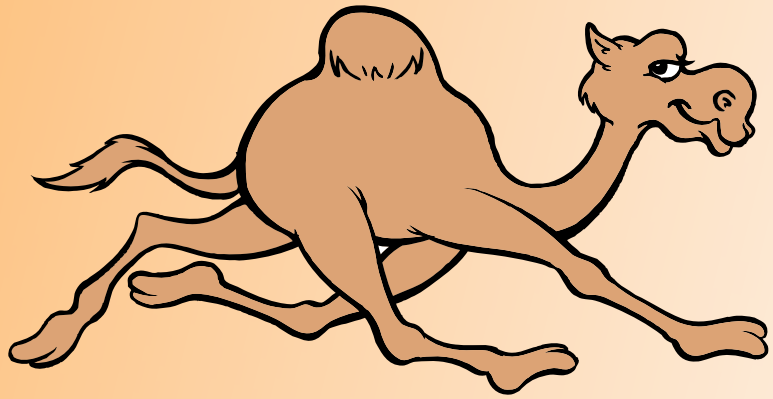
CamelOne 2013

June 10-11 2013

Boston, MA



Making Apache ActiveMQ Scale

Hiram Chirino Engineer Red Hat

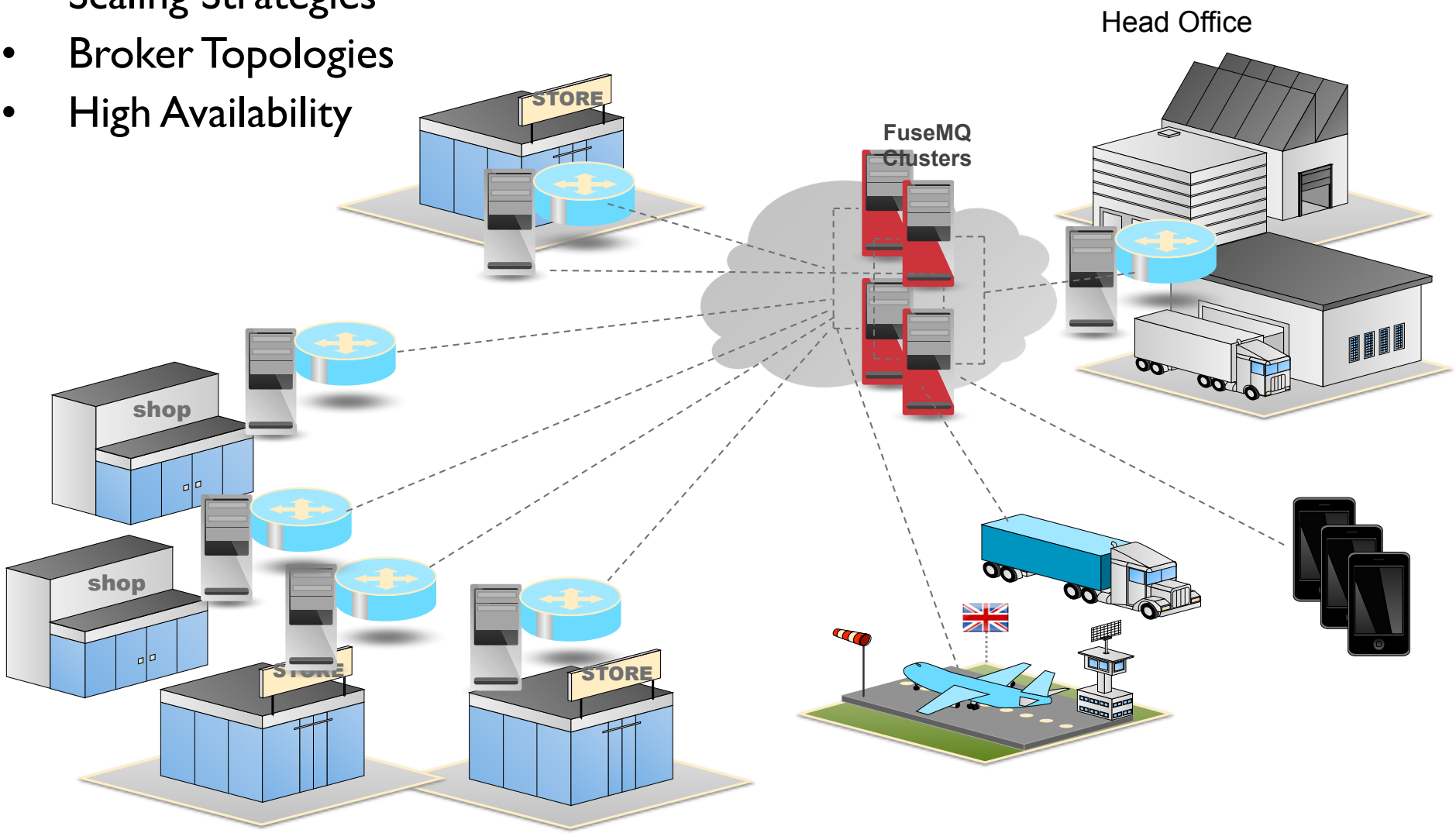
Presenter: Hiram Chirino

- Blog: http://hiramchirino.com/blog/
- Twitter: @hiramchirino
- GitHub: https://github.com/chirino

- Engineer at Red Hat
- Apache Member and ActiveMQ PMC Chair
- Apache Committer on: ActiveMQ, Camel, Karaf, ServiceMix, Felix, and Aries
- Lead of STOMP 1.1 SpecificationCo-Founder of many other OS projects

ActiveMQ – Enterprise Features

- Scaling Strategies
- Broker Topologies
- High Availability

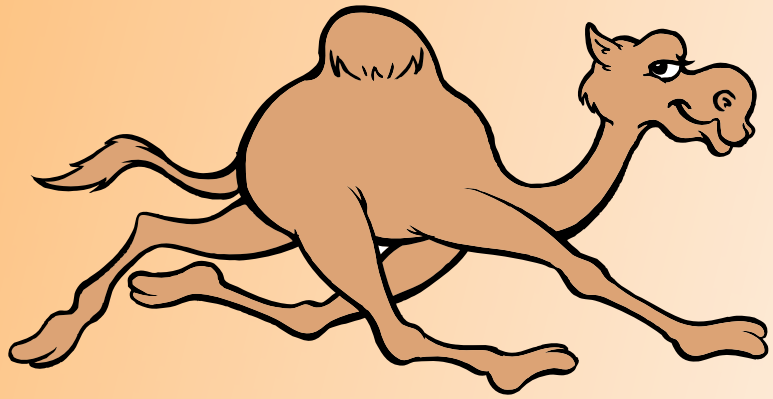


CamelOne

CamelOne 2013

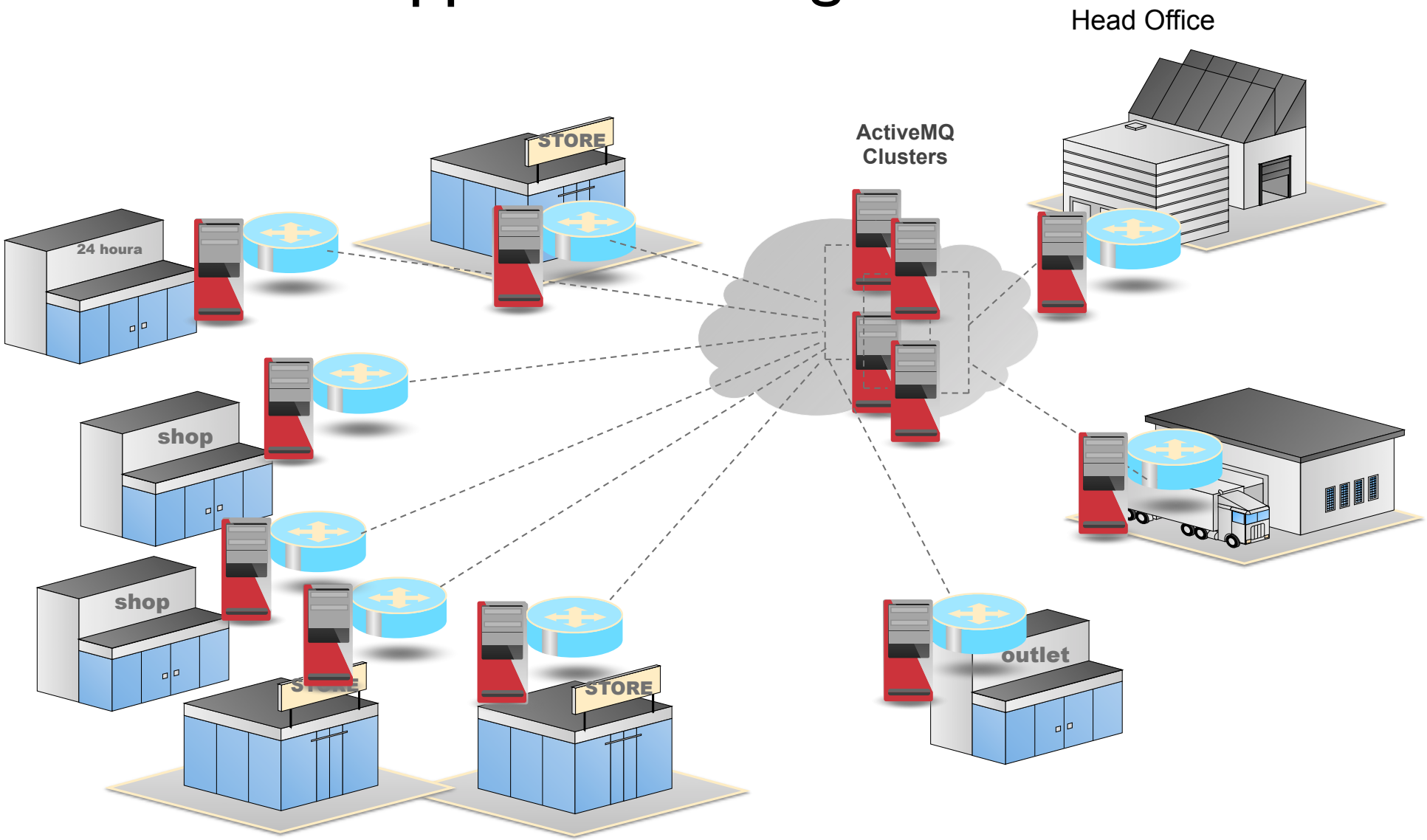
June 10-11 2013

Boston, MA



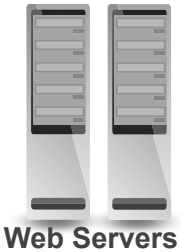
ActiveMQ Scaling Use Cases

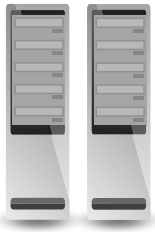
Example of Distributed Application Integration

