Apache ActiveMQ Performance Tuning

Christian Posta
Senior Consultant and Architect
Blog: http://christianposta.com/blog/
Twitter: @christianposta
GitHub: https://github.com/christian-posta

FuseSource
integration everywhere

ceposta

- Christian Posta
- Blog: http://christianposta.com/blog/
- Email: ceposta@apache.org
- Twitter: @christianposta
- GitHub: https://github.com/christian-posta
- Goolge+: https://plus.google.com/103810850748178017238

- Senior Consultant and Architect at Red Hat (formerly FuseSource)
- Committer on: ActiveMQ, Apollo
- Author of *Essential Camel Components* DZone Refcard

What Am I Talking About?

- What is ActiveMQ?
- Performance/Benchmarking Tools
- Tuning

Apache ActiveMQ

- OpenSource Messaging Server
  - Queues, Topics, persistent messaging, high availability, etc
- Apache v2.0 License
- Active community, Mature, Stable
- Used at top companies
  - Insurance, banking, retail, ecommerce, health care, aviation, shipping, et. al!
- 5.8.0 latest release

FuseSource
integration everywhere

Apache ActiveMQ

- High performance
- High Availability
- Light-weight
- Multi-protocol
- JMS compliant
- Supported in Production

FuseSource integration everywhere | redhat

FuseSource integration everywhere

Accessible!

- Java
- C/C++ http://activemq.apache.org/cms/
- .NET http://activemq.apache.org/nms/
- PHP
- Python
- Ruby
- PHP
- JavaScript
- Telnet!
- Any that can make a TCP connection and send Text!

Transports

- TCP
- UDP
- SSL
- NIO
- HTTP/s
- VM
- WS (Web Sockets)
- WSS (Secure Web Sockets)

Wire Protocols

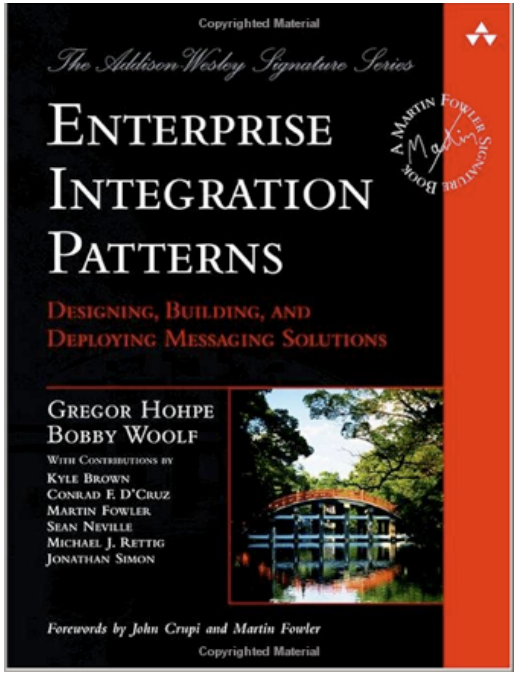
- Openwire
  http://activemq.apache.org/openwire.html
- Simple Text Oriented Messaging Protocol (STOMP)
  http://stomp.github.io
- AMQP 1.0
  http://www.amqp.org/resources/specifications
- MQTT
  http://mqtt.org
- HTTP/REST

ActiveMQ Features

- Master/Slave fault tolerance and High Availability
- Broker “networking” or clustering
- Pluggable persistence (KahaDB, LevelDB, JDBC)
- Broker interoperability (RabbitMQ, HornetQ, etc)
- Virtual Topics
- Mirrored Queues
- JMX Monitoring

When to use Messaging?

