

# Next Generation Open Source Messaging with Apache Apollo

**Hiram Chirino**

Software Fellow

**Blog:** <http://hiramchirino.com/blog/>

**Twitter:** @hiramchirino

**GitHub:** <https://github.com/chirino>

**FuseSource**  
integration everywhere

# About me



## ■ Hiram Chirino

**Blog:** <http://hiramchirino.com/blog/>

**Twitter:** @hiramchirino

**GitHub:** <https://github.com/chirino>

- Software Fellow at FuseSource - <http://fusesource.com>
- Apache Member and ActiveMQ PMC Chair
- Apache Committer on: ActiveMQ, Camel, Karaf, ServiceMix, Geronimo, Felix, and Aries
- Lead of STOMP 1.1 Specification Co-Founder of many other OS projects:
  - HawtDispatch, Scalate, LevelDBJNI, Jansi, And many more!

# Outline

- What is Apache Apollo?
- What makes it different?
- What's the trajectory?



# What is Apache Apollo?

- OpenSource Messaging Server
- Subproject of ActiveMQ
- Like ActiveMQ, it Supports:
  - ⑩ Multiple protocols and client APIs.
  - ⑩ Multiple message storage options

# But your happy /w ActiveMQ?


## Yay! Stick with it!

- ActiveMQ will be supported for many more years to come!
- Will a long time before Apollo:
  - Supply all of ActiveMQ's features
  - Provides migration tools
- Apollo bits are being back ported

# Why use Apollo?

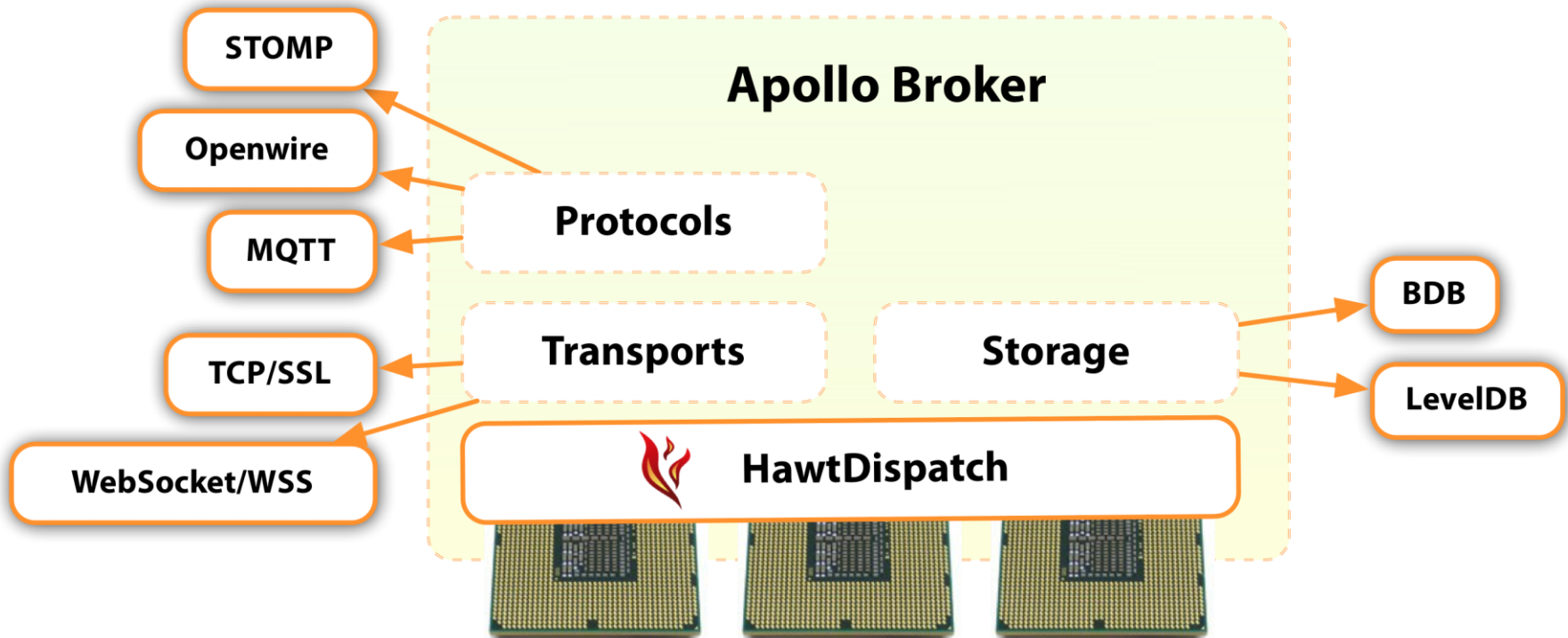
## Do you want:

- Lower CPU overhead
- Increased vertical scalability
- A reduced memory footprint
- Better Performance
- Runtime configuration reloading
- REST based management API



**What makes Apollo  
Different?**

# Apollo Architecture





# What is HawtDispatch?

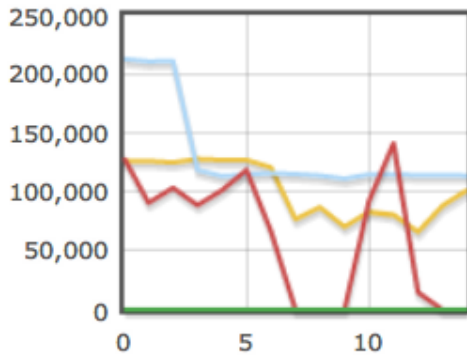


- Event Processing System
- Modeled after Grand Central Dispatch
- NIO Aware Fixed Size Thread Pool

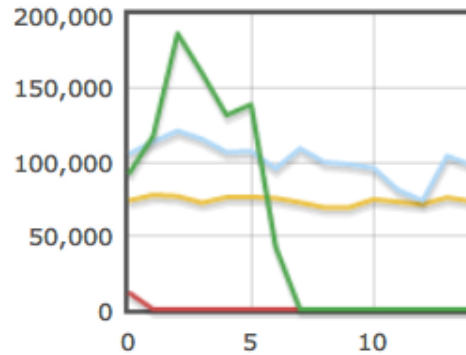
# Low Thread Contention...