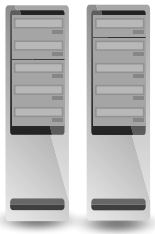


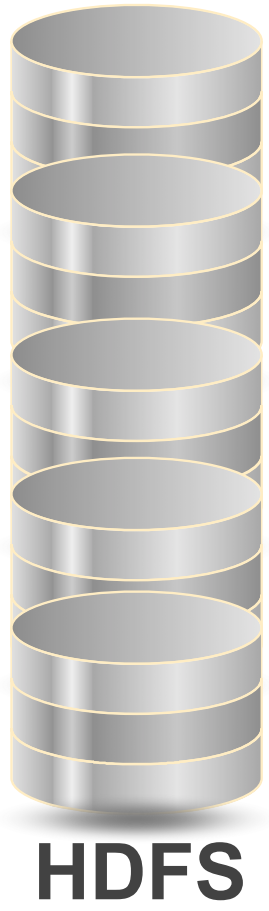
CamelOne

Ingestion for BigData Architecture:



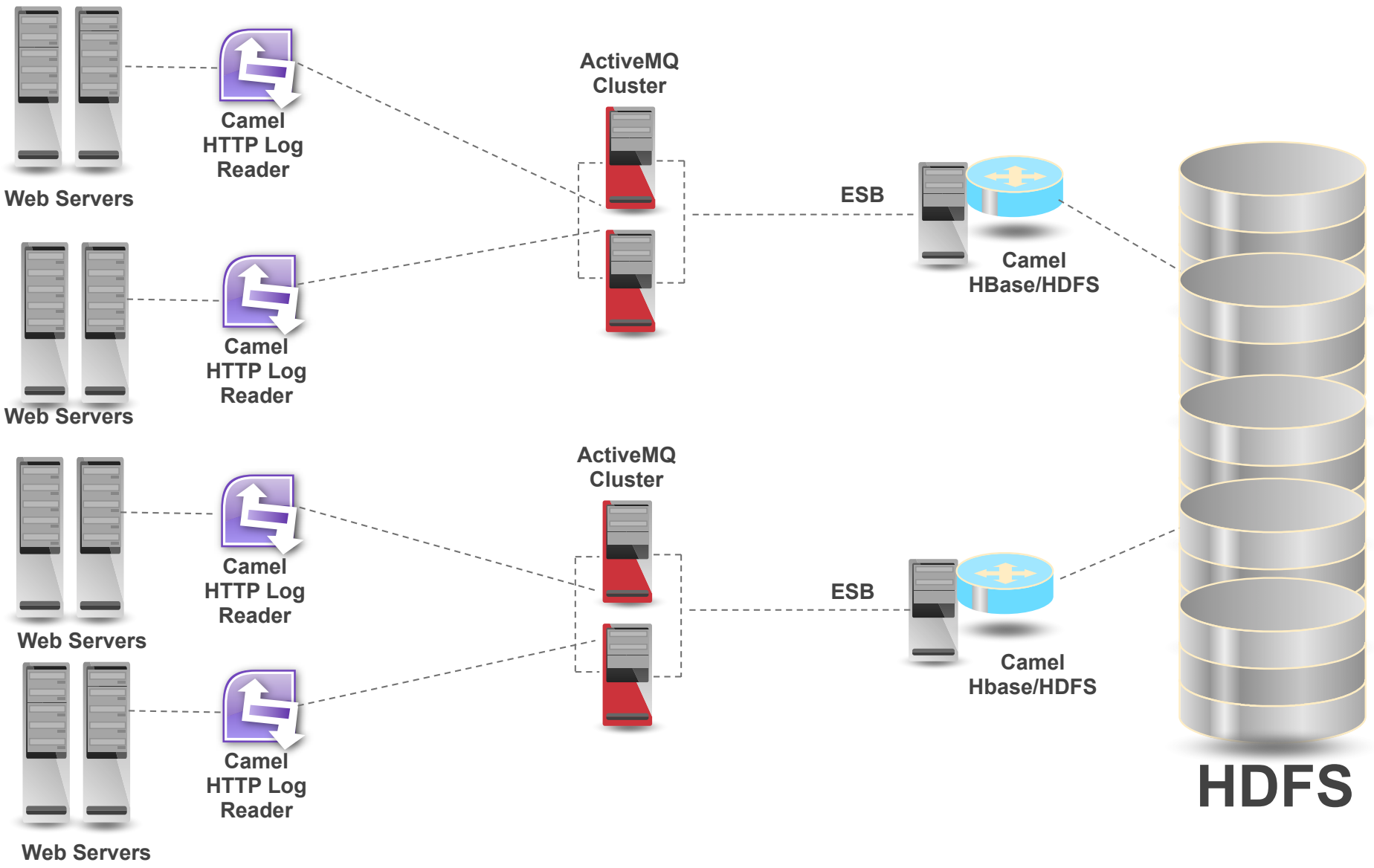






CamelOne

Ingestion for BigData Architecture:

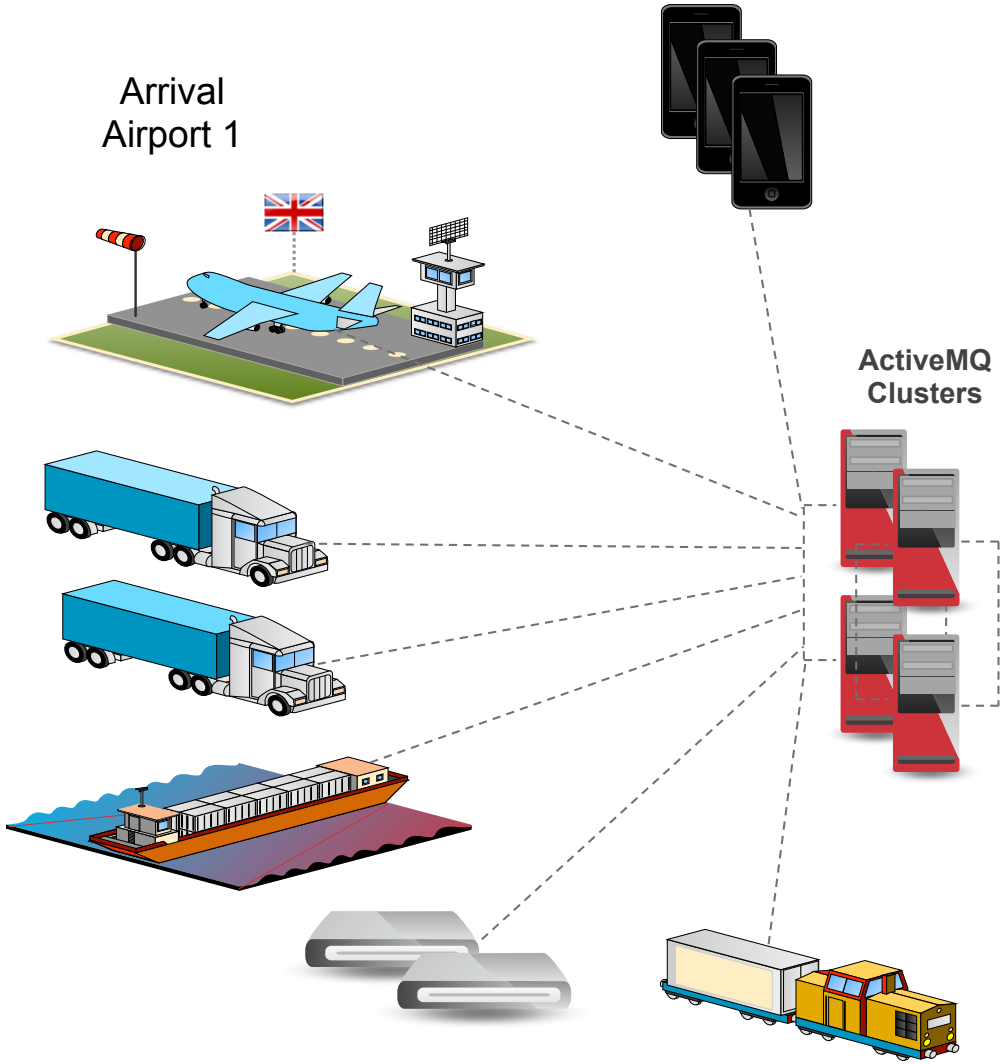


CamelOne

M2M Deployments

Connecting Things

- mobile devices
- meters
- industrial controls
- smart buildings
- asset tracking
- traffic control
- monitors
- sensors
- actuators

