Introduction to JMS & ActiveMQ

Aaron Mulder
Chariot Solutions

Agenda

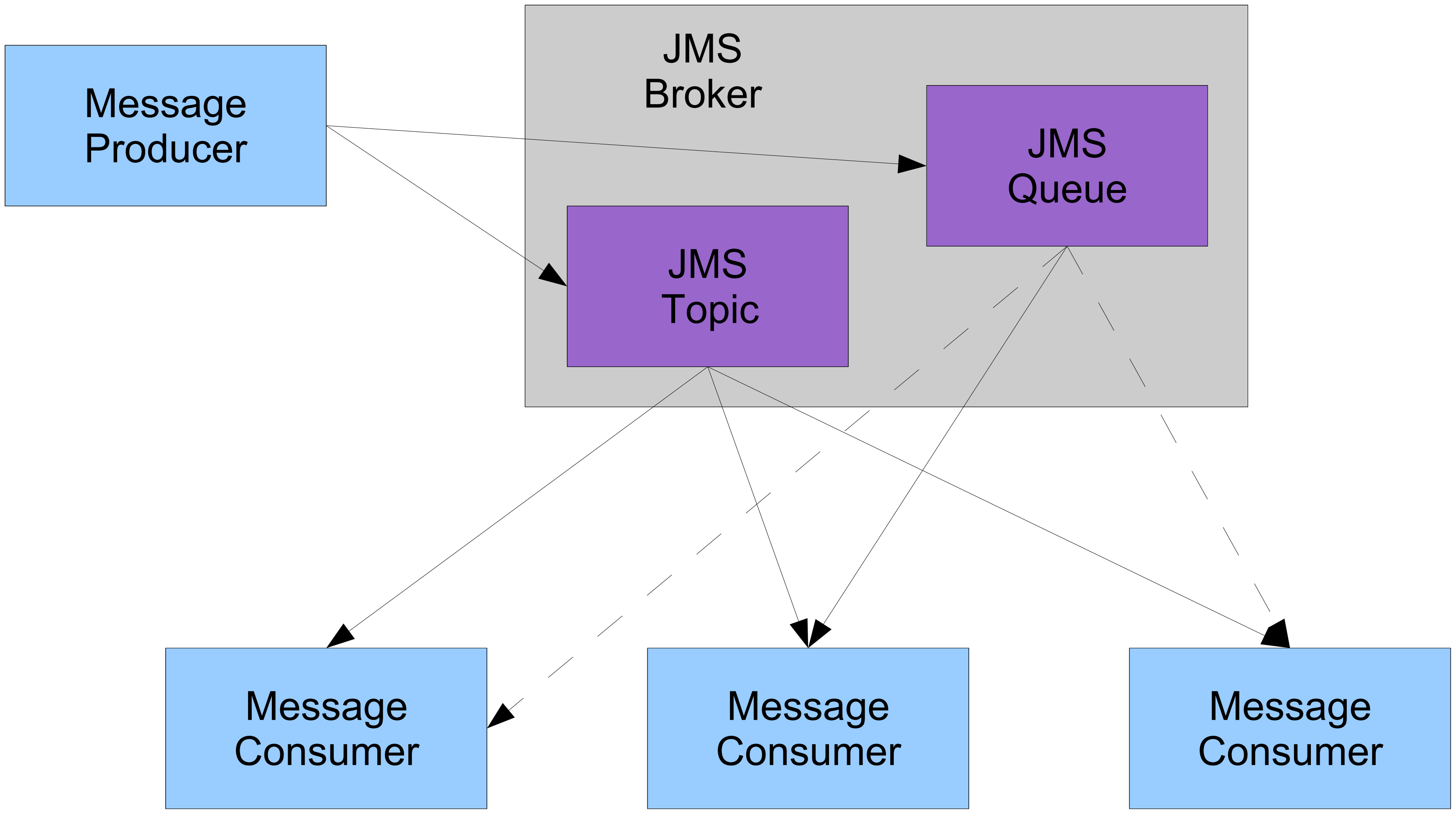
- Quick intro to JMS
- ActiveMQ Basics
- ActiveMQ Clustering

JMS

About JMS

- The Java API for messaging
- Included in Java EE
- Generally used for asynchronous operations, or to parallelize or throttle a bunch of work
- Key concepts include the message broker, message producers, message consumers, JMS topics vs. queues, and various message formats
- Includes a plain Java API (a little complex); simplified in Java EE, Spring, etc.

