Advanced Messaging with Apache ActiveMQ

Bosanac Dejan

May 2011

FuseSource
A Progress Software Company

About me

- Bosanac Dejan
- Senior Software Engineer at FUSESource – http://fusesource.com
- Apache ActiveMQ committer and PMC member
- Co-author of ActiveMQ in Action

FuseSource
A Progress Software Company

What we are going to cover?

- What is ActiveMQ
- The Basics
- Flow control
- Scaling
- High Availability
- Future
- Conclusion

What is ActiveMQ?

FuseSource
A Progress Software Company

Apache ActiveMQ

- Apache ActiveMQ
  - Leading Open Source messaging platform
  - Supported Java Standards:
    - JMS 1.1, J2EE 1.4, JCA 1.5 and XA
- Reliable, high performance messaging
  - Out-performs many legacy proprietary message queues
  - Configurable for many different deployments
- Multi-Protocol/Multi-Language Support

FuseSource

A Progress Software Company

Background

- ActiveMQ started in 2005 at CodeHaus
- Moved to Apache Software Foundation in 2006
- 1,117,537 lines of code
- 24 committers
- Now the most widely used open source messaging system on the planet

The Basics

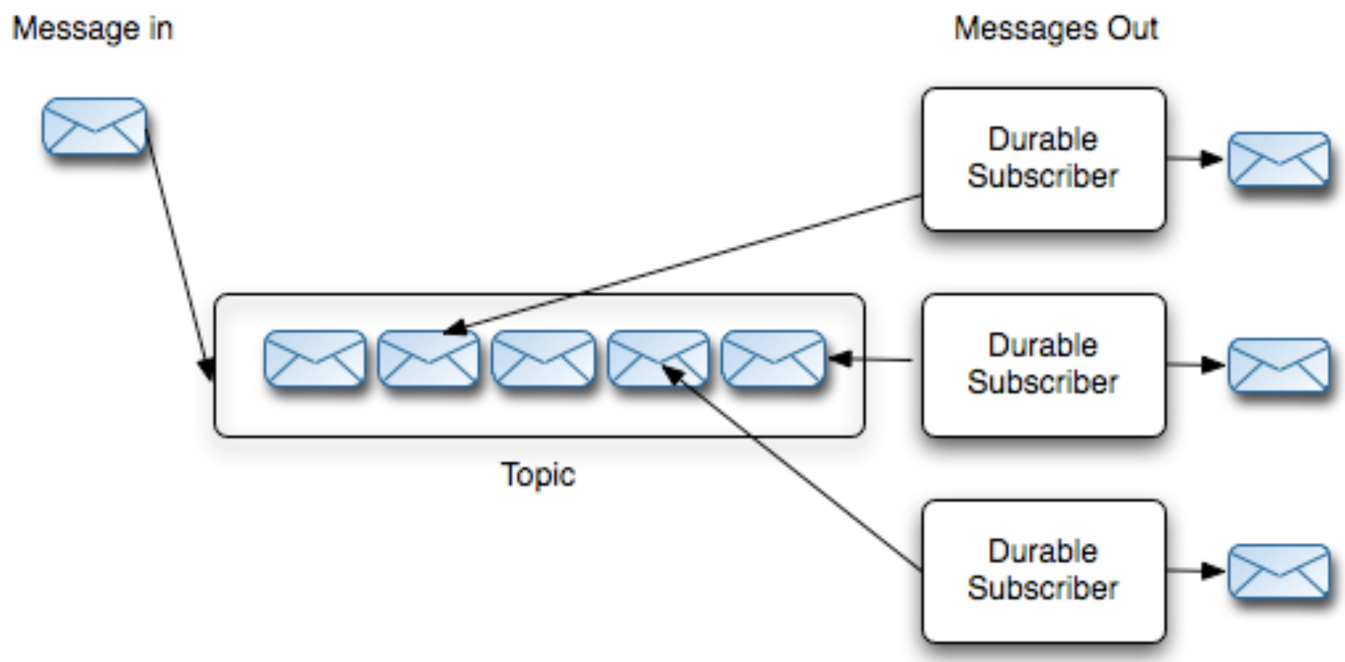
FuseSource

A Progress Software Company

Messaging is

- Loosely coupled exchange of messages between applications
- Location transparency
- Can be persistent or non-persistent
- Can be transactional