- Asynchronous communication/integration
- Durability
- Loose coupling
- Heterogenous integration
- Real-time data



FuseSource
integration everywhere

Configuration

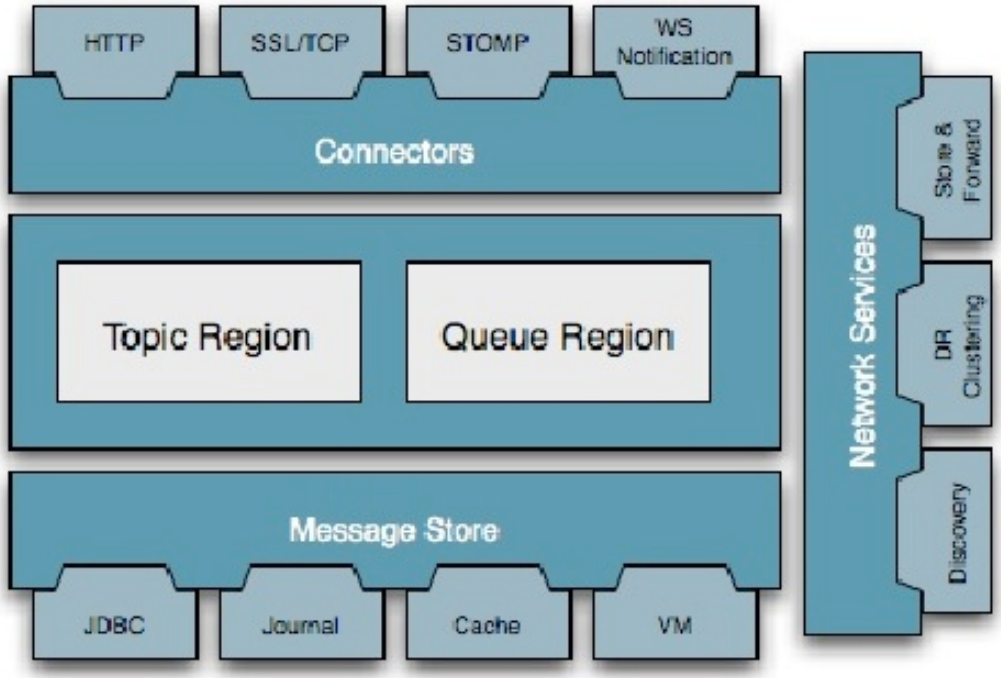
ActiveMQ is Highly Configurable!!

FuseSource
integration everywhere

Performance/Benchmarking Tools

So What Are We Covering?

- Tools
- How to approach performance tuning
- Areas to tweak!



FuseSource
integration everywhere

What are Your Objectives!?

- Please, please, please.. Know what you're trying to accomplish!
- Know your use cases!
- Know your hardware/OS
- Understand all of the broker config changes you make!
- VALIDATE YOUR CHANGES!

FuseSource
integration everywhere

Broker Benchmarking Tools

- ActiveMQ Performance Module
- jms-benchmark
- JMSTester
- Apache Jmeter
- Grinder
- …?