1 -> 1 -> 1

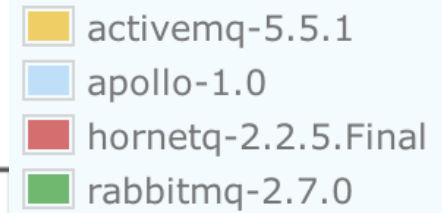
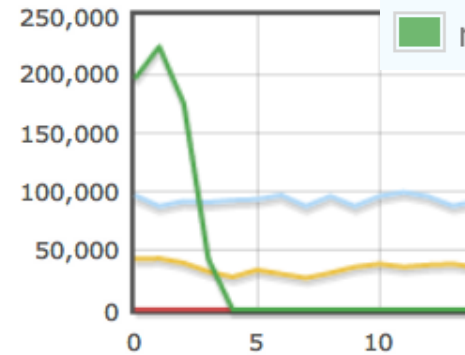
Producer Rates (msg/s):



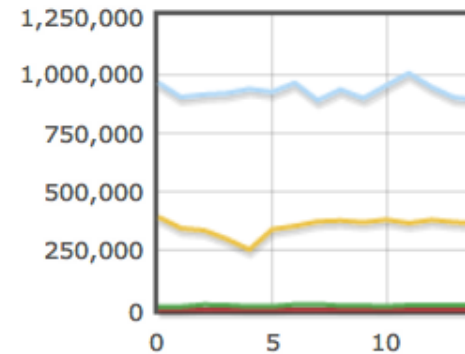
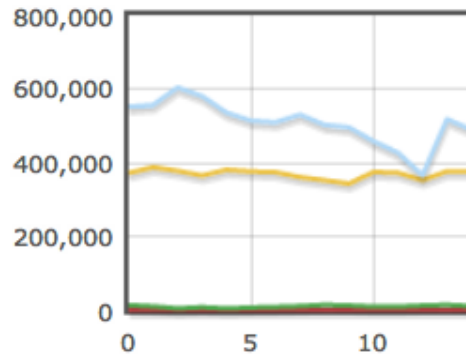
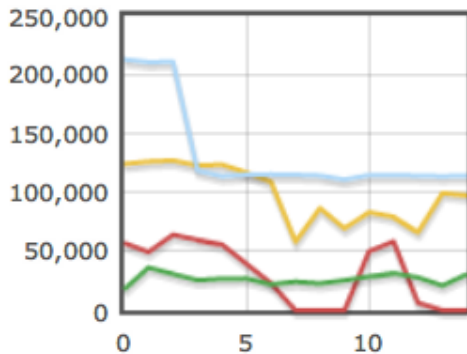
5 -> 1 -> 5



10 -> 1 -> 10



Consumer Rates (msg/s):



Source:

<http://hiramchirino.com/stomp-benchmark/ubuntu-2600k/index.html>

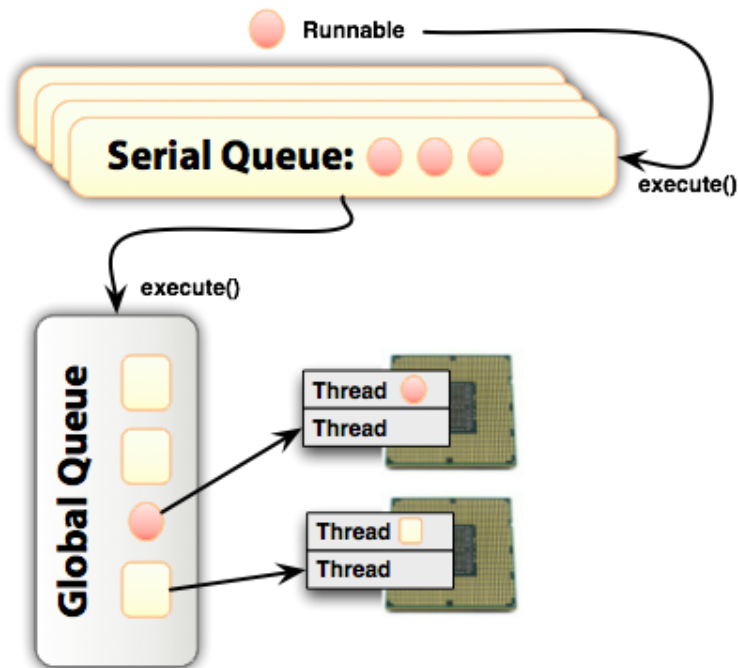
# HawtDispatch: Dispatch Queues

- Global Dispatch Queue
  - The fixed size Thread Pool

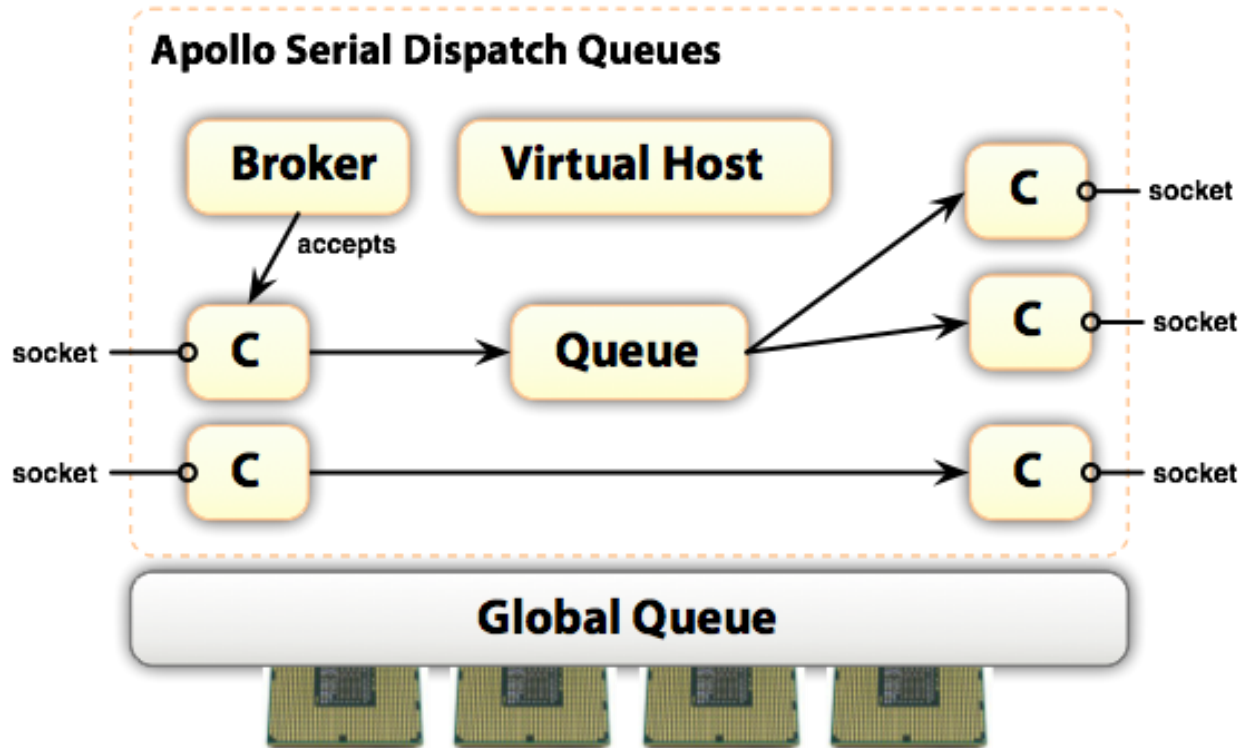
```
DispatchQueue queue = getGlobalQueue();
```

- Serial Dispatch Queue
  - Executes Runnable objects in order
  - CAS based Enqueues / Dequeues
  - Used like an Actor address

```
DispatchQueue queue = createQueue("My queue");
```



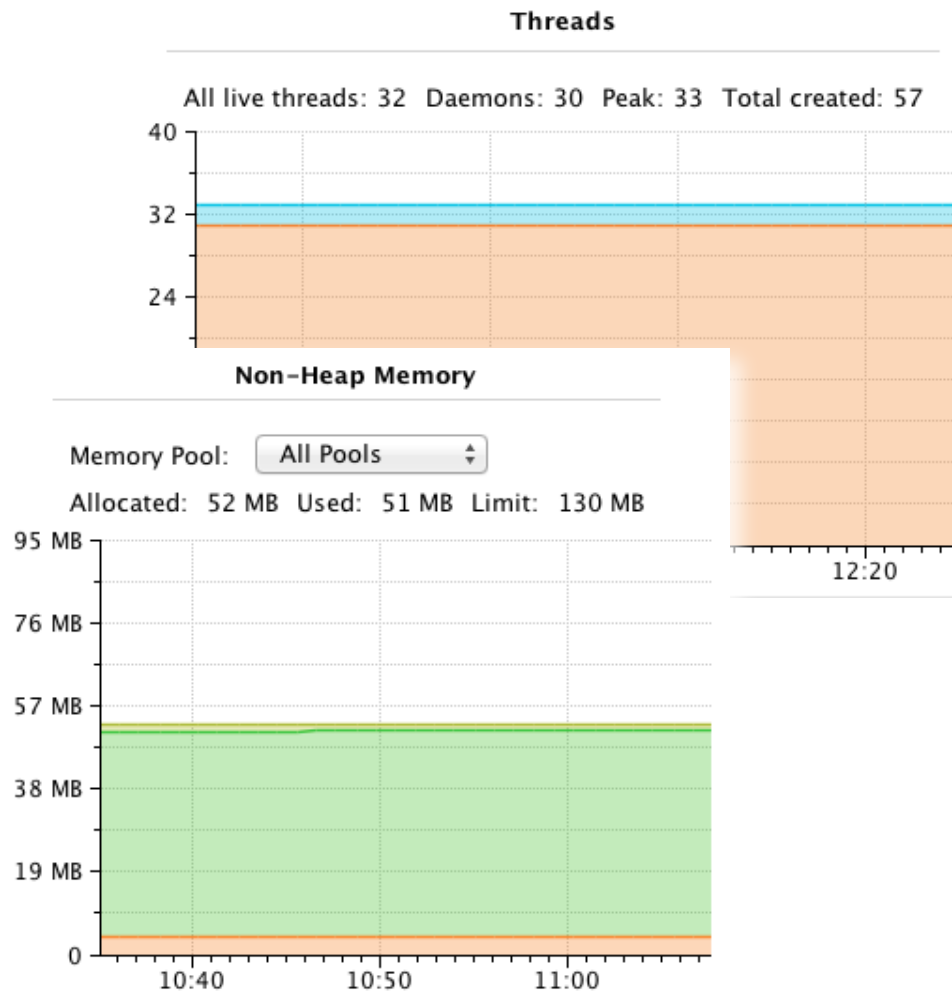
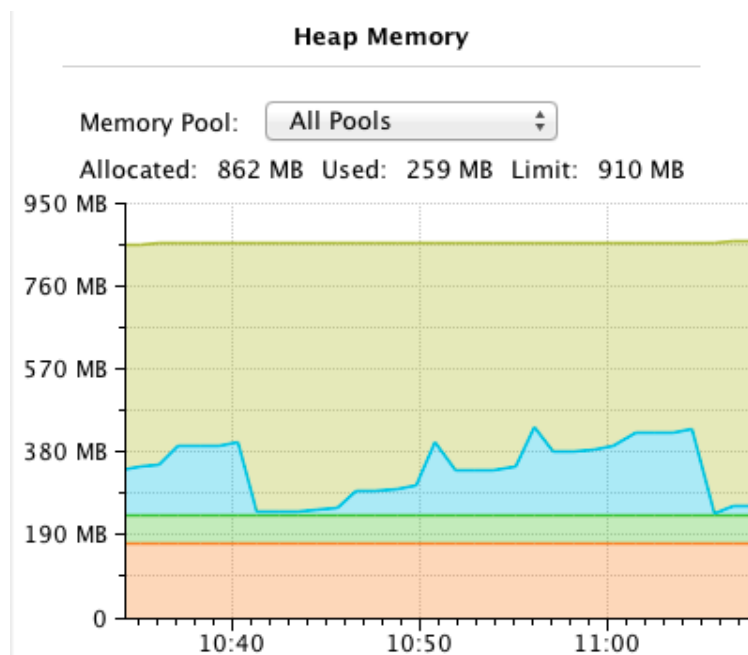
# Islands of Serialization in a Sea of Concurrency