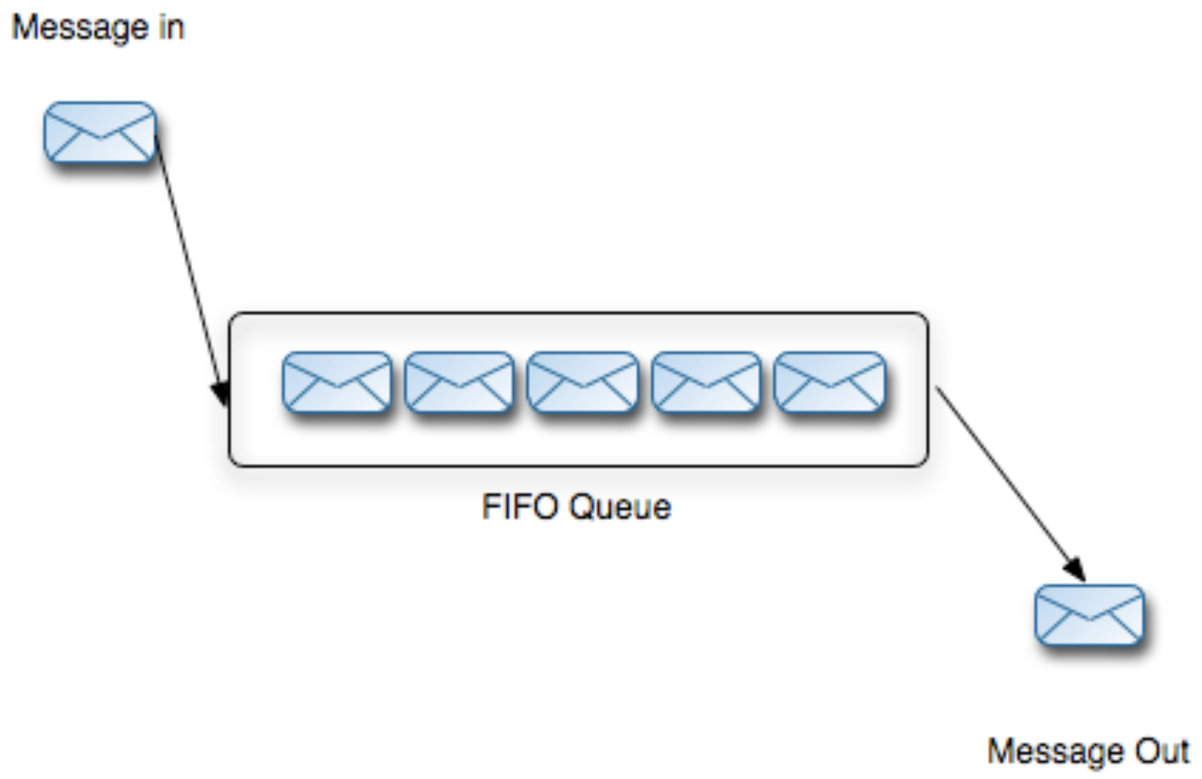
Topics


Message in
Messages Out
Durable Subscriber
Durable Subscriber
Durable Subscriber
Topic

Topics

- One message goes to 0-to-many consumers based on the current subscribers
- Think like mailing lists or discussion forums
- Ideal for publishing business events
- Distributed observer pattern
- Allows one part of your system to notify anyone else who may be interested in an event

Queues



Queues

- Messages are load balanced across many consumers
- Each message goes to exactly one consumer
- Consumers compete for messages
- Its easy to browse and monitor queues
- Ideal for grid style applications

FuseSource

A Progress Software Company

Challanges

- Create a general messaging platform
- Support variety of use-cases
  - Large number of clients
  - Large number of destinations
  - Slow consumers
- Provide enterprise feaures
  - Security
  - High availability
  - Management
  - etc

Flow Control

FuseSource
A Progress Software Company

Flow Control – Why?

- Dealing with deep queues
- Dealing with slow consumers
- We want to prevent broker from being flooded with messages
- We want to prevent broker running out of memory and other resources

FuseSource

A Progress Software Company

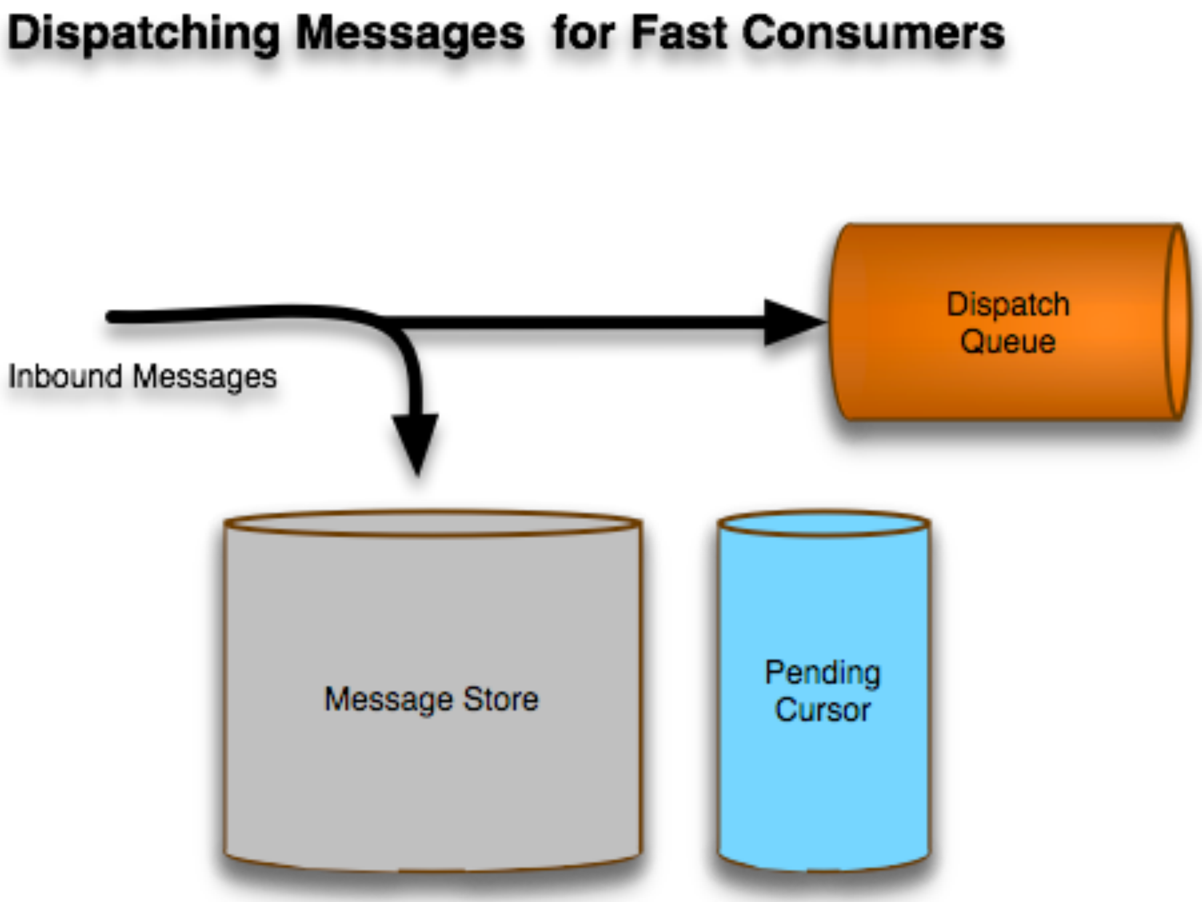
Flow Control – How?