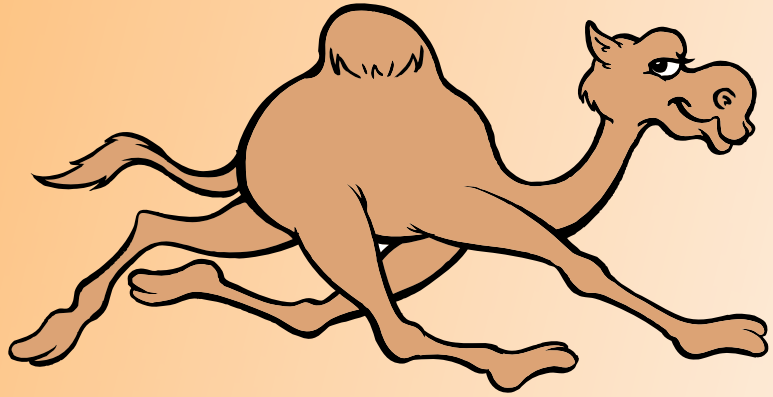


CamelOne

CamelOne 2013

June 10-11 2013

Boston, MA

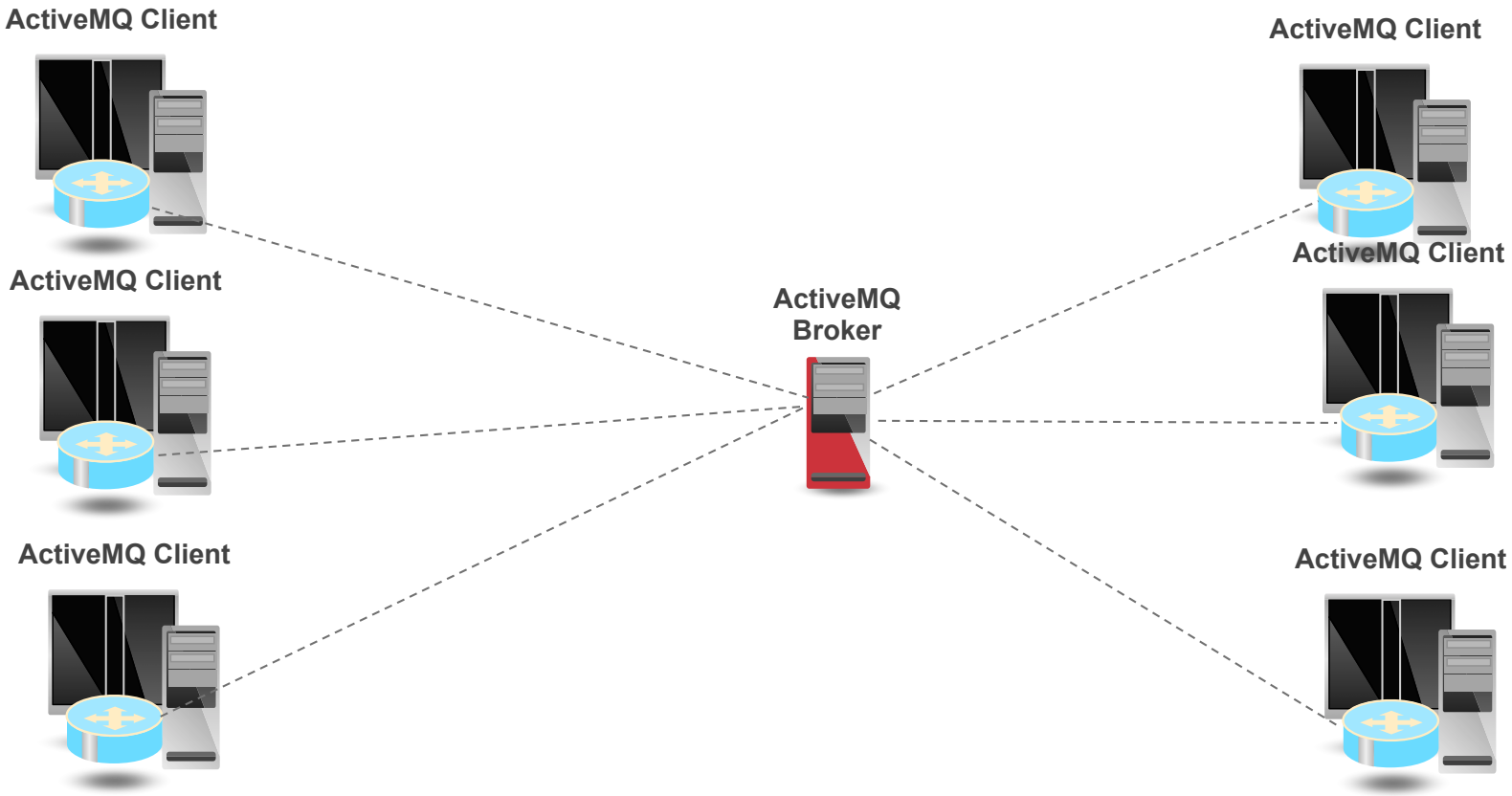


ActiveMQ Vertical Scaling

ActiveMQ – Vertical Scaling

CamelOne

ActiveMQ – Vertical Scaling



CamelOne

ActiveMQ – Vertical Scaling

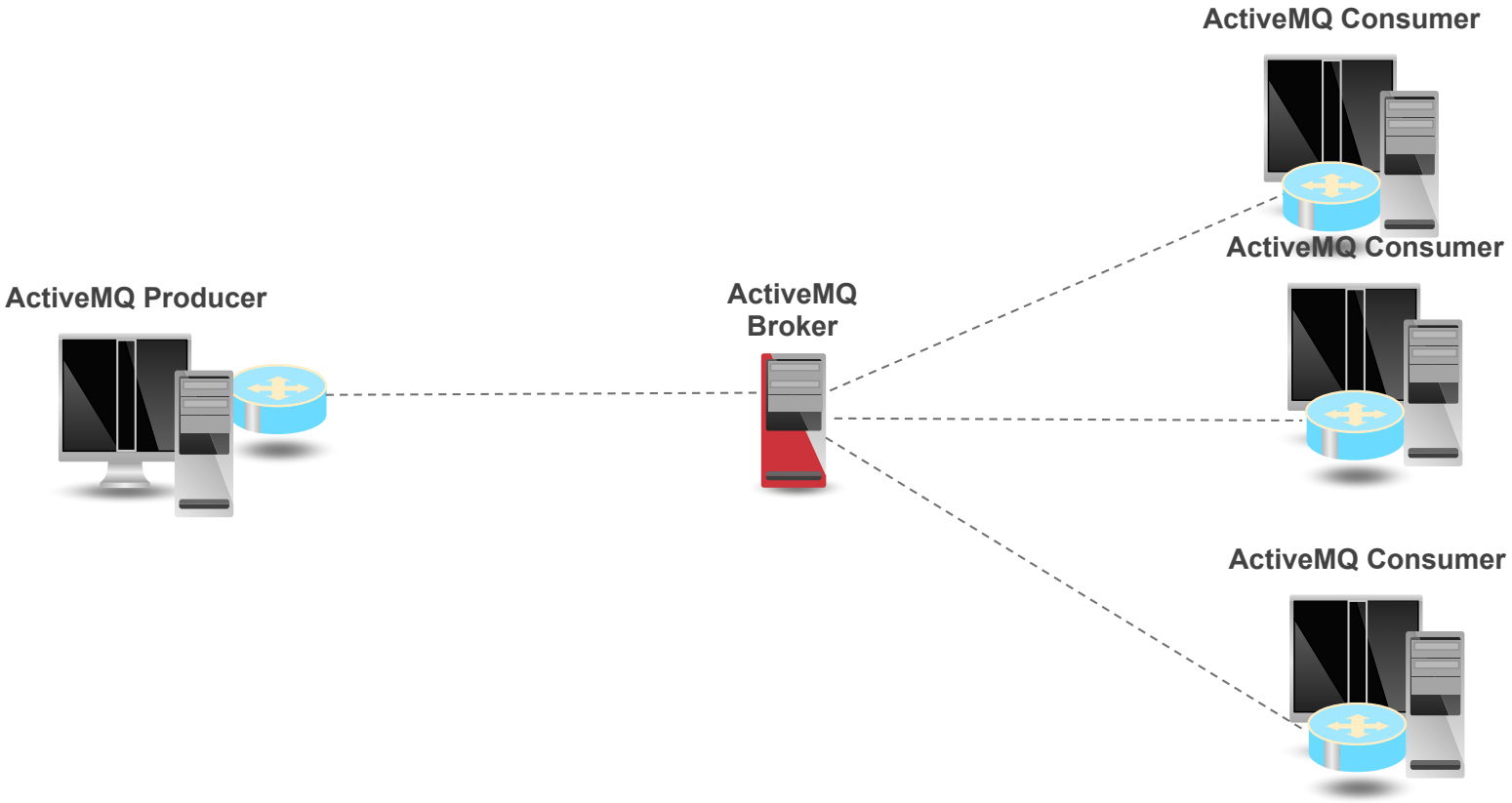
Reduce the number of Broker Threads

```
ACTIVEMQ_OPTS=
"-Xmx1024M -Dorg.apache.activemq.UseDedicatedTaskRunner=false"
```

