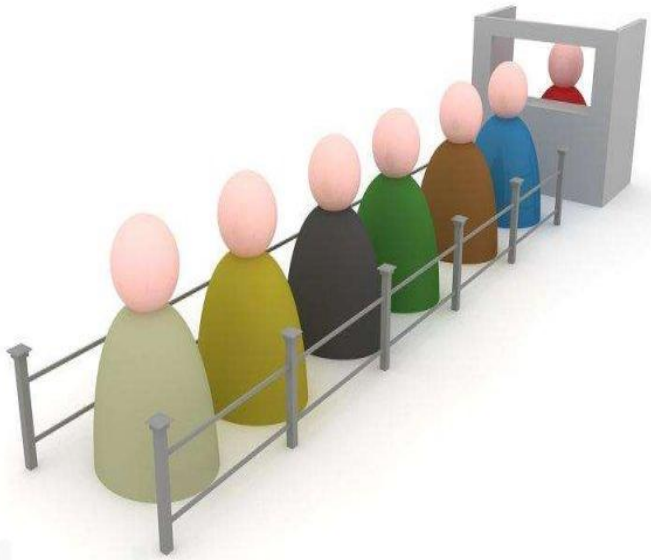
1. Собственно, очереди (queues) – каждому потребителю свое сообщение.
2. И доски объявлений (topics) – одно сообщение всем потребителям.
    a. durable consumers

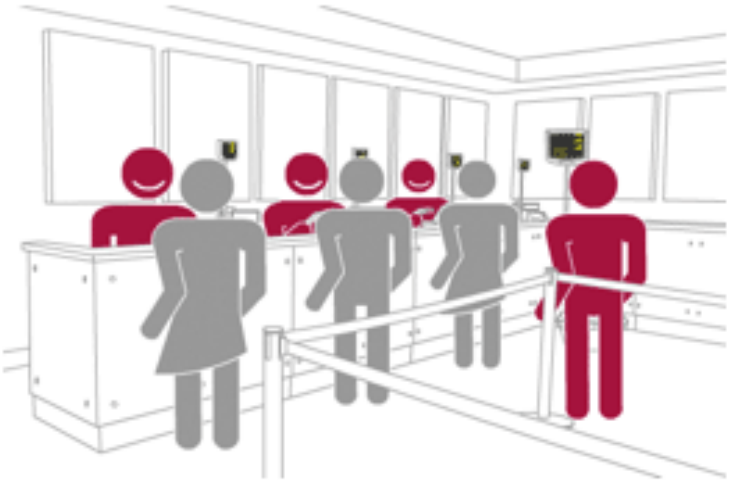
Зачем это надо?

- Асинхронность
- Контролируемый поток данных
- Слабая связность. Выстрелил и забыл.
- Возможность выполнять очень медленный код.

Очередь

Что это?





Очередь

Асинхронность и параллелизм: сообщения из очереди могут поступать в произвольном порядке и обрабатываться параллельно и независимо.

Реализации

1. Отдельные продукты – WebsphereMQ, ActiveMQ, HornetQ и др.

2. Встроенные в СУБД – SQL Server, Oracle AQ в Oracle.

3. JMS/JEE и сервера приложений.
   a. Spring.

4. JMS 2.0.
   a. Shared consumers.

Реализации

Достоинства отдельных продуктов – гетерогенность, универсальность, интеграция.

Недостатки – требуется дополнительная логика для доступа в/из СУБД. 2PC/идемпотентность.

Реализации

Достоинства встроенных – единая транзакционная модель, отсутствие отдельного выделенного сервера.

Недостатки – вопросы с гетерогенностью, интеграцией, в некоторых случаях транзакционность привносит дополнительные накладные расходы.

mbus и postgres

- **Простая** система передачи сообщений.
- Очереди и топики.
- Нет внешних зависимостей
  (за исключением стандартного hstore).

mbus за 20 секунд

Создаем очередь и работаем с ней:

```
create extension mbus;
select mbus.create_queue('queue',64);
select mbus.post('queue', hstore('key','value'));
select * from mbus.consume('queue');
```

mbus

```
create_queue(qname, parallel_consumers)
```

По сути, это топик с одним подписчиком (`'default'`).

Создаем другого:

```
create_consumer(qname, cname)
```

и читаем отдельно:

```
select * from mbus.consume(qname, cname)
```

mbus

Вот, собственно, и все введение.

Все очень просто.

Как же это работает?