- Message Cursors
- Producer Flow Control

FuseSource

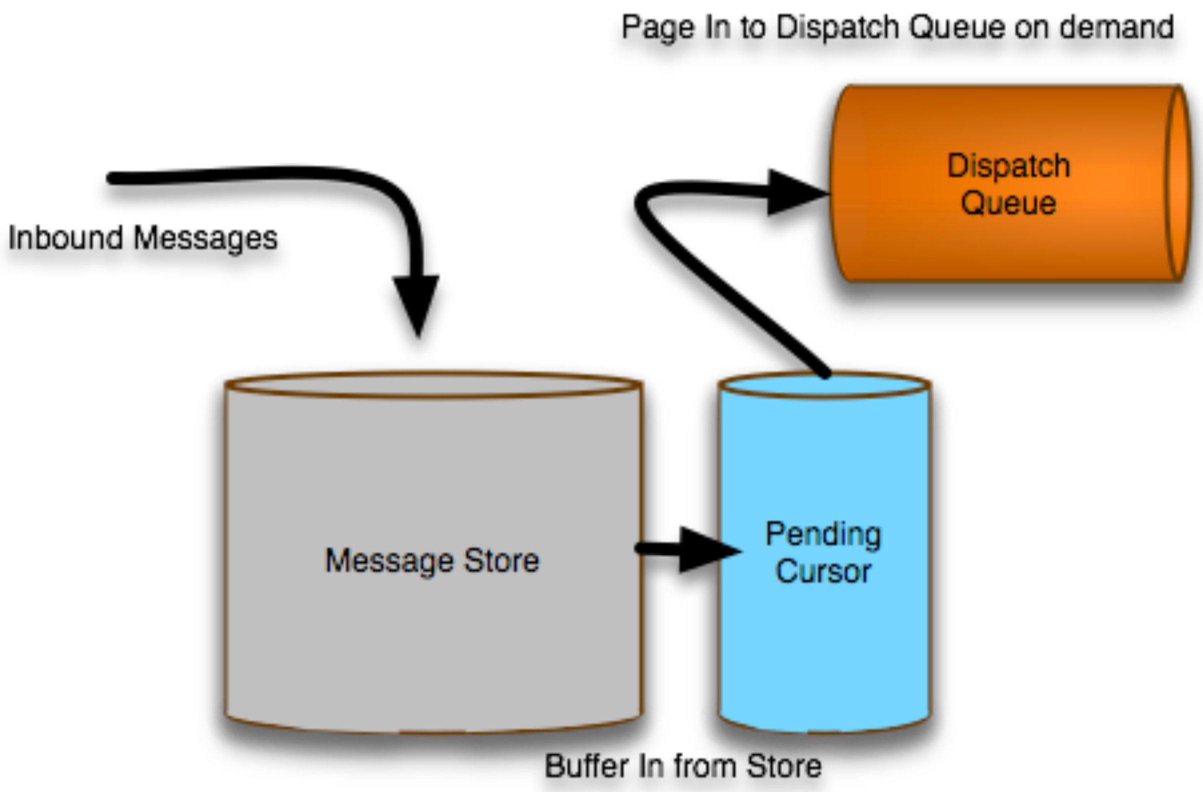
A Progress Software Company

Flow Control – Cursors

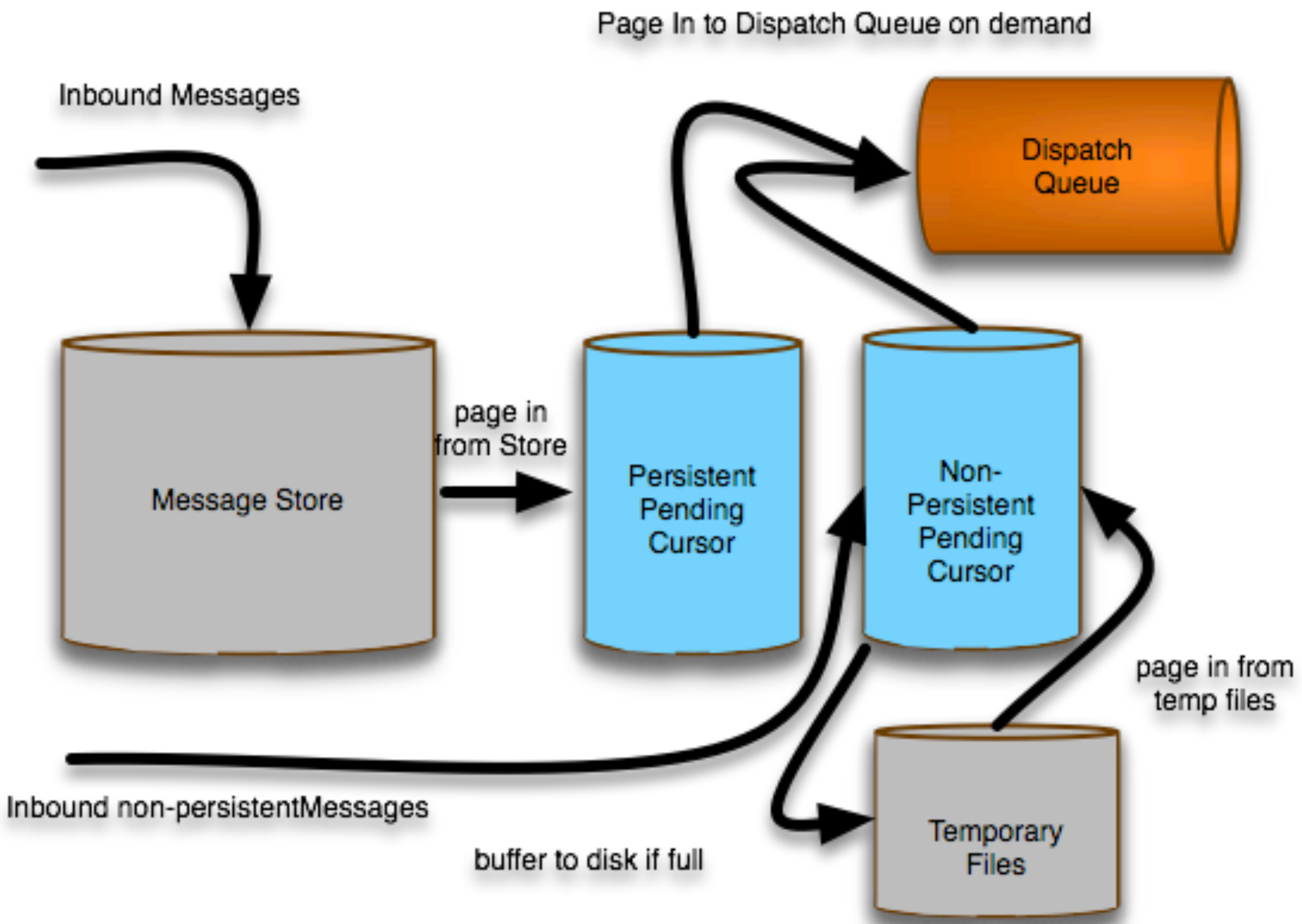


Flow Control – Store-based Cursor

Dispatching Messages if Dispatch Queue is Full



Flow Control – Non-persistent cursor



FuseSource

A Progress Software Company

Flow Control – Limits

Per destination

```
<destinationPolicy>
      <policyMap>
        <policyEntries>
          <policyEntry queue=">" memoryLimit="10mb"/>
        </policyEntries>
      </policyMap>
</destinationPolicy>
```