JMS Flow


Message Producer
JMS Broker
JMS Queue
JMS Topic
Message Consumer
Message Consumer
Message Consumer

Some Minutiae

- Both producers and consumers may use transactions
  - But the transaction only encompasses the exchange between the producer and broker or consumer and broker; it's not end-to-end
- Messages may be persistent (saved to e.g. disk in case of broker crash)
- Topic subscribers may be durable (if they disconnect and reconnect, they'll get the messages sent while they were offline)

JMS Messages

- Composed of headers and a body
- The headers are name/value pairs, and a consumer may filter on header values
  - Or could just use separate topics/queues instead
- The body is different for different types of messages, but most common is the text message with e.g. text, XML, YaML, etc.
- May request an acknowledgement or reply to a different destination (topic or queue)

ActiveMQ

About ActiveMQ

- An open-source message broker (compare to JBossMQ, or many commercial products)
  - See http://activemq.apache.org/
- Generally stable and high-performance
- Can be run standalone, or inside another process, app server, or Java EE application
- Supports everything JMS requires, plus various extensions
- Can also extend with e.g. Camel, ServiceMix

ActiveMQ Protocols/Formats

- Generally there are two main options – OpenWire (binary) and Stomp (text)
  - OpenWire is the default and has the most history and best support (including SSL)
  - Stomp is easiest to develop for and therefore has the most cross-language support (Perl, Python, Ruby, ...)
    - ActiveMQ 5 recommended for best Stomp support
- Also a variety of other special-purpose protocols (Jabber, adapters for REST/AJAX, etc.)

Additional Features

- Security (SSL and/or username/password required to connect)
- Management (JMX interface to the broker, as well as Web Console)
- JMS Extensions (Virtual Destinations, Retroactive Subscriptions, Exclusive Consumer & Message Groups, Mirrored Queues, ...)
- Various persistence implementations for persistent messages

ActiveMQ Configuration Example

```
<beans ...>
  <broker xmlns="http://activemq.org/config/1.0"
brokerName="MyBroker" dataDirectory="${activemq.base}/data">
    <transportConnectors>
       <transportConnector name="openwire"
                           uri="tcp://localhost:60010" />
       <transportConnector name="stomp"
                           uri="stomp://localhost:60020"/>
    </transportConnectors>
    <networkConnectors>
      <networkConnector name="Broker1ToBroker2"
                        uri="static://(tcp://localhost:60011)"
                        failover="true" />
    </networkConnectors>
  </broker>
</beans>
```

CHARIOT SOLUTIONS

Clustering

- Two clustering strategies:
  - Master/Slave(s) – best reliability, no improved scalability
  - Network of Brokers – best scalability, better availability, somewhat improved reliability
- Network of Brokers is best if you can live with the side effects
  - Messages may be delivered twice or substantially delayed (also out of order) in a failure scenario
  - Messages may be lost if a broker dies for good

Performance Example

- One application we benchmarked had topics with few messages but many (~500) consumers.
  - A single broker and network of brokers performed similarly, averaging about ½ to 1 second latency
  - A master/slave cluster was slower with 1.5-2 second latency
  - Smaller messages were delivered faster (1k twice as fast as 10k)
  - Persistence only matters if messages are not rapidly consumed
  - VM made a difference

CHARIOT SOLUTIONS

Performance Example, con't.

- Results would be different for:
  - better hardware
  - queues
  - more producers / fewer consumers
  - different messages
  - message selectors
  - durable subscriptions
  - security options
  - transactions
  - etc.

Q&A