EMI Messaging PT

Lionel Cons

Massimo Paladin

2nd EMI All-Hands Meeting – Lund, 30th May 2011

Work Plan

- Year 1
  - Collate messaging requirements from the other EMI Product Teams
  - Perform a paper survey of the various options for an EMI messaging service
  - Provide messaging introductory information including guidelines

- Year 2
  - Based on the requirements from year 1 as well as practical tests, recommend which messaging solution(s) should be used within EMI
  - Prototype the additional software that complements the chosen messaging solution(s)
  - Provide consultancy to help the other product teams with messaging integration
  - Improve the messaging documentation with best practices and examples

Year 1 - requirements

- Collate messaging requirements from the other EMI Product Teams
  - Requirements collected through questionnaire
  - Formalized in a twiki document
  - Validated by the interested PTs
  - Published in the twiki: http://goo.gl/a8h6c

Requirements Analysis (year 2)

- A monolithic broker cluster is not enough
- From 10000 feet, a broker simply forwards data
- The amount of data is given by:
  - The message rate (sent and received)
  - The message size
- The worst case scenario would be 10GB/s
- Even the (guessed) average is a lot of data
  - 100MB/s – 1GB/s range
  - Short-lived connections
  - Over the WAN
- Dedicated messaging services that scale to real needs is clearly the right way

Year 1 - paper survey

- Perform a paper survey of the various options for an EMI messaging service
  - A detailed 29 pages survey has been produced
  - It has been validated with the software providers
  - It has been accepted by the mailing list
  - Published in the twiki: http://goo.gl/Z6XYr

Features

- Many features taken in consideration
  - License, Open Source
  - Programming Language, Supported OSes
  - Security Features (encryption, authorization, authentication…)
  - Monitoring
  - Broker Features (virtual hosts, clustering…)
  - Consumers Features (durability, wildcards…)
  - Messages Features (persistence, DLQ…)
  - (Proxy certificates not supported by anyone)

Feature	ActiveMQ 5.x	ActiveMQ Apollo (0)	HornetQ	MRG-M (1)	OpenMQ	Qpid C++	Qpid Java	RabbitMQ	0MQ
License	Apache 2.0	Apache 2.0	Apache 2.0	on purchase	CDDL 1.1	Apache 2.0	Apache 2.0	Mozilla PL	LGPL
Programming Language	Java	Scala	Java	Java or C++	Java	C++	Java	Erlang	C/C++
Open Source	✓	✓	✓	✗	✓	✓	✓	✓	✓
Clustering	✓	⚠	✓	✓	✓	✓	✗	✓	✗
Failover	✓	⚠	✓	✓	✓	✓	✗	✓	✗
Federation	✓	⚠	✓	✓	✗	✓	⚠	⚠	✗
Virtual Hosts	✗	✓	✗	✗	✗	✗	✓	✓	✗
Durable Consumers	✓	✓	✓	✓	✓	✓	✓	✓	✗
Exclusive Consumers	✓	✓	✓	✓	✓	✓	✓	✓	✗
Last Value Queue	✓	⚠	✓	✓	✗	✓	⚠	⚠	✗
Selectors	✓	✓	✓	✗	✓	✗	✓	✗	✗
Wildcards	✓	✓	✓	✓	✓	✓	✓	✓	✗
Composite Destinations	✓	✓	✗	✓	✗	✓	✗	✓	✗
Dead Letter Address	✓	⚠	✓	✓	✓	✓	✗	✓	✗
Expiration	✓	⚠	✓	✓	✓	✓	✓	✓	✗
Message Compression	✓	⚠	✗	✗	✓	✗	✗	✗	✗
Persistence	✓	✓	✓	✓	✓	✓	✓	✓	✗
Transaction	✓	✓	✓	✓	✓	✓	✓	✓	✗
Access Control List	✓	✓	✓	✓	✓	✓	✓	✓	✗
JAAS Authentication	✓	✓	✓	✗	✓	✗	✗	⚠	✗
SASL Authentication	✓	✓	✗	✓	✗	✓	✓	✓	✗
SSL	✓	✓	✓	✓	✓	✓	✓	✓	✗
JMS compliance	✓	⚠	✓	✓	✓	✓	✓	⚠	✗
JMX console	✓	⚠	✓	✓	✓	✓	✓	⚠	✗
Monitoring	✓	✓	✓	✓	✓	✓	✓	✓	✓
Queue Browser	✓	✓	✓	✓	✓	✓	✗	✗	✗
Web Sockets	✓	⚠	✓	✗	✗	✗	✗	✓	✗