System settings

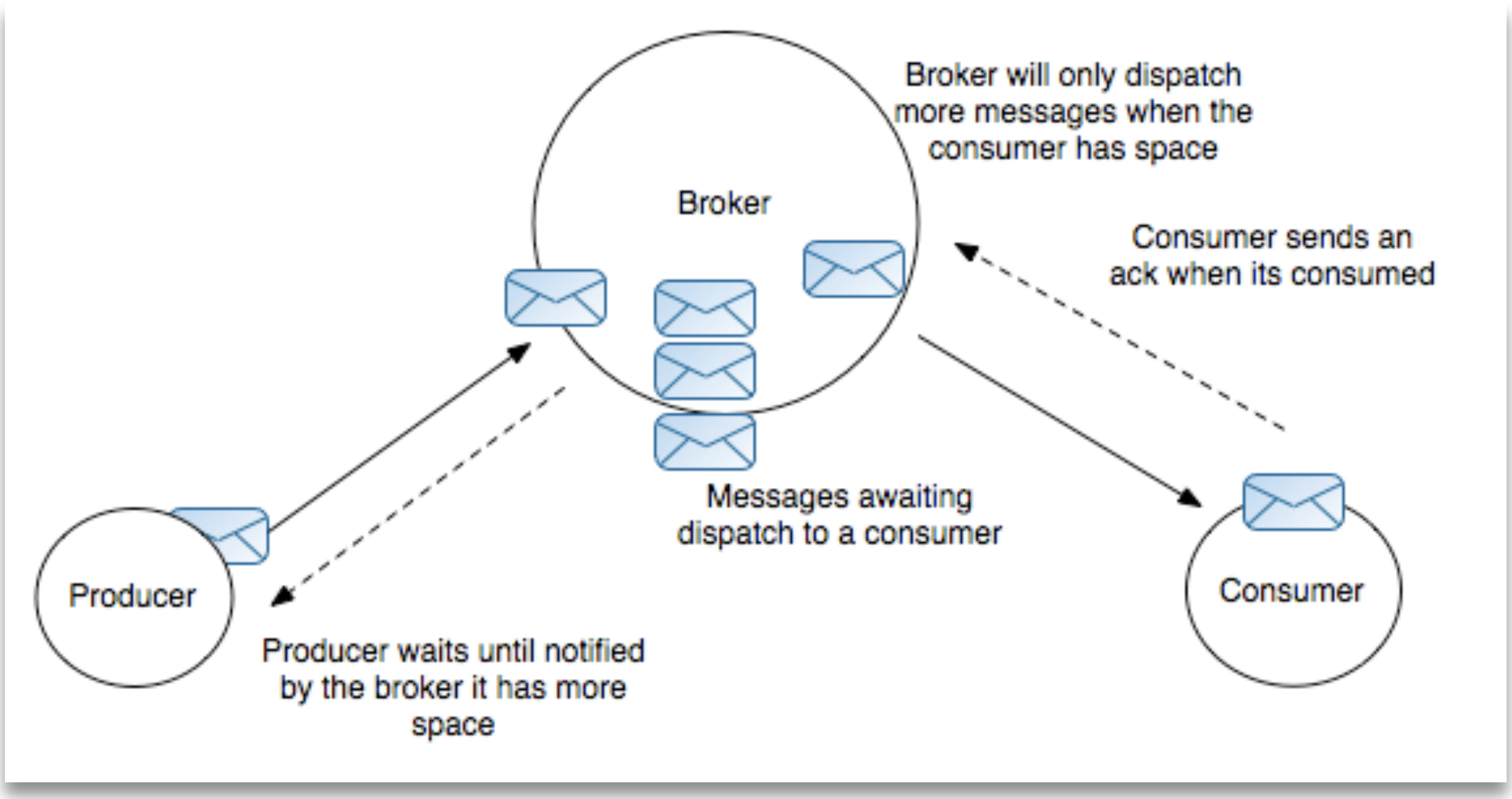
```
<systemUsage>
  <systemUsage>
    <memoryUsage>
      <memoryUsage limit="256 mb" />
    </memoryUsage>
    <storeUsage>
      <storeUsage limit="100 gb" />
    </storeUsage>
    <tempUsage>
      <tempUsage limit="10 gb" />
    </tempUsage>
  </systemUsage>
</systemUsage>
```

FuseSource

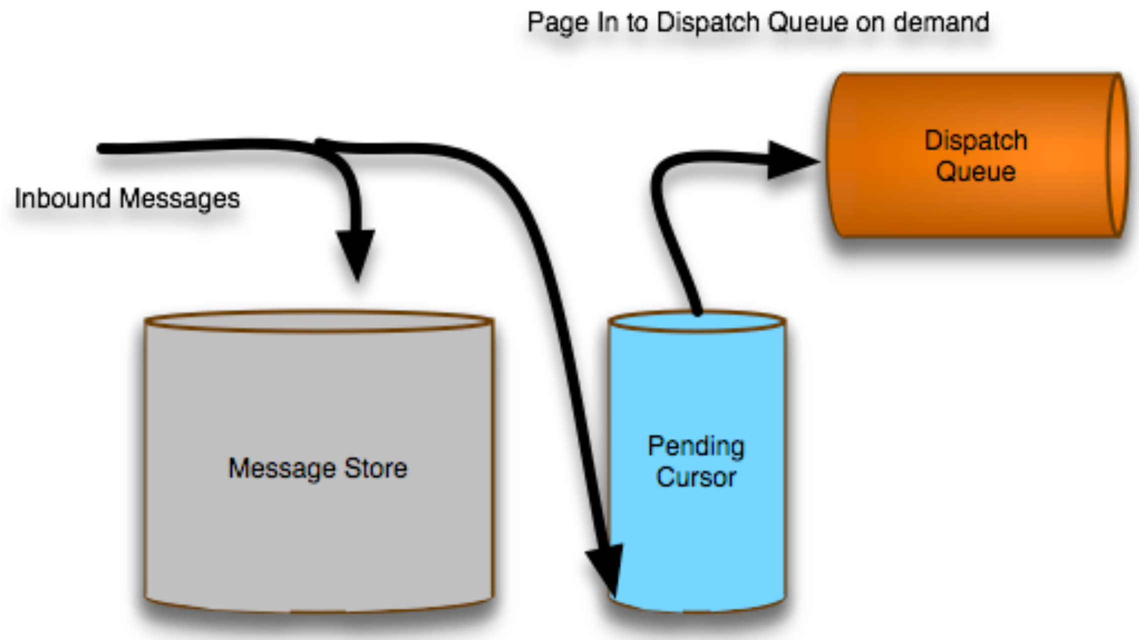
A Progress Software Company

Flow Control – Producer Flow Control

- Throttling producer speed to the speed of consumers


Broker will only dispatch more messages when the consumer has space
Broker
Consumer sends an ack when its consumed
Messages awaiting dispatch to a consumer
Producer
Consumer
Producer waits until notified by the broker it has more space