FuseSource
integration everywhere

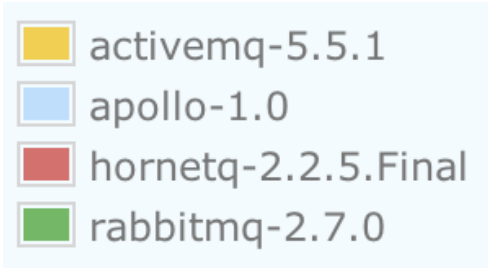
ActiveMQ Performance Module

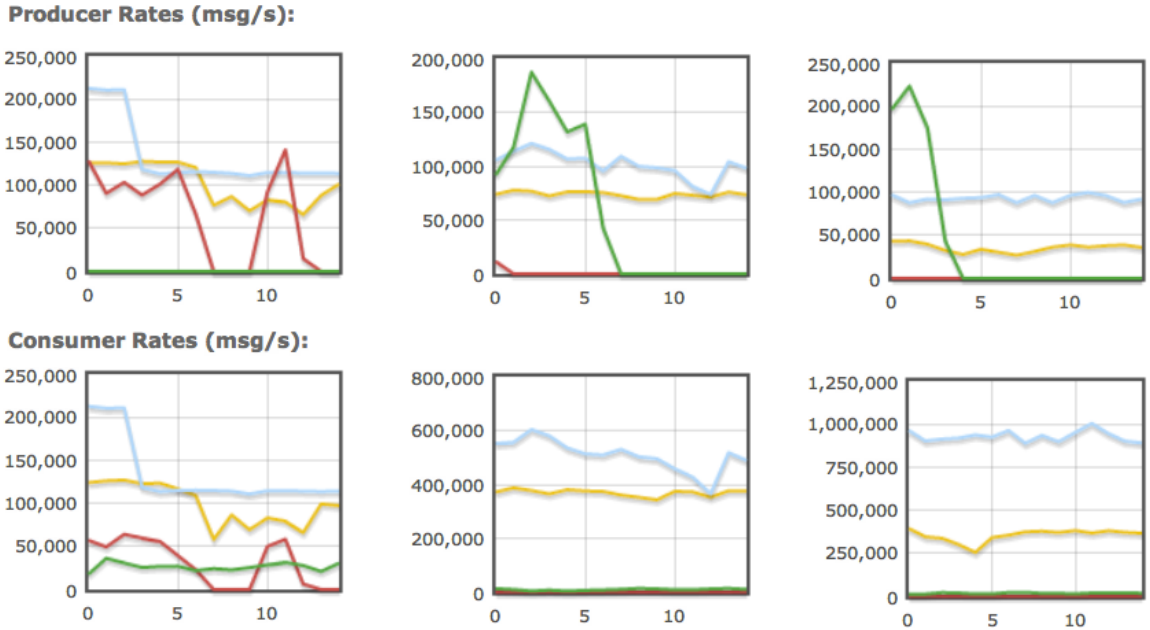
- Part of the ActiveMQ Build
  http://activemq.apache.org/activemq-performance-module-users-manual.html
- Maven plugin
- Quick and easy to get started
- Flexible
- Records throughput, basic statistics, and CPU metrics

FuseSource
integration everywhere

jms-benchmark

- Opensource at github, from Hiram Chirino
  http://github.com/chirino/jms-benchmark
- Comprehensive benchmarks
- Configurable
- All-on-one
- Multiple brokers
- Pretty graphs!





FuseSource
integration everywhere

JmsTester

- From FuseSource
  http://jmstester.fusesource.org
- My branch:
  https://github.com/christian-posta/jms-tester
- Allows more complicated load testing
- Distributed
- Records messaging throughput and CPU, Memory, Network IO, Disk IO, et al.
- Thorough documentation
- My blog post:
  http://www.christianposta.com/blog/?p=268

JMeter

- Swiss army knife of load testing
- From Apache
  http://jmeter.apache.org
- JMS, HTTP, TCP, SOAP, JDBC, and many others
- Many aggregators and reporting features
- Mature product, well known

OS Tools

- Lest we forget… rely heavily on the tools your Operating System exposes!
- Linux
  - top
  - iostat
  - netstat
  - vmstat
- Windows
  - Built in process, resource monitors

FuseSource
integration everywhere

ActiveMQ Performance Module

- Maven plugin

```
<plugin>
    <groupId>org.apache.activemq.tooling</groupId>
    <artifactId>activemq-perf-maven-plugin</artifactId>
    <version>${activemq-version}</version>
</plugin>
```

- Can use existing project: https://svn.apache.org/repos/asf/activemq/sandbox/activemq-perftest
- Spin up producers, consumers
- Control number of threads, test durations, etc
- Stats and Reporting
  - Throughput, min/max, broken down per client

Using Performance Module

- Example:

```
user@computer> mvn activemq-perf:producer –Dfactory.brokerURL=tcp://localhost:61616
```

- Out of the box examples:

```
user@computer> mvn activemq-perf:producer \
 –DsysTest.propsConfigFile=AMQ-prod-1-1-queue-persistent.properties
```

- Other out of the box profiles:
  - Producer
    - AMQ-Prod-1-1-queue-nonpersistent.properties
    - AMQ-Prod-10-1-queue-nonpersistent.properties
    - AMQ-Prod-10-10-topic-persistent.properties
  - Consumer
    - AMQ-Cons-1-1-queue.properties
    - AMQ-Cons-10-1-topic-durable.properties
    - AMQ-Cons-10-10-queue.properties

FuseSource
integration everywhere

What Broker to Use?