Задача

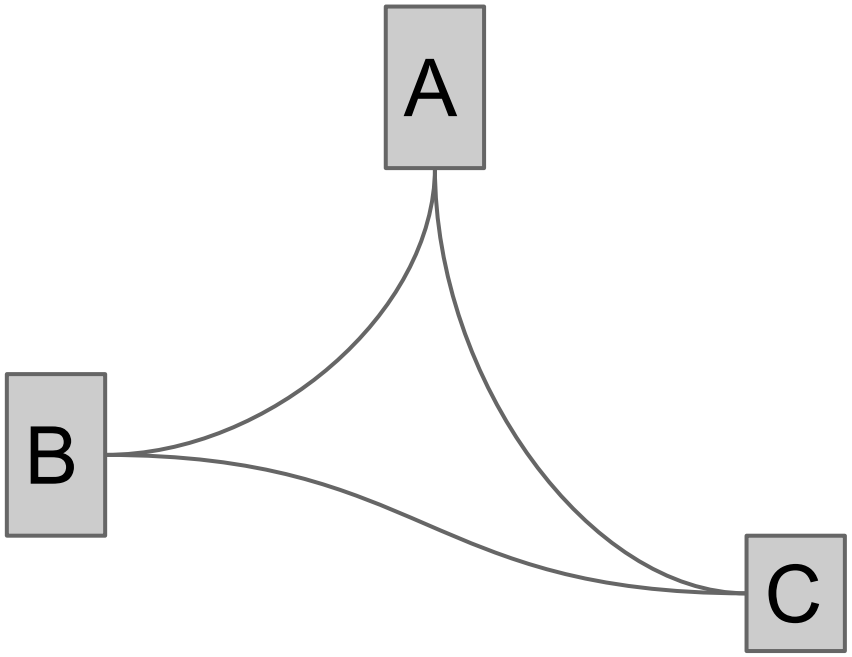
Есть три ж/д станции – A, B, C. На каждой продают билеты на любой участок на поезд от A до C.

Требуется реализовать систему продажи билетов с ожидаемыми требованиями – продать максимальное количество билетов на поезд без ошибок (два и более билета на одно место).

Связь между станциями **ненадежна.**

Решение

Постоянный обмен сообщениями



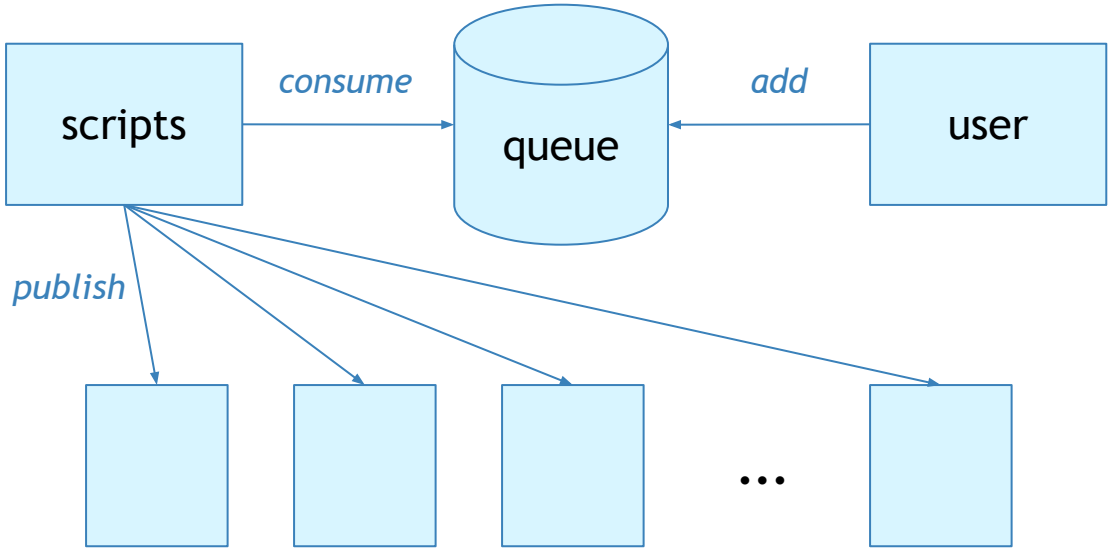
Задачи из практики

От мала до велика. От тривиальных вещей – к сложным.

Типовые задачи (без нагрузок):

1. отправка почты,
2. on-line чат с пересылкой сообщений в CRM,
3. разливка “контента” на кластер из N серверов,
4. периодические пересчеты чего-нибудь для кого-нибудь,
5. формирование поисковых выдач, построение “горячих” таблиц, расчет баллов за “ачивки” и многое другое в проекте соц. сети.

Задачи из практики



1. Server off-line? Delayed until, изображение готово к раздаче.
2. Server non-responsive? Delayed until, изображение отложено.

Дополнительные возможности

1. Селекторы для потребителей.
2. Временные очереди.
    a. Request-Reply не получится :-(
3. Порядок сообщений.

Порядок сообщений

1. Сообщения не всегда независимы.

2. Некоторые сообщения могут быть обработаны только после других и не ранее.
   a. Например, создать пользователя и внести ему деньги на счет.

3. Oracle – DBMS_AQ SEQUENCE_DEVIATION
   a. DBMS_AQ.BEFORE

Вместо или вместе?

1. Прекрасно интегрируется с чем угодно с помощью Apache Camel.

2. Вебсокеты в хранимые процедуры, вебсервисы из триггеров и т.д.

3. “Смутное ощущение грандиозных возможностей”.

4. EIP (http://www.enterpriseintegrationpatterns.com/).

Вопросы?