Flow Control – VM Cursor



```
<policyEntry queue=">" producerFlowControl="true" memoryLimit="1mb">
  <pendingQueuePolicy>
    <vmQueueCursor/>
  </pendingQueuePolicy>
</policyEntry>
```

FuseSource
A Progress Software Company

Scaling

FuseSource
A Progress Software Company

Scaling – Types

- Vertical scaling
- Horizontal scaling
- Traffic partitioning

Vertical Scaling

Increase load capacity using a single broker can handle.

Problems:

- Thread count
- Memory usage
- CPU usage

FuseSource

A Progress Software Company

Vertical Scaling – Threads

What are threads used for

- For Connections – Thread per Connection (blocking transport)
- For Dispatching – Thread per Destination

Vertical Scaling – Number of Connections

Use non-blocking transport

```
<transportConnectors>
  <transportConnector name="nio" uri="nio://0.0.0.0:61616"/>
</<transportConnectors>
```

Enables handling large number of clients

Vertical Scaling – Number of Destinations

Don’t use dedicated task runner

```
ACTIVEMQ_OPTS="-Dorg.apache.activemq.UseDedicatedTaskRunner=false"
```

Use optimized dispatch for queues

```
<destinationPolicy>
      <policyMap>
        <policyEntries>
          <policyEntry topic=">" optimizedDispatch="true">
          ...
          </policyEntry>
        </policyEntries>
      </policyMap>
</destinationPolicy>
```

FuseSource

A Progress Software Company

Vertical Scaling – Memory

- Give broker enough memory

```
ACTIVEMQ_OPTS="-Xmx2048M -Dorg.apache.activemq.UseDedicatedTaskRunner=false"
```

- Configure big enough memory usage

```
<systemUsage>
  <systemUsage>
    <memoryUsage>
      <memoryUsage limit="1024 mb" />
    </memoryUsage>

    ...

  </systemUsage>
</systemUsage>
```

FuseSource
A Progress Software Company

Vertical Scaling – CPU

- Disable tight encoding

It uses more CPU to create smaller packets

```
uri = "failover://(tcp://localhost:61616wireFormat.tightEncodingEnabled=false)";
```

FuseSource

A Progress Software Company

Vertical Scaling – Conclusion

There is a limit to the scalability a single machine can give

Horizontal Scaling

Increase load capacity using networked brokers

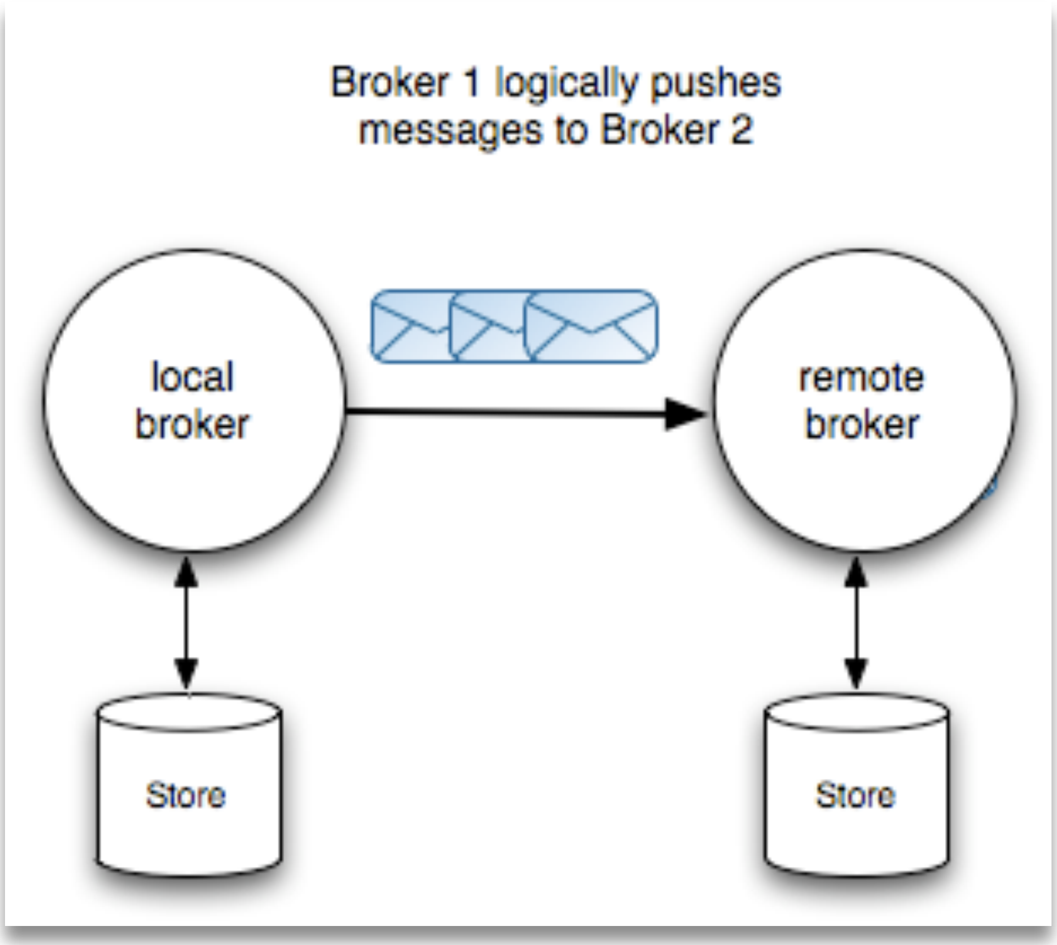
Concepts:

- Network of Broker

FuseSource

A Progress Software Company

Horizontal Scaling – Network of Brokers



Horizontal Scaling – NOB Usage

Configuration

```
<networkConnector name="broker1-broker2"
    uri="static:(tcp://broker2:61617)"
    dynamicOnly="true"
    prefetchSize="1000"
    conduitSubscriptions="true"
    decreaseNetworkConsumerPriority="true"
    suppressDuplicateTopicSubscriptions="true"
    networkTTL="3">
</networkConnector>
```

Connecting

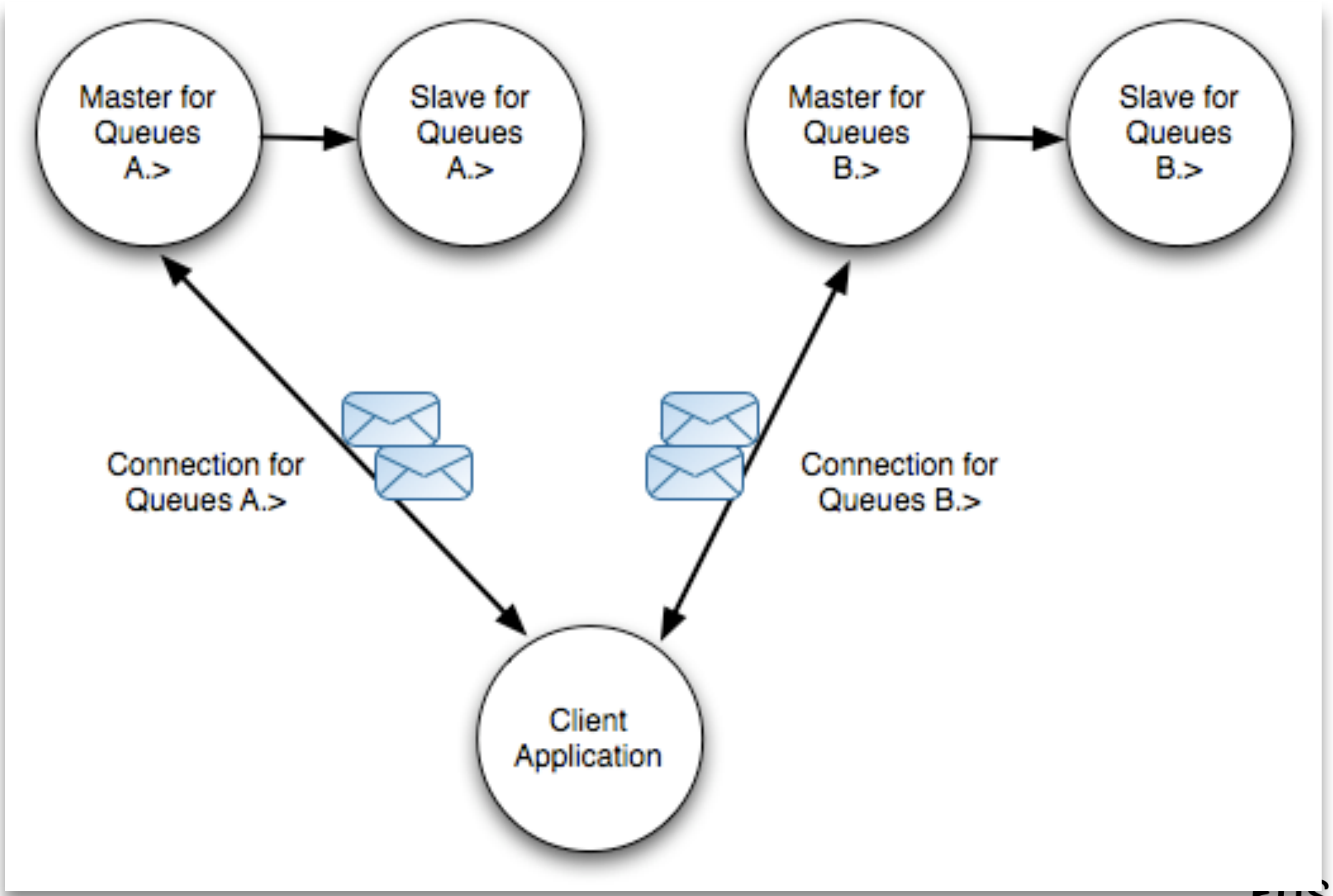
```
failover://(tcp://broker1:61616,tcp://broker2:61616)?randomize=true
```

Horizontal Scaling – Conclusion

- More latency in processing messages
- Beware of complex topologies

Hybrid Scaling

Partition traffic to more non-connected brokers



FuseSource
A Progress Software Company

Hybrid Scaling – Conclusion

- Pros
  - You can use all the tuning techniques used in Vertical scaling
  - Have better Horizontal scaleability than using Network Of Brokers (Less broker cross talk)
- Cons
  - Added complexity required on the end user Application