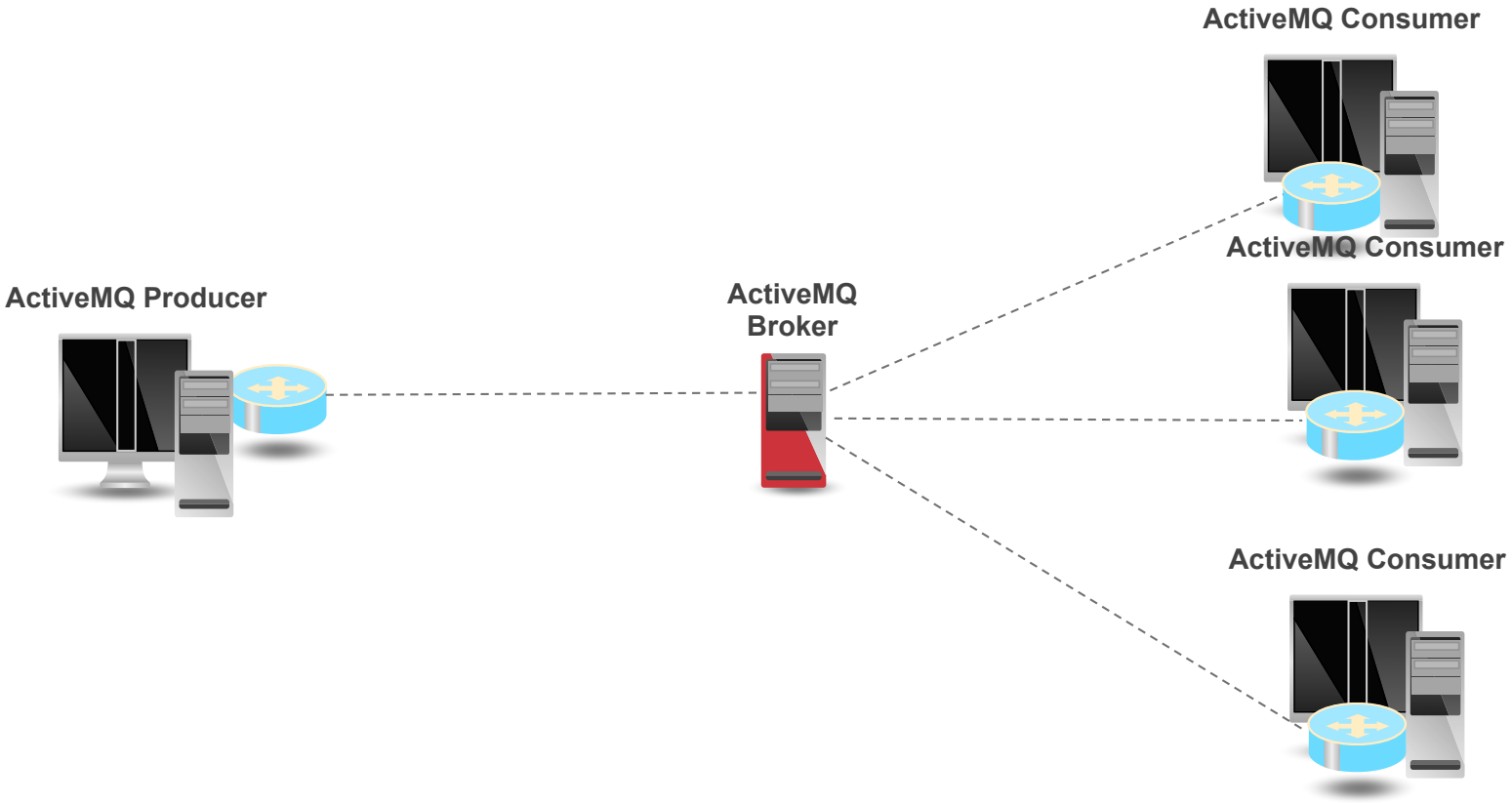
Reduce thread usage by Destinations

```
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry queue=">" optimizeDispatch=”true” />
    </policyEntries>
  </policyMap>
</destinationPolicy>
```

ActiveMQ – Synchronous Sends



ActiveMQ – Synchronous Sends



ActiveMQ – Synchronous Sends

Synchronous send: JMS Client

Set `alwaysSyncSend` on the ActiveMQConnectionFactory

You can set:
`sendTimeout` on the ActiveMQMessageProducer

In ActiveMQ 5.6 – you can get a callback – e.g.:

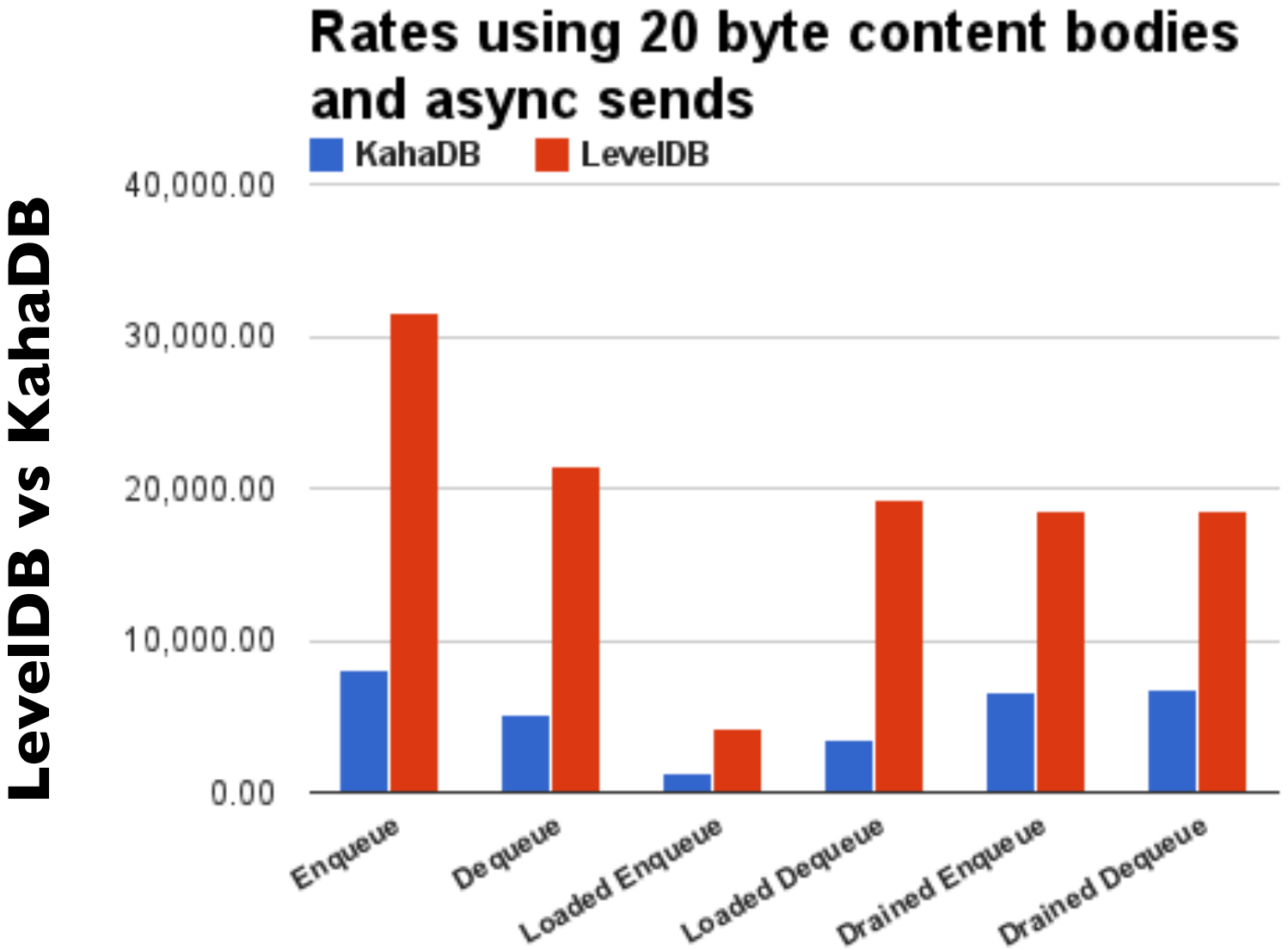
```
producer.send(session.createTextMessage("Hello"), new AsyncCallback() {
        public void onSuccess() {}

        public void onException(JMSException exception) {
                    exception.printStackTrace();
        }
});
```

ActiveMQ – Vertical Scaling

Use the LevelDB Store

- Fewer index entries per message than KahaDB
- LevelDB index out-perform B-Tree index at sequential access .
- LevelDB indexes support concurrent read access.
- Pause-less data log file garbage collection cycles.
- Fewer IO ops to load stored messages.
- Composite sends stores a message only once

ActiveMQ – Vertical Scaling



ActiveMQ – Vertical Scaling

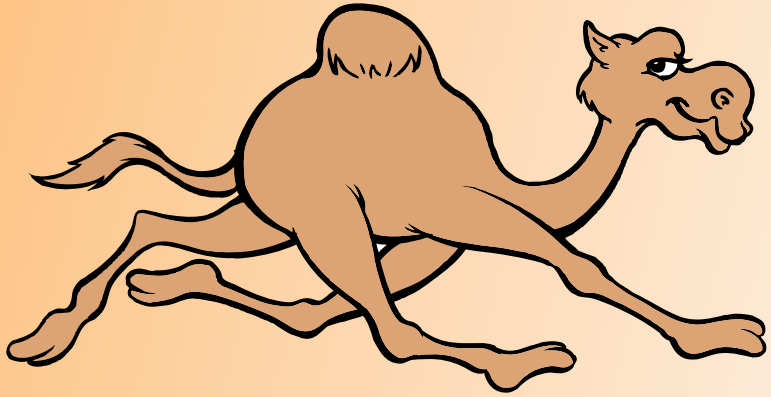
Vertical Scaling: Other things to improve Vertical Scaling

- Use the NIO transport on the ActiveMQ broker
- Increase the amount of memory available to the broker
- Reduce the default JVM stack size of each thread by use of -Xss option
- Use LevelDB!