- Load up an external broker
  - Existing broker
  - No maven access
  - Different machine (good idea!)

- Load one from the perf module
  - Tests and broker config all in one
  - Simple maven command

```
user@computer> mvn activemq-perf:broker -Durl=tcp://localhost:61616
```

Defaults

- Overall
  - tp,cpu reports
  - 5 min
  - 30s warm up 30s cool down
  - 1s sample rate
- Producer
  - Non-persistent
  - Auto ack
  - 1K message size
- Consumer
  - Non durable subscription
  - Auto ack
  - No tx

Quick Demo!

Results explanation

```
##########################################
####    SYSTEM THROUGHPUT SUMMARY    ####
##########################################
System Total Throughput: 2589232
System Total Clients: 2
System Average Throughput: 10788.466666666667
System Average Throughput Excluding Min/Max: 10699.841666666667
System Average Client Throughput: 5394.233333333334
System Average Client Throughput Excluding Min/Max: 5349.920833333334
Min Client Throughput Per Sample: clientName=JmsProducer1, value=3452
Max Client Throughput Per Sample: clientName=JmsProducer1, value=7215
Min Client Total Throughput: clientName=JmsProducer1, value=1294421
Max Client Total Throughput: clientName=JmsProducer0, value=1294811
Min Average Client Throughput: clientName=JmsProducer1, value=5393.420833333334
Max Average Client Throughput: clientName=JmsProducer0, value=5395.045833333334
Min Average Client Throughput Excluding Min/Max: clientName=JmsProducer1, value=5348.975
Max Average Client Throughput Excluding Min/Max: clientName=JmsProducer0, value=5350.866666666667
300699 [main] INFO org.apache.activemq.tool.reports.XmlFilePerfReportWriter - Created performance report: /Users/cposta/dev/sandbo
x/activemq-perftest-https/./JmsProducer_numClients2_numDests1_all.xml
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5:21.125s
[INFO] Finished at: Fri Apr 19 08:05:16 MST 2013
[INFO] Final Memory: 9M/102M
[INFO] ------------------------------------------------------------------------
cposta@FusePostaMac(activemq-perftest-https) $
```


FuseSource
integration everywhere

While Running…

- Use JConsole!
- A more powerful toolkit? YourKit?
- Rely on OS monitoring, top, iostat, etc

Tuning ActiveMQ

Approach

- Know your requirements
- Use real hardware, real scenarios, real settings
- Know what config changes do!
- Tune
- Run tests
- Capture results
- Repeat

- Change only one setting at a time!

FuseSource
integration everywhere

Where are the Bottlenecks?

- Overall system
- Network latencies
- Disk IO!
- Threading overheads
- JVM optimizations

Network

- Mbits/Gbits per second throughput
- Topology
  - Hops
  - Switches
  - Routes
  - Gateways
- Firewalls
- Internal/External clients

Disk

- Read/Write speeds
- Mechanical vs SSD
- Shared disks, SAN, NAS, NFS, etc
- ActiveMQ DiskBenchmark:

```
user@computer> java -classpath lib/activemq-kahadb-store-5.8.0.jar \
org.apache.activemq.store.kahadb.disk.util.DiskBenchmark \
/Users/cposta/temp/test.dat
```

```
Benchmarking: /Users/cposta/temp/test.dat
Writes:
  159232 writes of size 4096 written in 10.244 seconds.
  15543.928 writes/second.
  60.718468 megs/second.

Sync Writes:
  33213 writes of size 4096 written in 10.001 seconds.
  3320.968 writes/second.
  12.972531 megs/second.

Reads:
  5160332 reads of size 4096 read in 10.001 seconds.
  515981.6 writes/second.
  2015.5531 megs/second.
```

Tune What Needs Tuning!