**C** = Client Connection

# Low Memory Overhead...

- 1000 Producer Connections
- 1000 Topics
- 5000 Consumer Connections



# Why is Apollo using Scala?

- **Java API** example:

```
queue.execute(new Runnable(){  
    public void run() {  
        System.out.println("Hi!");  
    }  
});
```

- Same thing in the
- **Scala API:**

```
queue {  
    System.out.println("Hi!");  
}
```

- Terse closures FTW!

# Transports

- Are Plugins
- Comes with:
  - TCP
  - SSL
  - WebSockets
  - Secure WebSockets
  - UDP

# Message Protocols

- Are Plugins
- Protocols are Plugins
  - ⑩ STOMP 1.0/1.1
  - ⑩ MQTT v3.1
  - ⑩ Openwire
- All protocols can share a single Transport port.

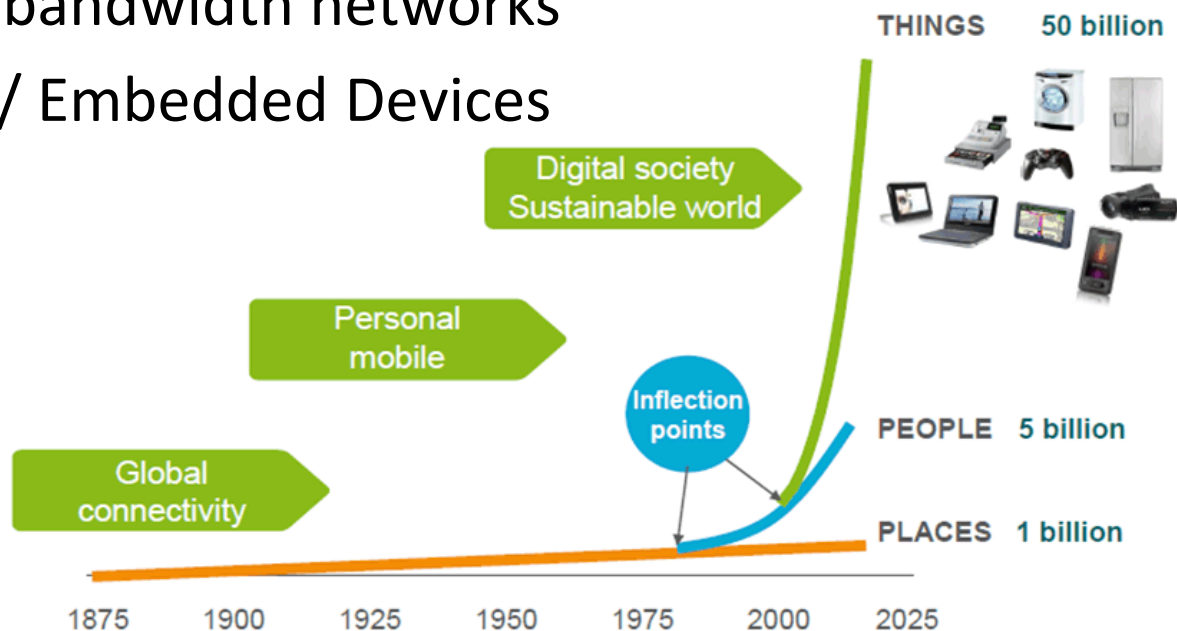


# Protocol: STOMP

- <http://stomp.github.com/>
- **Simple Text Orientated Messaging Protocol**
- Uses Text Headers like HTTP
- Many Clients APIs in Java, C#, C, Ruby, Python, JS, PHP, etc.
- Interoperates with ActiveMQ, RabbitMQ, HornetQ, ...

# Protocol: MQTT

- Get at <https://github.com/fusesource/fuse-extra/>
- Focused on:
  - Pub/Sub
  - Unreliable, low bandwidth networks
  - Small footprint / Embedded Devices
- Interoperates with WebsphereMQ, Mosquitto, ...