Main Protocols

- STOMP
  - Lightweight protocol
  - Very interoperable, supported by many solutions
  - Version 1.1 well defined
- AMQP
  - Not mature
  - 1-0 still in validation process
  - 1-0 incompatible with previous versions
  - iMatix quit from AMQP
- REST (architectural style)
  - Good for some use cases
  - Lightweight publishing is an example

Other Protocols

- Won't be taken in consideration

- MQTT
  - extremely lightweight publish/subscribe protocol
- OpenWire
  - JMS compliant wire protocol native to ActiveMQ
- XMPP
  - Very generic protocol
  - Born for Instant Messaging

Messaging Solutions 1/3

- Apache ActiveMQ
  - Widely used product
  - Will be replaced soon with new generation Apollo
- Apache ActiveMQ Apollo
  - New generation broker
  - First release by end of 2011
- FUSE Message Broker
  - ActiveMQ but packaged by FUSE
- RabbitMQ
  - Widely used, active community
  - Pushed by Vmware (SpringSource)

Messaging Solutions 2/3

- HornetQ
  - Recent product, consistent roadmap
  - Tim Fox (head) hired by SpringSource
- Apache Qpid
  - 2 implementations: C++ and Java
  - Only AMQP restricted support
  - Does not work with other AMQP products
- MRG-M (RedHat product)
  - Should be similar to Qpid

Messaging Solutions 3/3

- IBM WebSphere Message Broker
  - No interoperability, no open source
- OpenAMQ
  - Dead project, iMatix dropped its support
- OpenMQ
  - Java specific project
- ØMQ
  - Brokerless, interesting integration with RabbitMQ

Year 1 - guidelines

- Provide messaging introductory information including guidelines
  - Presented in EMI 1st A-H Meeting + twiki pages
  - Validated via the mailing list
  - Published in the twiki: http://goo.gl/yykzQ

General Recommendations

- Minimize the amount of code that you write for messaging
  - Re-use existing code
  - Isolate technology independent code from the rest (brokers, destination...)
- Prepare for bad things
  - Messages can get lost
  - Messages can arrive out of order
  - Messages can be delivered multiple times
- Security
  - Do not trust data by default
  - Use cryptography if needed: encryption and signing

Messaging Protocol

- STOMP is the recommended protocol
  - Good enough for most use cases
  - Supported by many messaging solutions
  - Recommended client libraries:
    - Perl: Net::STOMP::Client
    - Python: stomp.py
- Other protocols might bind you to a specific messaging solution
- Although is not a protocol, JMS is probably the best Java client solution
  - JMS is only API
  - Most of the brokers are JMS compliant
  - Changing messaging solution => changing JMS provider

Message

- Header
  - List key/value pairs
  - Contains simple information, like message body metadata
  - Avoid conflict with other keys (i.e. use prefix)
  - Keys: ASCII letters + digits + dots + dashes
  - Values: ASCII printable characters
  - Avoid many keys and long values
- Body
  - Contains data to send
  - JSON recommended for simple applications
    - Well supported by programming languages
    - Widely used
  - Avoid using to a custom format
    - Messaging used for software components integration

Message Size

- Messages should be small
  - They are copied many times
  - 1KB to 10KB is the optimal range
  - 1MB should be seen as absolute maximum
- Large messages incompatible with high rates
- Too small messages are not efficient
- Applications can easily adjust message size
- Compression can be used to reduce the size
  - But don't forget about the CPU time

Message Rate

- Real performance is the one measured in a realistic environment
- Establishing a session can be expensive
  - Especially using X.509
  - Try to minimize the number of sessions
  - Long lived connections can be problematic too
- What matters is the total number of messages in and out
  - Topic with 10 subscribers => 11 messages for each incoming message
- 1k msg/s practical maximum on WAN with STOMP
- If messages are persistent the rate should be reduced
- 1KB messages * 1K msg/s is 10Mbit/s at network level

Year 2 - messaging solution choice

- Based on the requirements from year 1 as well as practical tests, recommend which messaging solution(s) should be used within EMI
  - In progress, more work is needed
  - By the end of the summer
- Difficult to recommend any product
- No clear winner
- We can recommend a technology
  - STOMP as protocol
  - Independent interoperable brokers

Selection Process

- More than just looking at documentation
- Monitoring of the project
  - Activity
  - Community
  - Future plans
- Benchmark + Scalability + Stress testing
- Requirements evaluation
- Cost of the eventual required extensions
- Monitoring and Accounting (from EGI)