CamelOne 2013

June 10-11 2013

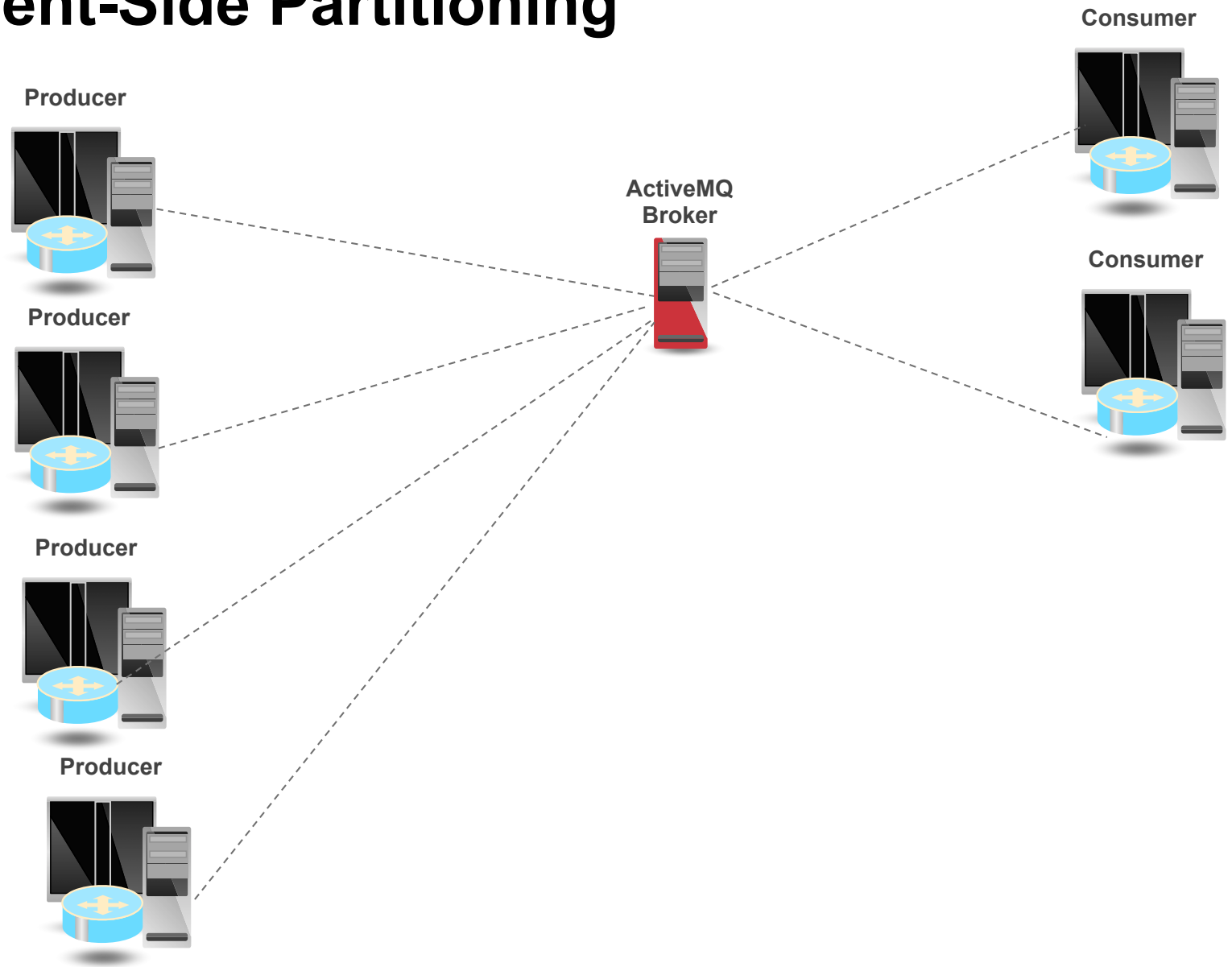
Boston, MA



ActiveMQ Horizontal Scaling

ActiveMQ – Horizontal Scaling

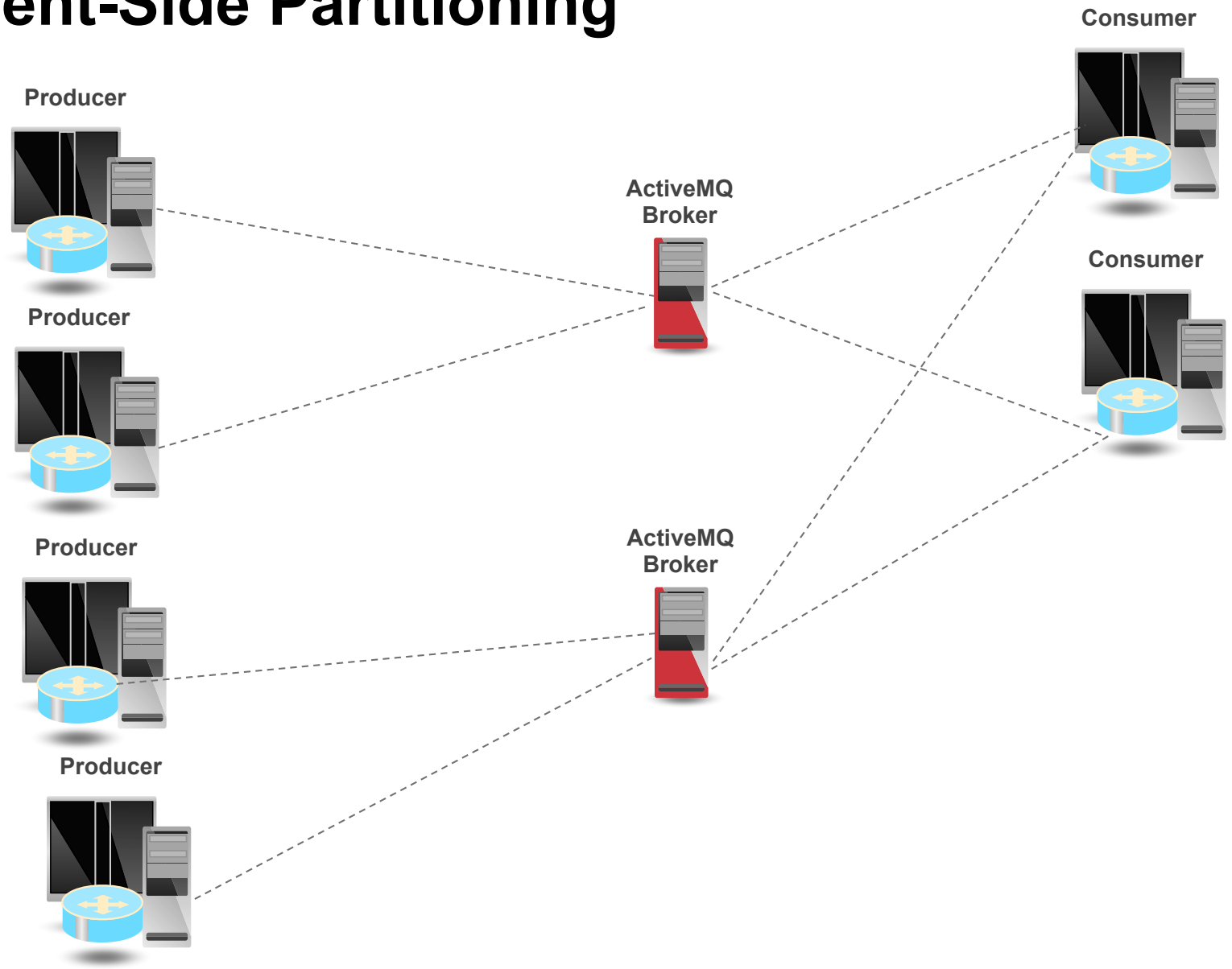
Client-Side Partitioning



CamelOne

ActiveMQ – Horizontal Scaling

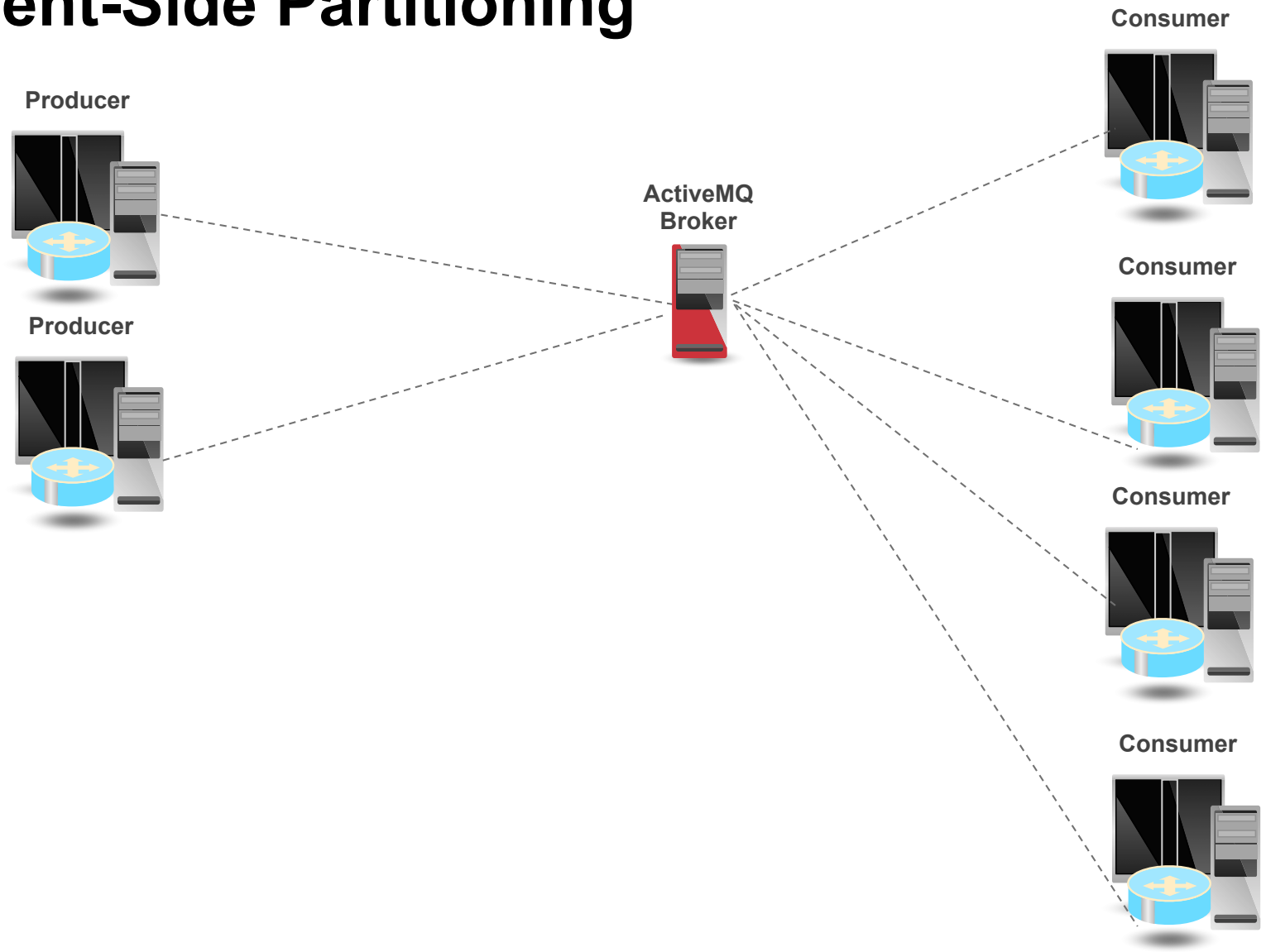
Client-Side Partitioning



CamelOne

ActiveMQ – Horizontal Scaling

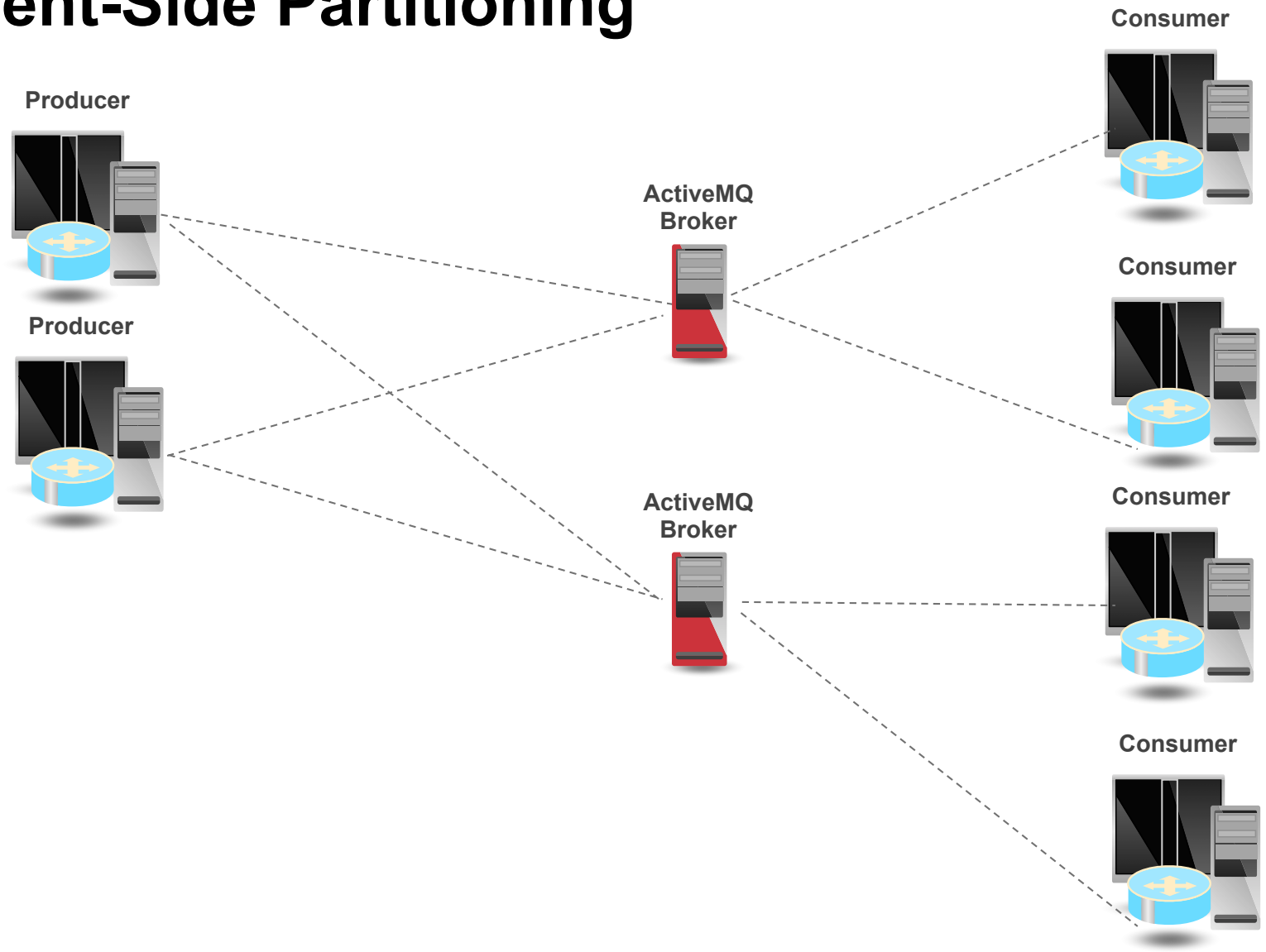
Client-Side Partitioning



CamelOne

ActiveMQ – Horizontal Scaling