Source: Ericsson AB, "Infrastructure Innovation - Can the Challenge be met?," Sept 2010

# Protocol: Openwire

- Openwire is the native binary protocol implemented by ActiveMQ
- API options:
  - JMS 1.1 Client of ActiveMQ 5.x
  - NMS Client for C# Apps
  - CMS Client for C++ Apps
- Not Yet Supported
  - XA Transactions (distributed transactions)

# Message Stores

- Are Plugins
- Ships with 2 Options
  - LevelDB Store
  - BDB Store
- Used to store
  - persistent messages
  - non-persistent messages that needs to be swapped out of memory
- Non-persistent messages that get swapped out do not get dropped on restart
- Delayed Writes

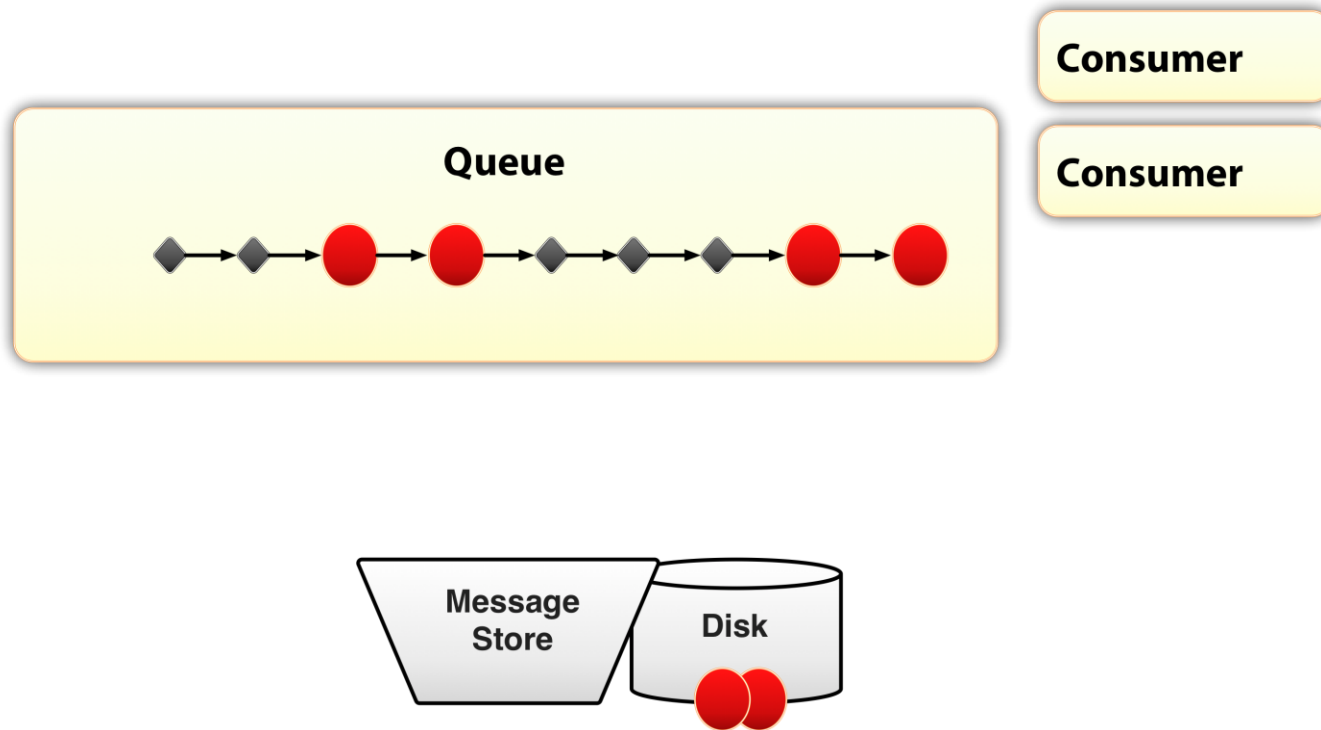
# Message Store: LevelDB Store

- A Journal + LevelDB based index
- The pure ASL 2.0 licensed option
- Uses a JNI implementation on Linux and OS X
  - Fastest Store available
- On all other platforms a pure Java implementation is used
  - Not as fast or robust as the JNI version
- LevelDB indexes are awesome for sequential r/w access patterns