High Availability

FuseSource
A Progress Software Company

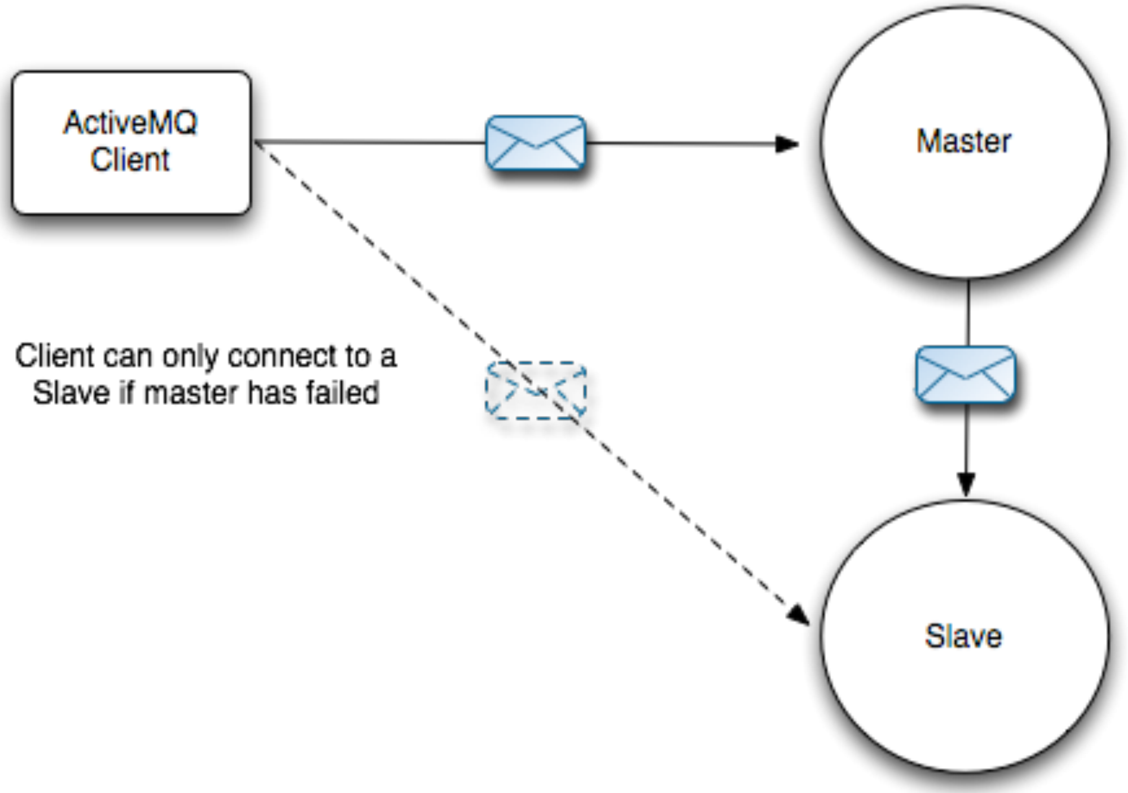
High Availability

- Pure Master/Slave
- JDBC Master/Slave
- Shared File System Master/Slave

FuseSource

A Progress Software Company

Pure Master-Slave



Client can only connect to a Slave if master has failed

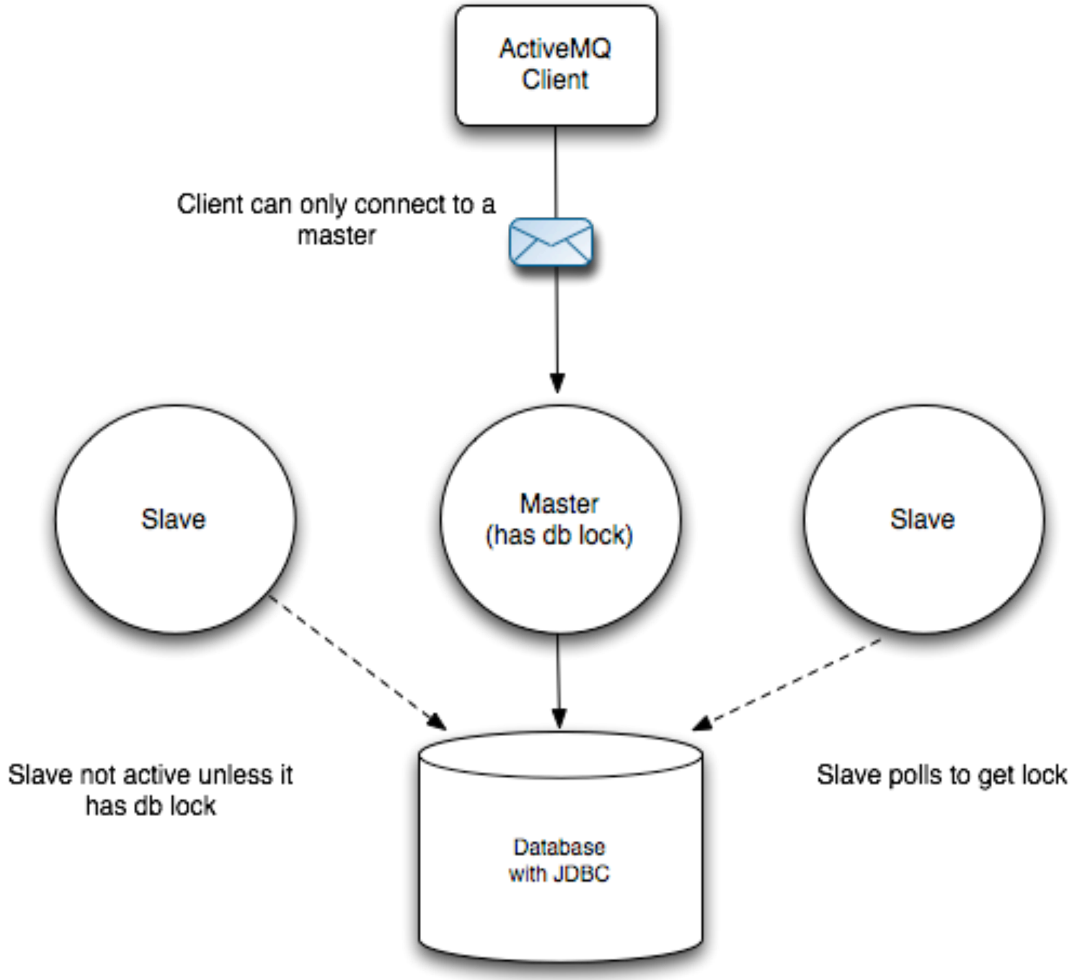
Slave gets full replicated state

FuseSource
A Progress Software Company

Pure Master-Slave

- Shared nothing
- Fully replicated
  - All messages
  - All acknowledgements
  - All transactions
- Slave does not start any transports or network connections