The following aren't things you **must** or **need** do…

they are things to consider and can be worth tuning.

Your specific use cases will illuminate what needs tuning

FuseSource
integration everywhere

Transport Tuning

- TCP Tuning
  - Timeouts (Handshake, Teardowns)
  - Congestion avoidance, sliding windows
  - Send/Receive Buffers!!
  - Default 64K, but increase to Bandwidth Delay Product
    - Buffer = Bandwidth * RTT
      http://www.speedguide.net/bdp.php
    - See http://en.wikipedia.org/wiki/TCP_tuning
- Configure the socket:

```
<transportConnector name="openwire"
    uri="tcp://0.0.0.0:61616?maximumConnections=1000&
     wireformat.maxFrameSize=104857600&transport.socketBufferSize=131072" />
```

- transport.*
- socket.*

FuseSource
integration everywhere

Transport Tuning…

- Buffering between Socket and Protocol Codec
- IOBufferSize
  - Default 8k

```
<transportConnector name="openwire"
    uri="tcp://0.0.0.0:61616?maximumConnections=1000&
     wireformat.maxFrameSize=104857600&transport.ioBufferSize=65536" />
```

Wire Format Tuning

- OpenWire (default protocol)
  - wireFormat.tcpNoDelayEnabled – whether or not to use Nagle's batching algorithm (default: true ..that means no Nagle)
  - wireFormat.cacheSize – how many command object constants to cache (default: 1024)
  - wireFormat.cacheEnabled – turn cache on/off (default: true)
  - wireFormat.tightEncoding – reduce network load (default: true)
  - wireFormat.maxFrameSize – message sizes (default: Long.MAX_LONG)
  - wireFormat.maxInactivityDuration – monitors period of inactivity (default: 30000ms, set to 0 to disable))

```
<transportConnector name="openwire"
    uri="tcp://0.0.0.0:61616?wireFormat.tcpNoDelayEnabled=false" />
```

Tuning Clients

- Compression
  - ?jms.useCompression=true

```
<transportConnector name="openwire” uri="tcp://0.0.0.0:61616?jms.useCompression=true" />
```

  - jms.* configures underlying connection factory
  - Uses java.util.zip.Deflater
- Encoding
  - JMS Message types
    - Object
    - Bytes
    - Text
    - Stream
    - Map

Tuning Producers

- Producer Flow Control
  - Persistent vs non persistent messaging
  - Broker vs Client
  - Broker memory, client producer window
  - Default sync send for persistent, async send for non-persistent and transactions (except for commit/rollback operation)
  - Entire connection flow control, individual destination flow control, TCP flow control
  - Disable flow control

```
<policyEntry queue="queueWildcard" producerFlowControl="false" />
```

- jms.alwaysSyncSend
- jms.useAsyncSend
- Send fail if no space

Failover

- Use failover transport for automatic reconnects
- backup
- priorityBackup
- trackMessages (default false)
- backOffMultiplier
- maxReconnectAttempts
- randomize
- Can automatically get updates from broker when cluster situation changes

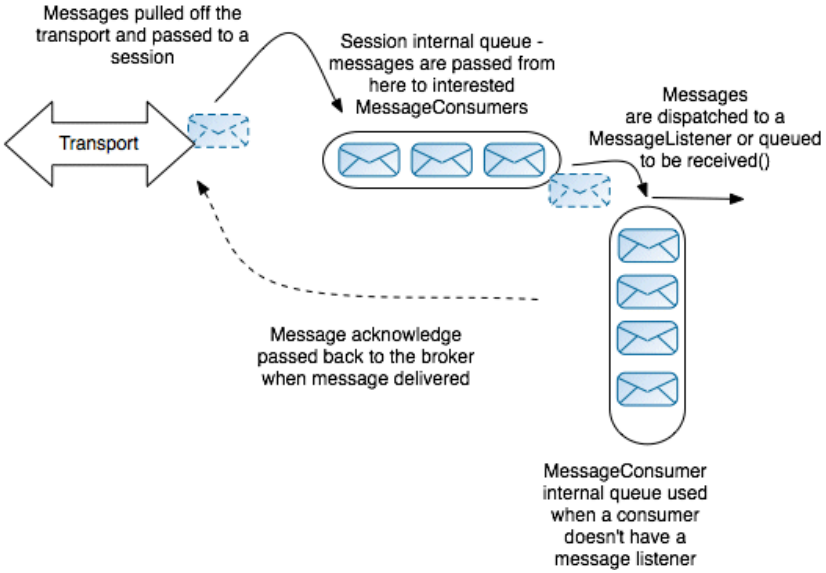
```
failover:(tcp://primary:61616,tcp://secondary:61616)?randomize=false
```