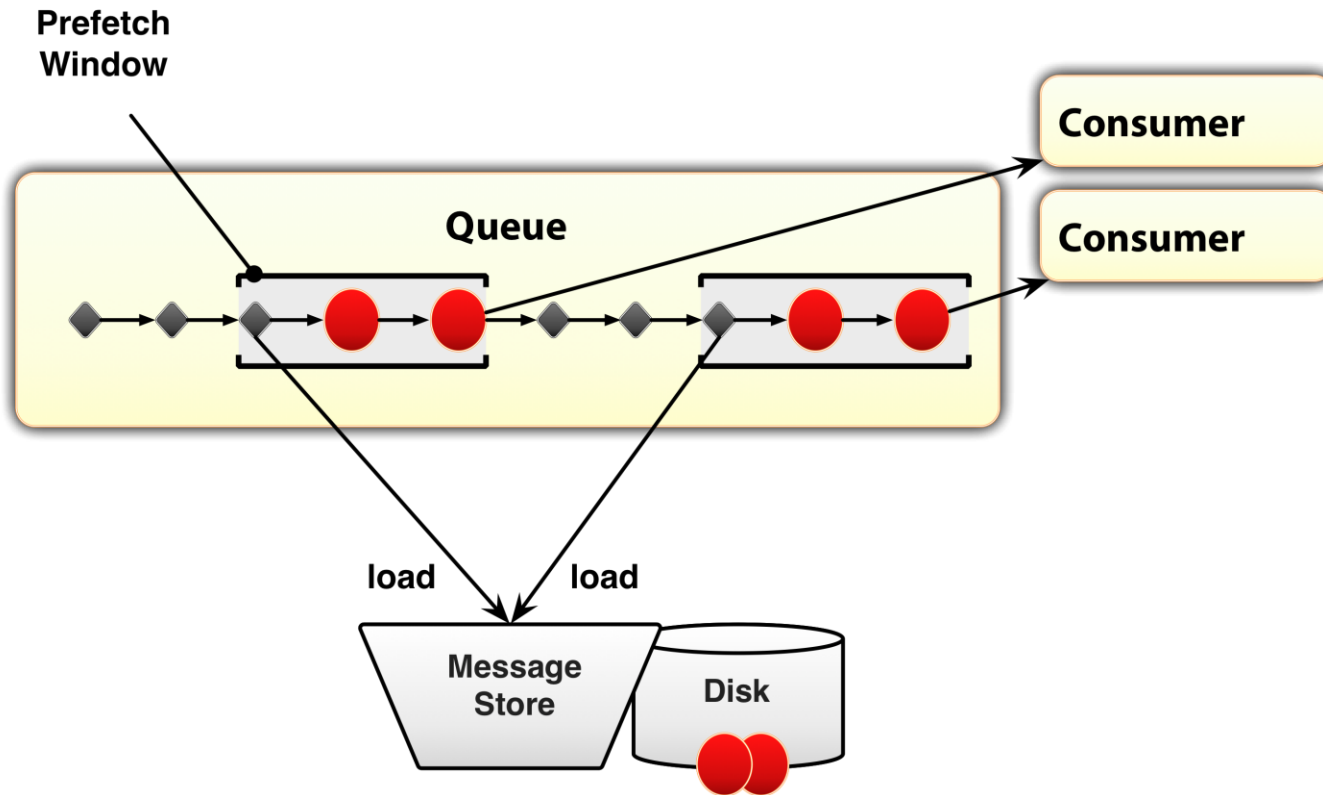
# Message Store: BDB Store

- Not ASL 2.0! You have to Agree to the BDB license & download from Oracle.
- Pure Java implementation
- Very robust
- The BDB library supports advanced features like replication (not yet exploited)

# Per Consumer Store Prefetch

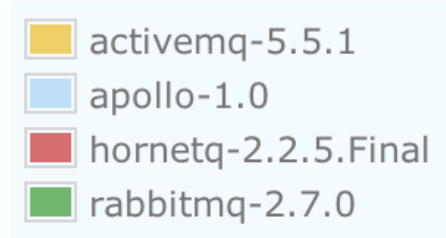
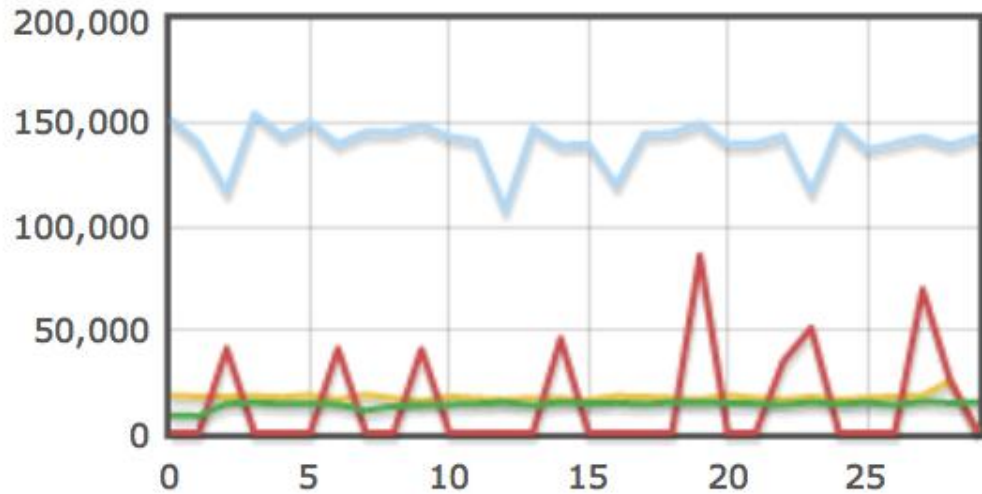