Client-Side Partitioning



CamelOne

ActiveMQ – Horizontal Scaling

Client-Side Partitioning

- Multiple broker nodes are used by the clients
- Brokers are NOT networked
- The client application send message to different brokers, typically based on some defined partitioning of the data.

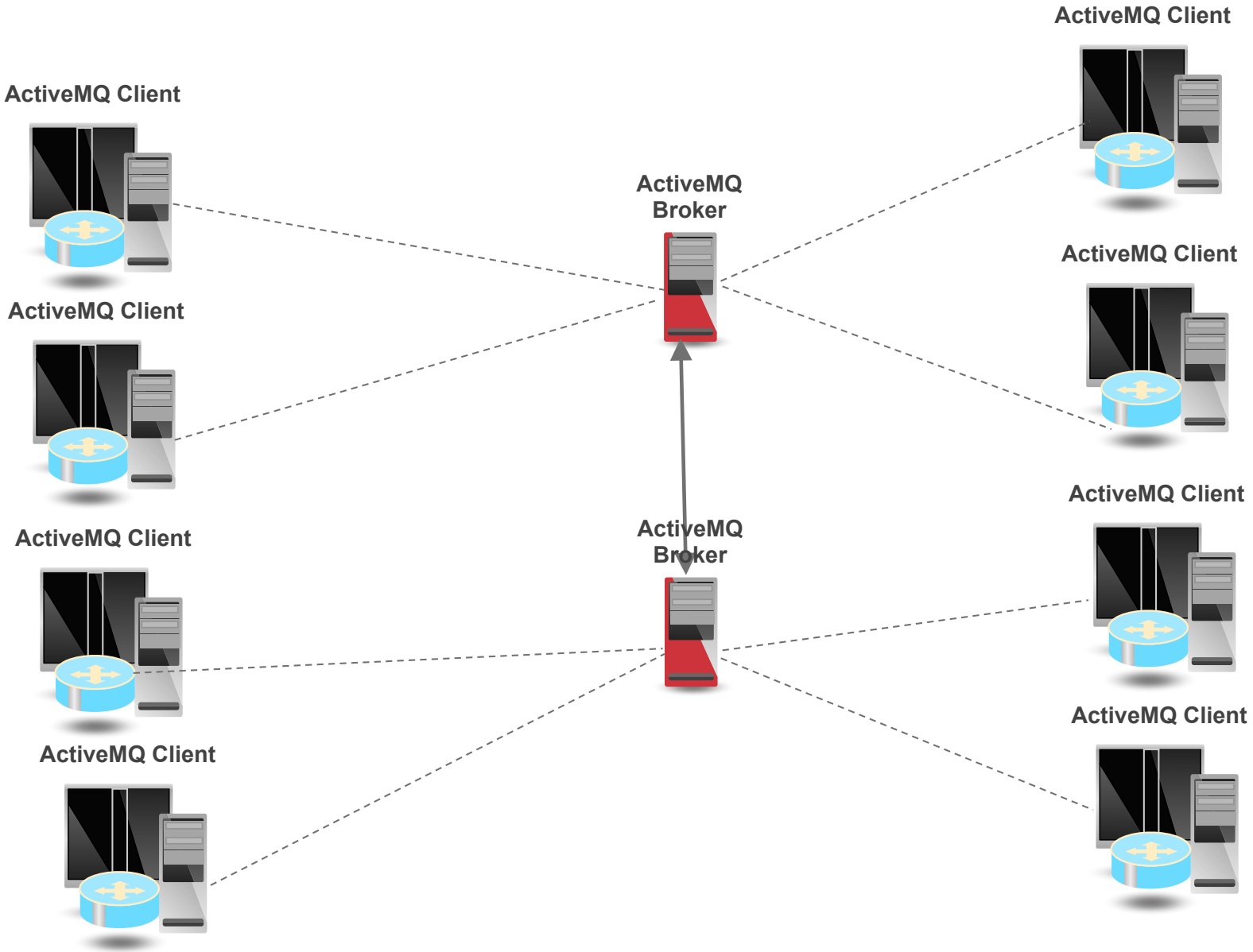
Pros

- You can use all the tuning techniques used in Vertical scaling
- Have better Horizontal scalability than using Network Of Brokers (Less broker cross talk)