Other Producer Parameters

- jms.watchTopicAdvisories=false
- jms.copyMessageOnSend
- Batch using transactions

Tuning Consumers

- Prefetch
  - Queue – 1000
  - Queue Browser – 500
  - Topic – 32K
  - Durable Topic – 100
- Broker side memory
- Client side memory
- Consumer starvation



FuseSource
integration everywhere

Prefetch Settings

- Broker

```
<broker ... >
  <destinationPolicy>
    <policyMap>
      <policyEntries>
        <policyEntry queue="queue.>" queuePrefetch=”1”/>
        <policyEntry topic="topic.>" topicPrefetch=”1000”/>
      </policyEntries>
    </policyMap>
  </destinationPolicy>
</broker>
```

- Consumer

```
jms.prefetchPolicy.queuePrefetch=100
```

- Destination

```
new ActiveMQQueue("TEST.QUEUE?consumer.prefetchSize=10")
```

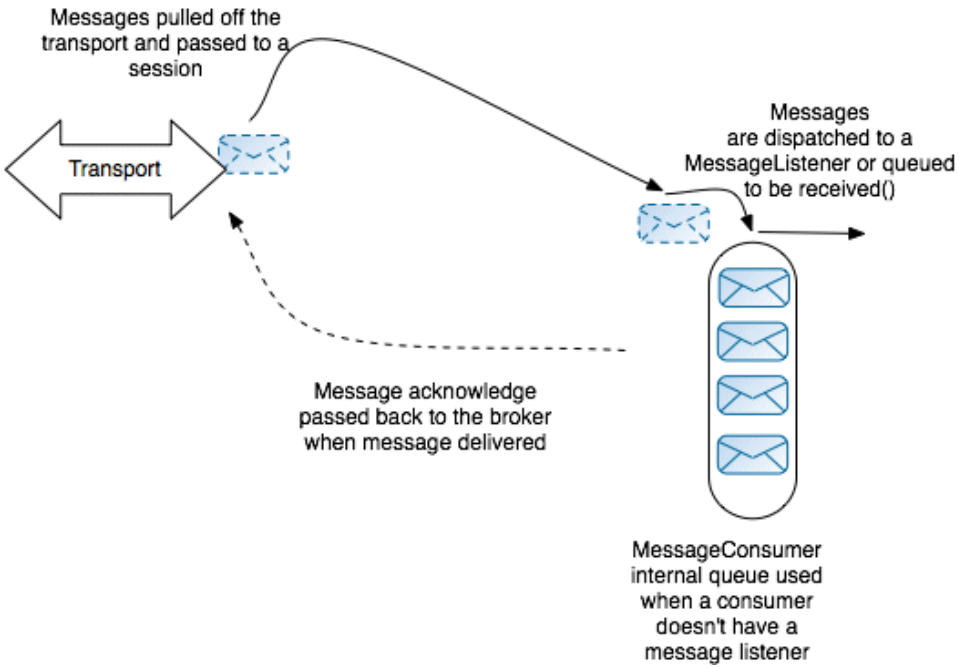
Consumer ACK Modes

- Session.AUTO_ACKNOWLEDGE
- Session.CLIENT_ACKNOWLEDGE
- Session.DUPS_OKAY_ACKNOWLEDGE
- Session.SESSION_TRANSACTED
- ActiveMQSession.INDIVIDUAL_ACKNOWLEDGE
- jms.optimizedAcknowledge with AUTO_ACK

- Note the differences with DUPS_OKAY_ACKNOWLEDGE when using topics or queues.