# Per Consumer Store Prefetch





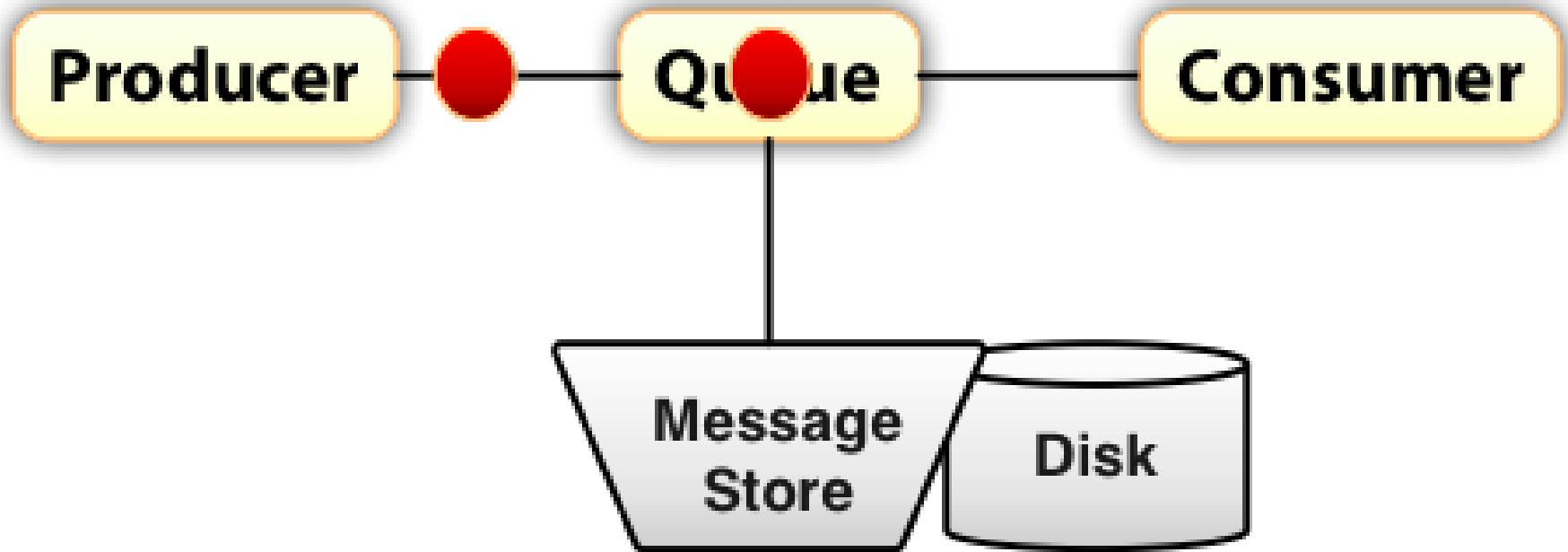
# Per Consumer Store Prefetch



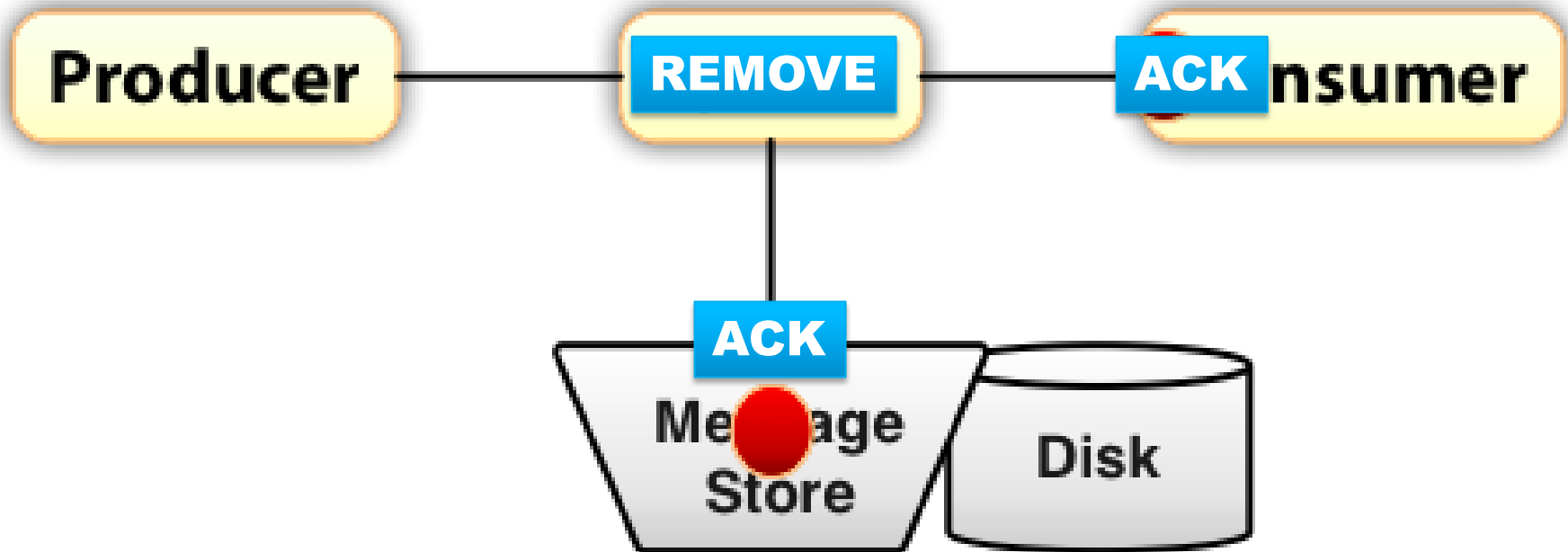
Source:

<http://hiramchirino.com/stomp-benchmark/ubuntu-2600k/index.html>

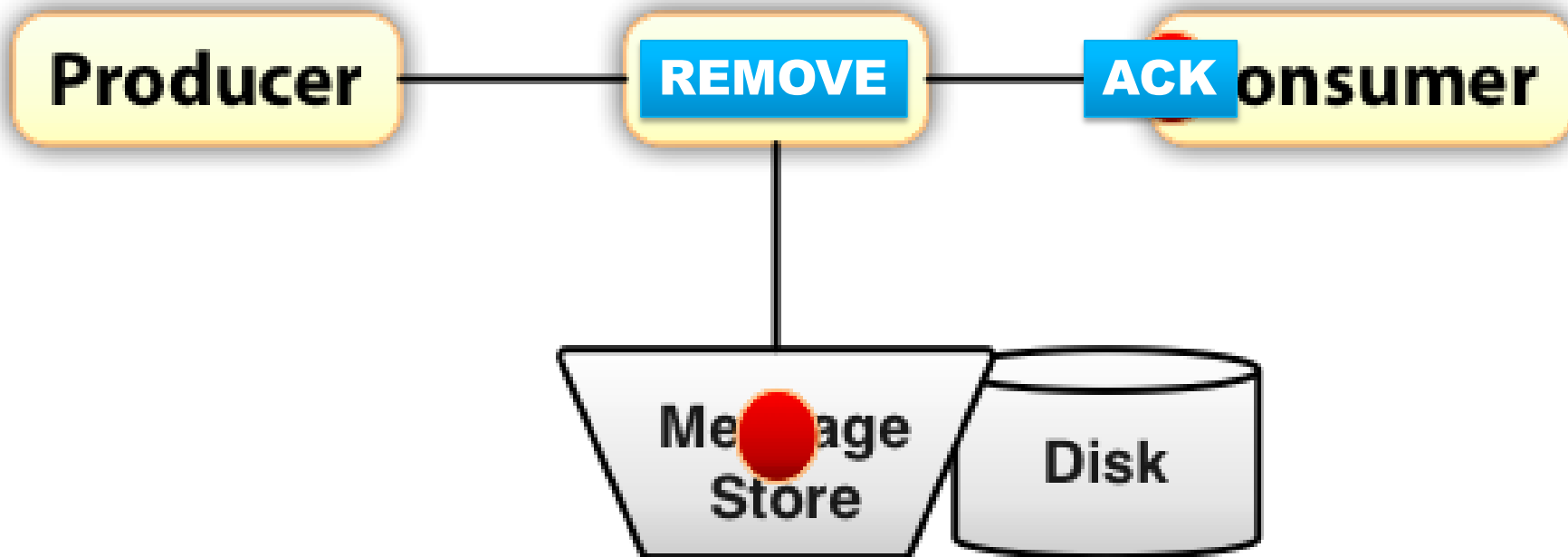
# Message Store: Store and Dispatch



# Message Store: Store with No Delay



# Message Store: Store with Delay



# Apollo's Trajectory

# Features! Features! Features!

## Road Map Features

- Networks of Brokers
- Priority Support
- Message Groups
- Message Scheduling
- XA Transactions
- JMX Management API

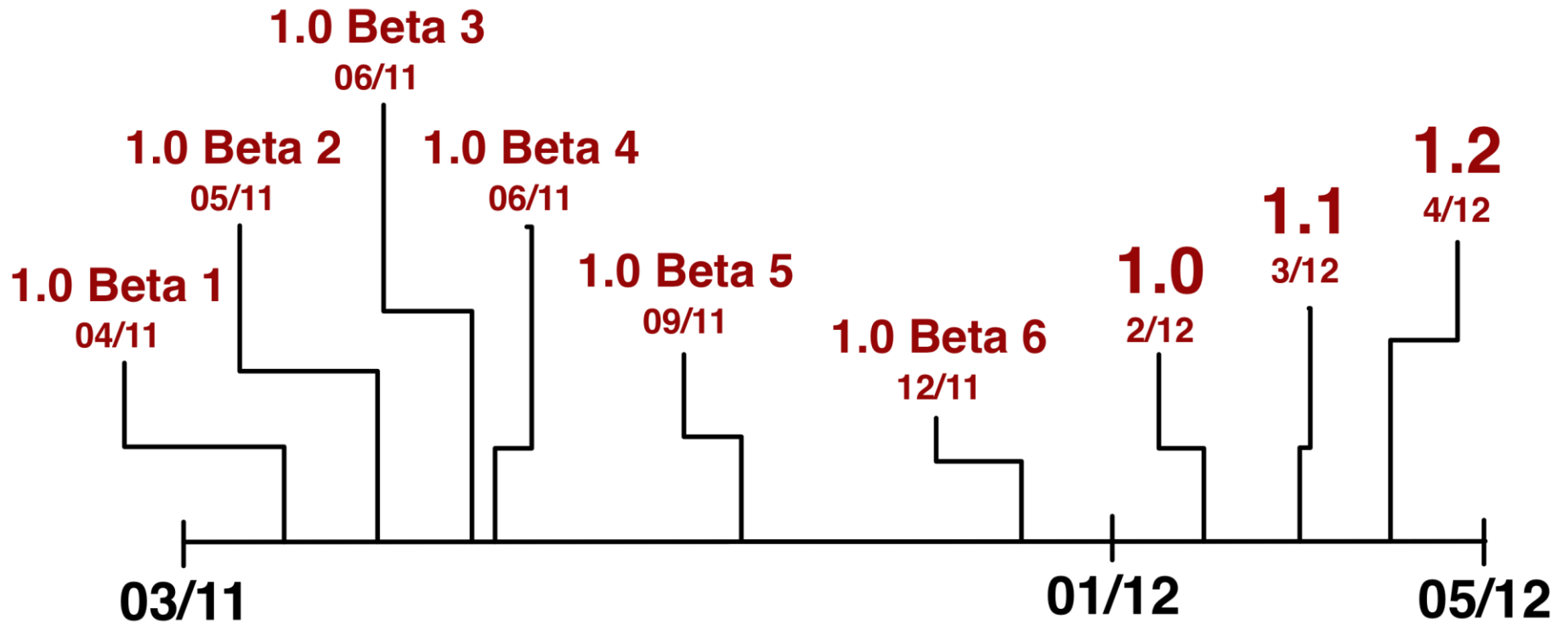
## Back Ported Apollo Features

- LevelDB Store
- MQTT Protocol
- STOMP 1.1 Support

## Pending Back Port

- Store Delays

# Release Velocity





# Questions?



# The Link Bonanza

- **Apache Apollo**  
<http://activemq.apache.org/apollo/>
- **STOMP Benchmarks**  
<http://hiramchirino.com/stomp-benchmark/>
- **MQTT Protocol Plugin for Apollo**  
<https://github.com/fusesource/fuse-extra>
- **HawtDispatch**  
<http://hawtdispatch.fusesource.org/>
- **StompJMS**  
<https://github.com/fusesource/stompjms>