Cons

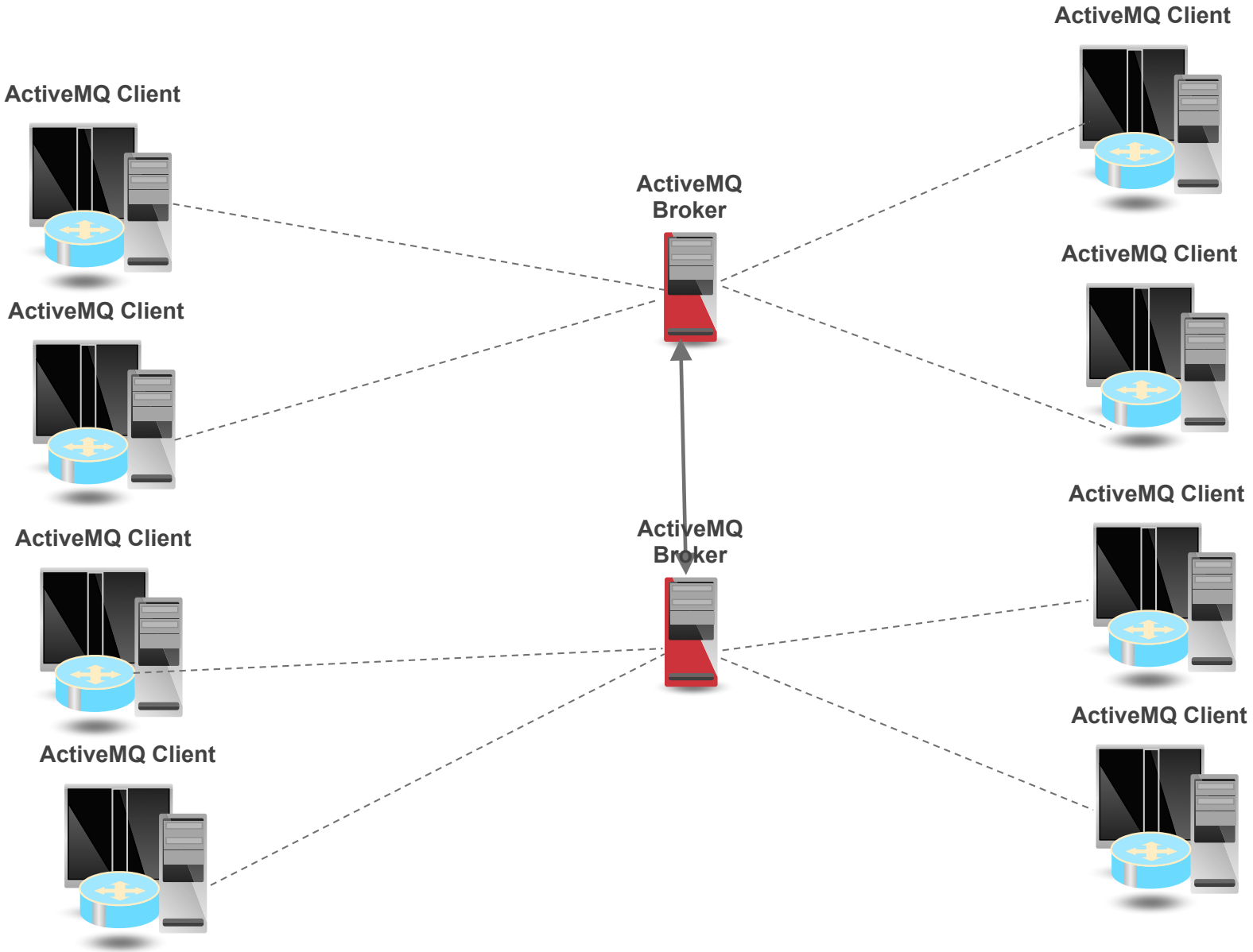
- Added complexity required on the end user Application

ActiveMQ – Horizontal Scaling



CamelOne

ActiveMQ – Horizontal Scaling



CamelOne

ActiveMQ – Horizontal Scaling

Horizontal Scaling: Increase load capacity using many brokers

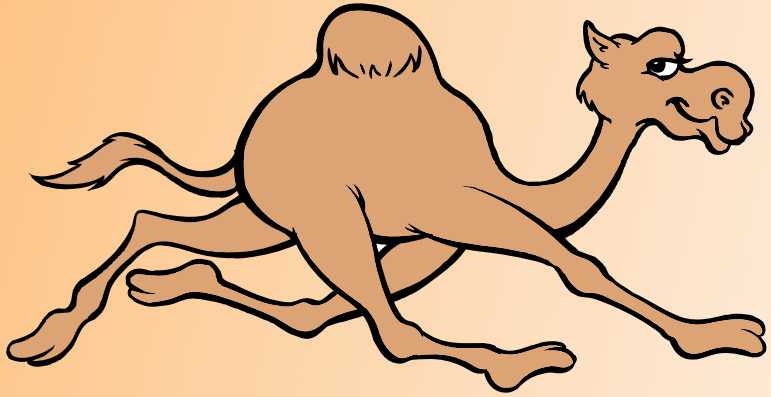
- Use ActiveMQ Networks
- Messages are forwarded between brokers with interested consumers
- Networks lift the limits of using a single machine

- Problems:
  - Complex topologies can lead to non-optimal message routing
  - Orphaned Messages on failure (use networks and clusters)

CamelOne 2013

June 10-11 2013

Boston, MA



Apache ActiveMQ Broker Topologies

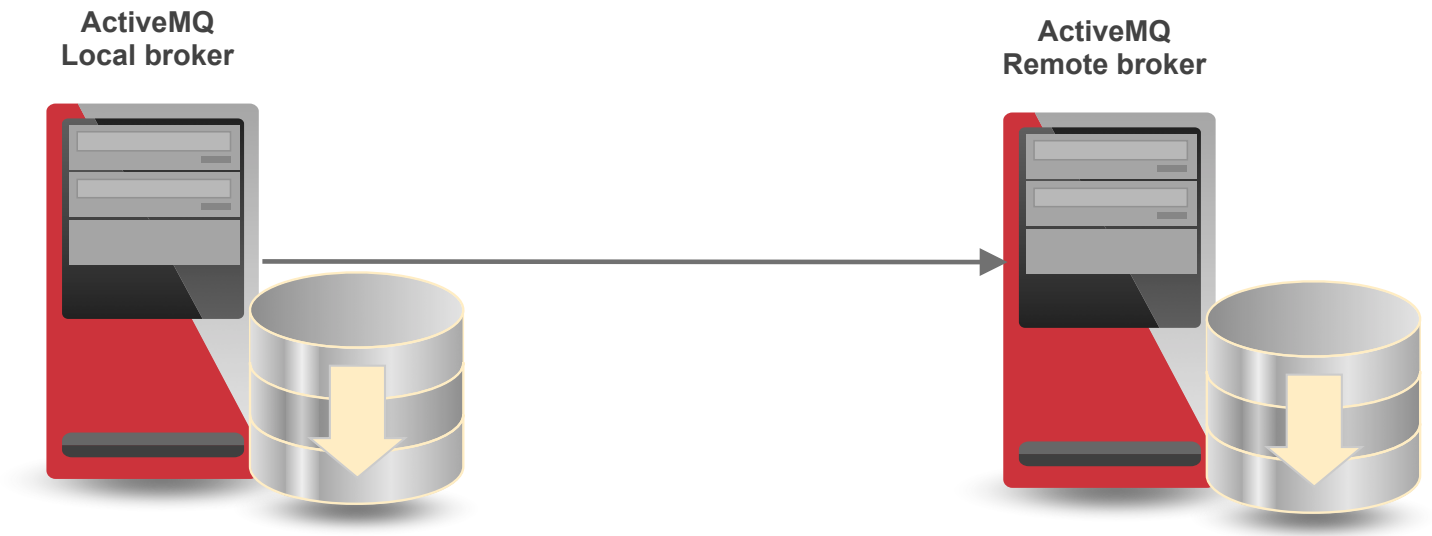
ActiveMQ – Broker Topologies

Networks

- Link ActiveMQ Brokers together
- Use Store and Forward
- Are uni-directional by default
- All Destinations are global