Other Consumer Parameters

- jms.alwaysSessionAsync reduce context switching
- MessageListener faster
- Spring DMLC: Cache consumers and connections!



FuseSource
integration everywhere

Tuning Broker Core

- Keep an eye on number of threads
  - UseDedicatedTaskRunner – use a dedicated dispatch thread for queues
  - optimizedDispatch – don't take the penalty of context switch, just let the transport thread do the dispatch
  - asyncDispatch – don't take the penalty of context switch, let the dispatching thread send to the consumer's transport
  - NIO connector nio://localhost:61616

Broker Resource Settings

- MemoryUsage settings for messages
- Caching for persistent vs non-persistent
- Policy settings, memory limits for destinations, cursor high water mark, etc
- Temp usage for offlining non-persistent messages
- Use the right combination of memory, producer flow control, and message cursors

Configuration of Resources

```
<policyEntry queue=“queueWildcard” memoryLimit=“10M” />


 <!– defaults -->
 <systemUsage>
  <systemUsage>
   <memoryUsage>
    <memoryUsage limit="64 mb" />
   </memoryUsage>
   <storeUsage>
    <storeUsage limit="100 gb" />
   </storeUsage>
   <tempUsage>
    <tempUsage limit="10 gb" />
   </tempUsage>
  </systemUsage>
 </systemUsage>
```


FuseSource
integration everywhere

KahaDB

- Current messaging database
- Fast, optimized for messaging
- Journal + Index + write ahead log
- Fast recovery
- Fewer file descriptors

Tuning KahaDB

- What levels of reliability must you have?
- Durability vs throughput
- fsync vs fflush
- Areas of tuning
  - Concurrent store and dispatch
  - Index paging, caching, page size, sync, checkpointing
  - Journal sync, file length, checkpointing

KahaDB Journal Settings

- enableJournalDiskSyncs (default: true)
- cleanupInterval (default: 30000ms)
- checkForCorruptJournalFiles (default: false)
- journalSize (default: 32MB)

```
<broker brokerName="broker" ... >
  <persistenceAdapter>
    <kahaDB directory="activemq-data" journalMaxFileLength="32mb"
                cleanupInterval="150000" checkForCorruptJournalFiles="true"/>
  </persistenceAdapter>
  ...
 </broker>
```

KahaDB Index Settings

- How often the index writes updates
  - checkpointInterval (default: 5000), indexWriteBatchSize (default: 1000)
  - Can amortize cost across small writes? Or batch them up for larger writes
- Size of the index in memory
  - indexCacheSize (default: 10000)
- Sync to disk
  - enableIndexDiskSyncs (default: true)

```
<broker brokerName="broker" ... >
  <persistenceAdapter>
    <kahaDB directory="activemq-data" journalMaxFileLength="32mb”
                indexCacheSize=“40000” enableIndexDiskSyncs=“false”/>
  </persistenceAdapter>
  ...
</broker>
```


FuseSource
integration everywhere

Tradeoffs

- Point to point, publish subscribe
- Levels of durability
- Duplicate messages
- Security
- Throughput
- Priority

Summary

- ActiveMQ is a highly tunable, configurable piece of server software
- Know your use cases and requirements
- Use tools to not only load the broker, but monitor the bottlenecks
- Tune for producers/consumers, broker core, network, and disk IO
- Ask the computer!

Questions?

Useful Links

- Apache ActiveMQ
  http://activemq.apache.org/
- JMS Benchmarks
  http://github.com/chirino/jms-benchmark
- ActiveMQ Performance module tools:
  http://svn.apache.org/repos/asf/activemq/sandbox/activemq-perftest

- Lots of blogging about ActiveMQ ☺
  http://christianposta.com/blog

FuseSource
integration everywhere