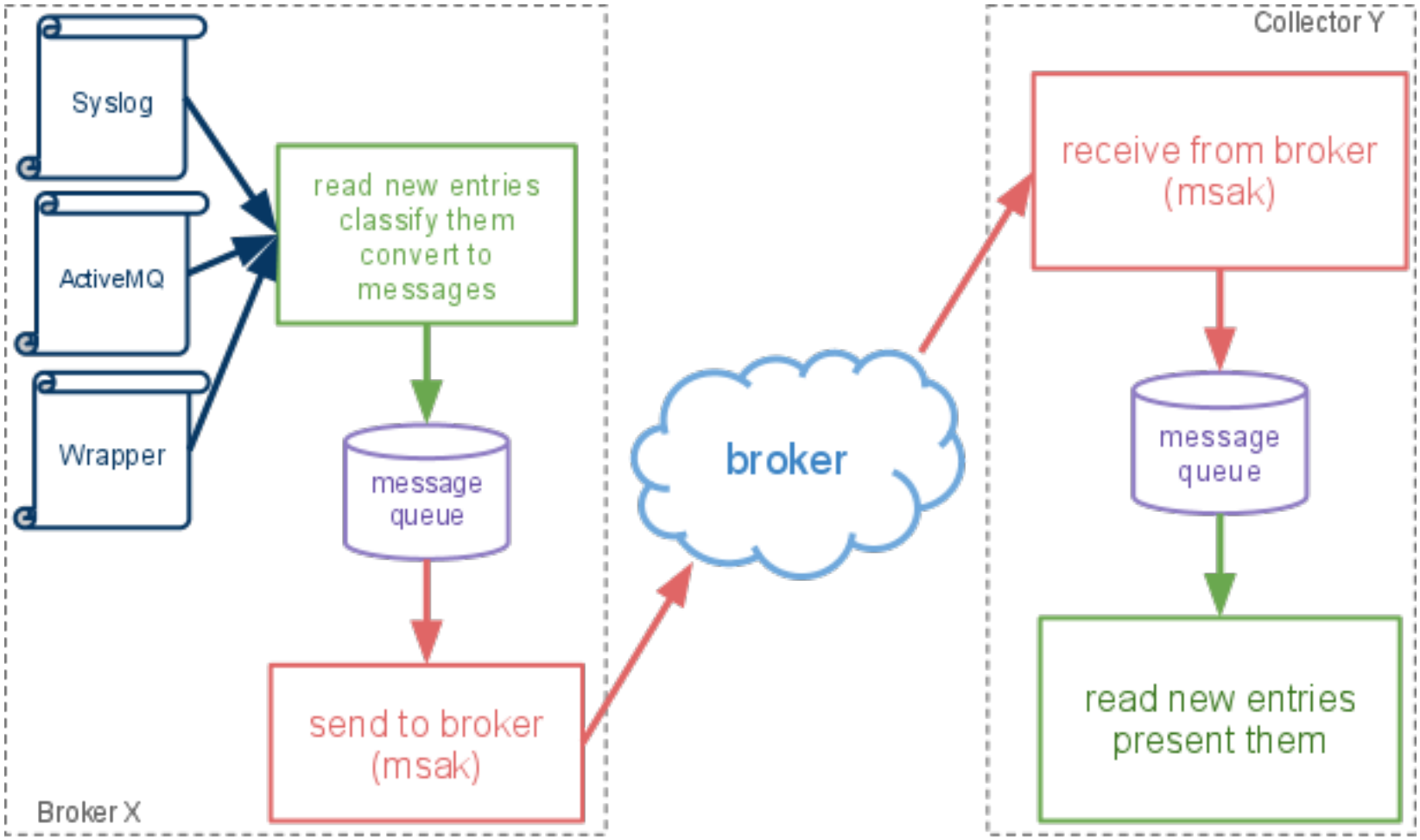
Cost of eventual extensions

- Required features might be missing
- Evaluate the cost of adding required features
  - Proxy certificates
  - Monitoring
  - Accounting
  - …
- Good software has a cost!

Year 2 - additional software

- Prototype the additional software that complements the chosen messaging solution(s)
  - In progress
- We have nice ideas and internal prototypes
- We want to provide good and robust
  - Libraries
  - Basic blocks to work with messaging

...example


Syslog
ActiveMQ
Wrapper
read new entries
classify them
convert to
messages
message
queue
send to broker
(msak)
Broker X
broker
Collector Y
receive from broker
(msak)
message
queue
read new entries
present them

Year 2 - consultancy

- Provide consultancy to help the other product teams with messaging integration
  - We are happy to help you
  - Use the mailing list

emi-jra1-messaging@eu-emi.eu

Year 2 - documentation

- Improve the messaging documentation with best practices and examples
  - We are constantly improving documentation
  - Feedback is very wellcome
  - The more feedback we get the better it will get!

Thank you!

EMI is partially funded by the European Commission under Grant Agreement RI-261611

Year 3 - Work Plan

- Develop and document the additional software part of the EMI messaging solution(s)
- Provide consultancy to help the other product teams with messaging integration
- Improve and finalise the messaging documentation