JDBC Master-Slave



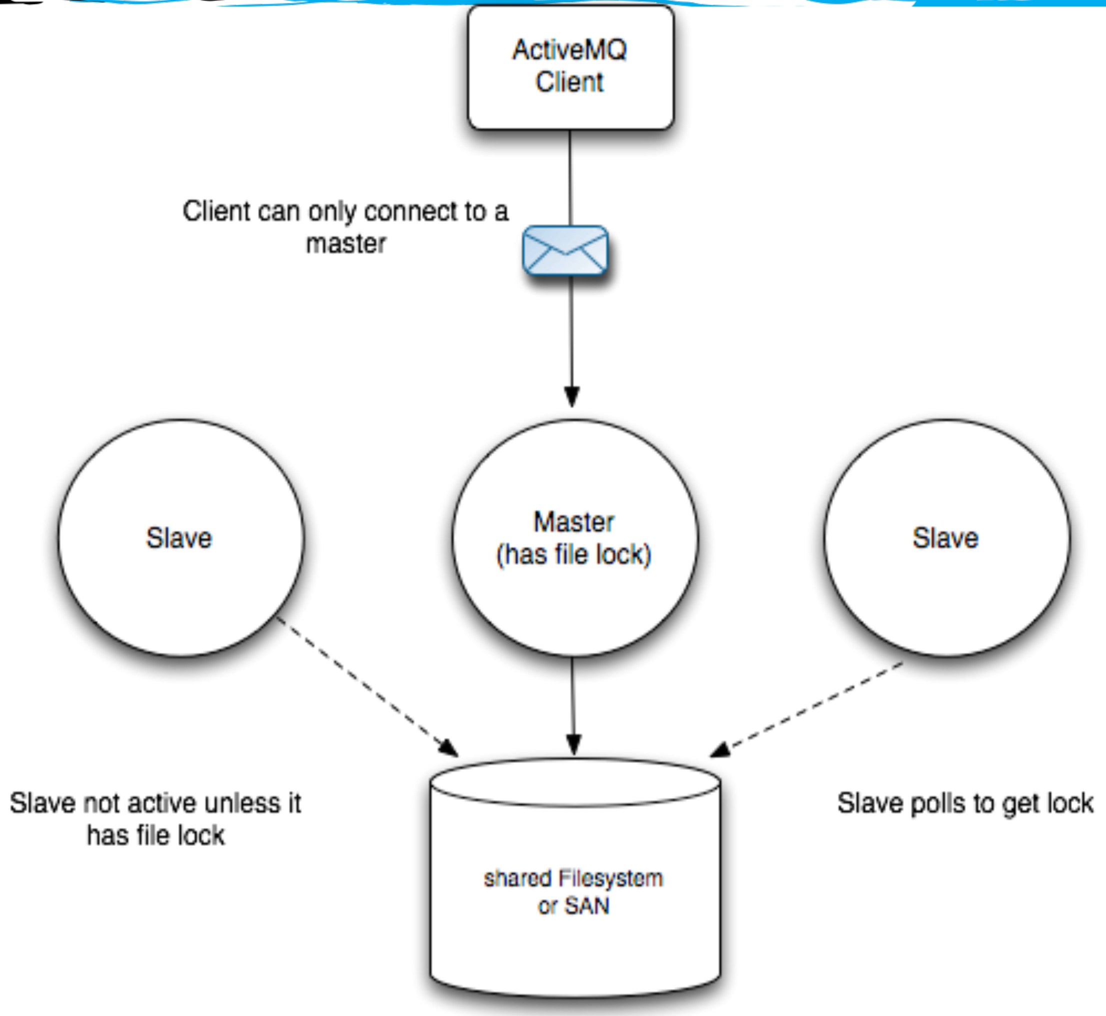
FuseSource

A Progress Software Company

JDBC Master-Slave

- Extreme reliability – but not as fast
- Recommended if already using an enterprise database
- No restriction on number of slaves
- Simple configuration

Shared Storage Master-Slave



Shared Storage Master-Slave

- Recommended if you have a SAN
- No restriction on number of slaves
- Simple configuration
- N.B. – ensure file locking works – and times out – NFSv4 good!

Future

FuseSource
A Progress Software Company

Future – ActiveMQ Apollo

- http://activemq.apache.org/apollo
- ActiveMQ 5.x reached scalability and performance limits with the current architecture
- New broker core

FuseSource

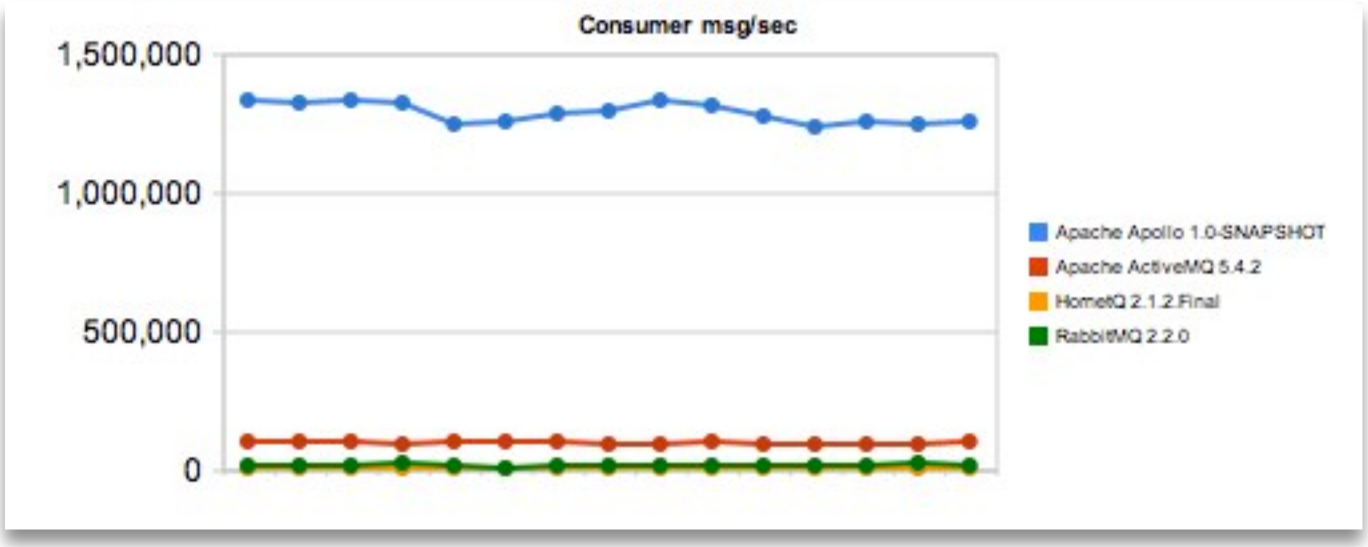
A Progress Software Company

Future – ActiveMQ Apollo

- Reactor Based Thread Model
- Scala 2.8 Implementation
- Protocol Agnostic
- REST Based Management

Future – ActiveMQ Apollo Performance

http://hiramchirino.net/blog



10 producers/10 consumers
single topic
using Stomp
20 byte payload

FuseSource

A Progress Software Company

Conclusions

- Dynamic community
- Leading in terms of messaging innovation
- Built for Enterprise
- Scalable, Good Performance, Reliable

Questions?

- ActiveMQ Web sites:
  - http://activemq.apache.org/
  - http://fusesource.com/products/enterprise-activemq/
- Blog:
  - http://www.nighttale.net/
- Twitter:
  - http://twitter.com/dejanb
  - http://twitter.com/fusenews

FuseSource

A Progress Software Company