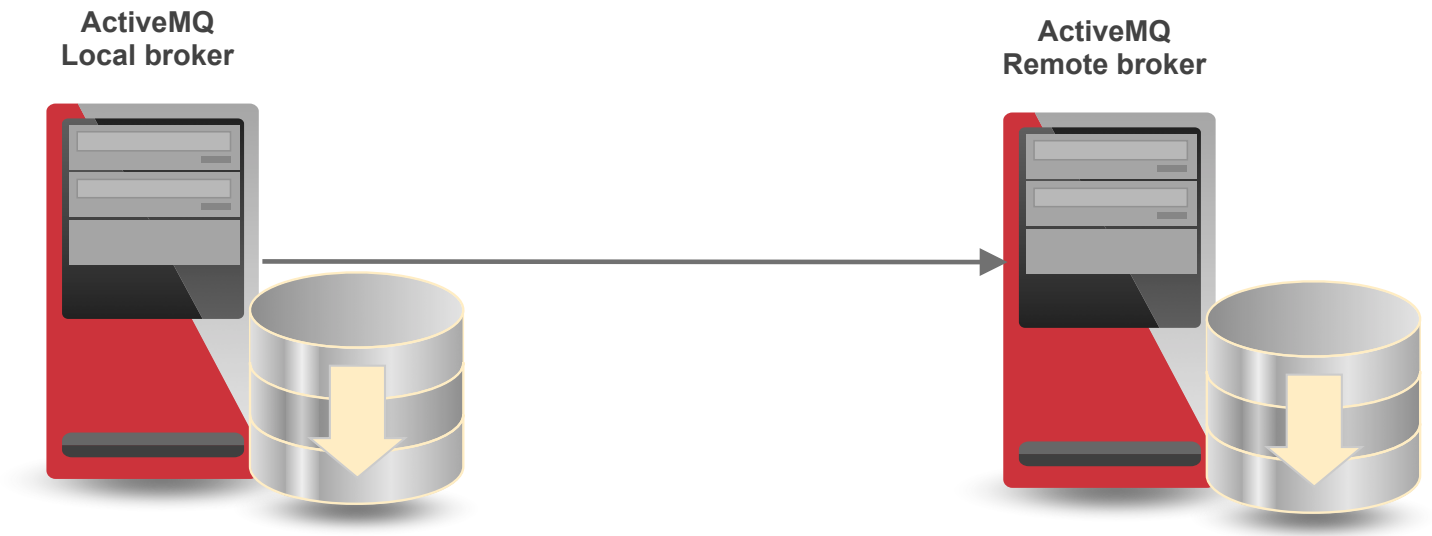
ActiveMQ – Broker Topologies

Store and Forward



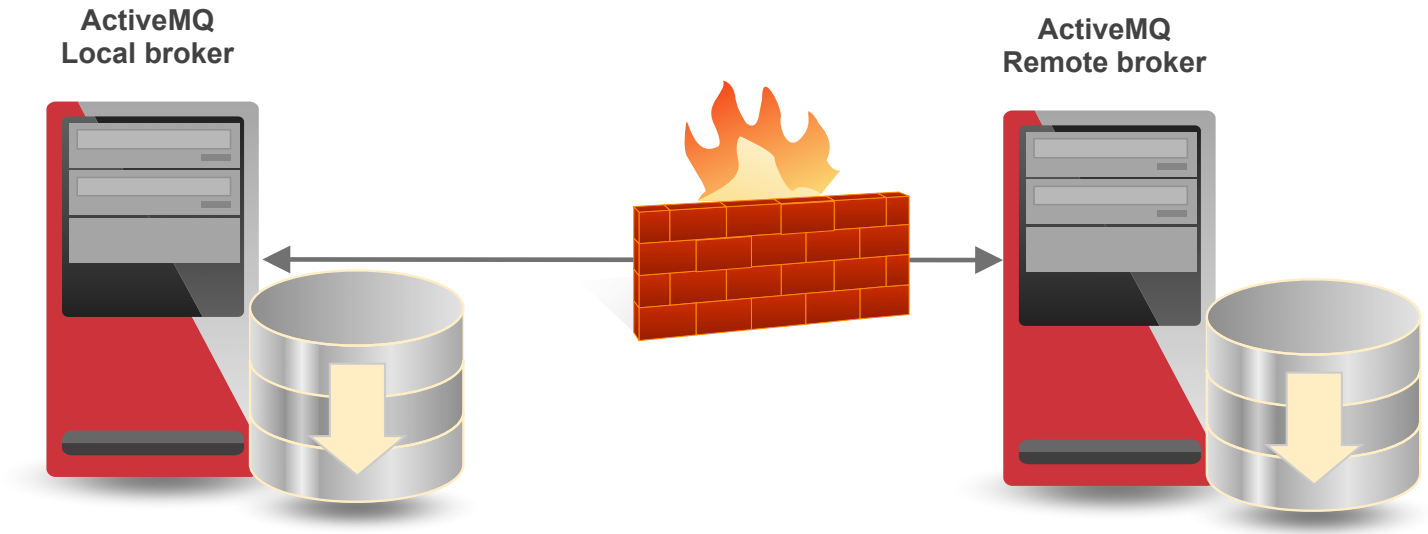
ActiveMQ – Broker Topologies

Store and Forward



ActiveMQ – Broker Topologies

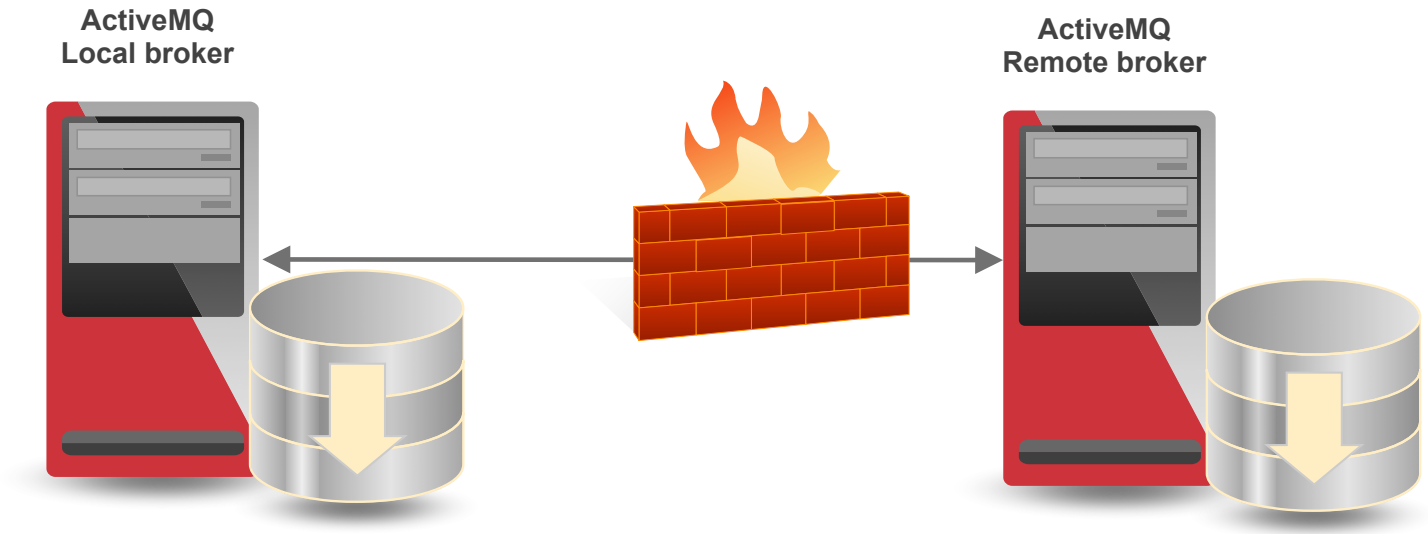
Bi-directional network



CamelOne

ActiveMQ – Broker Topologies

Bi-directional network



CamelOne

ActiveMQ – Broker Topologies

Bi-directional network - Configuration

```
<networkConnectors>
    <networkConnector name="backoffice"
      uri="static://(tcp://backoffice:61617)"
      duplex="true">
    </networkConnector>
</networkConnectors>
```

CamelOne

ActiveMQ – Broker Topologies

Networks – Configuration

- Filters: dynamicallyIncludedDestinations

```
<networkConnectors>
    <networkConnector uri="static:(tcp://remote:61617) "/>
     <dynamicallyIncludedDestinations>
         <queue physicalName=”free.food.>"/>
         <queue physicalName=”free.beer.>"/>
         <topic physicalName=”cricket.scores.>"/>
      </dynamicallyIncludedDestinations>
</networkConnectors>
```

ActiveMQ – Broker Topologies

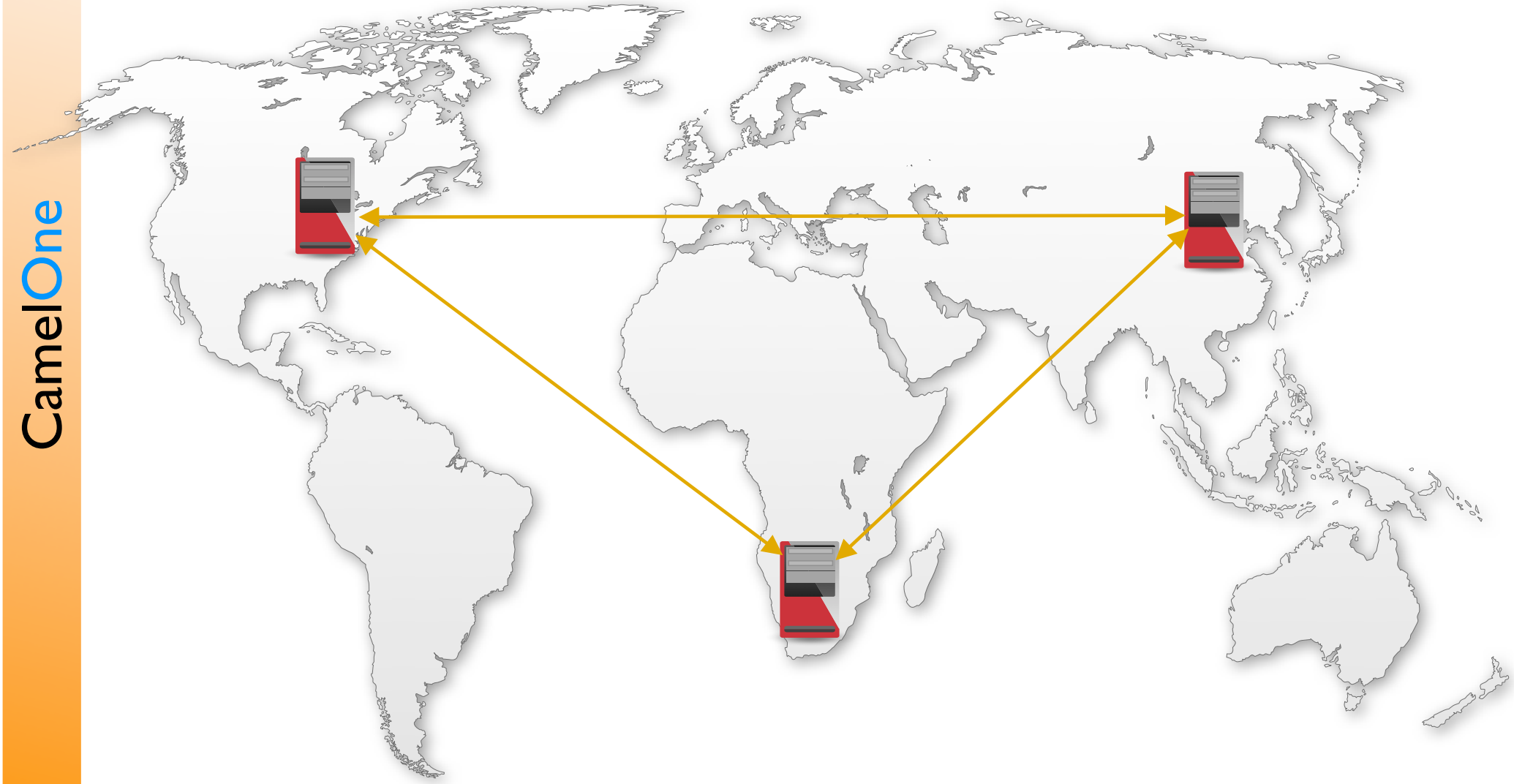
Networks – Configuration

- Filters: staticallyIncludedDestinations

```
<networkConnectors>
    <networkConnector uri="static:(tcp://remote:61617)?
useExponentialBackOff=false"/>
      <staticallyIncludedDestinations>
          <queue physicalName="management.queue-1"/>
          <queue physicalName="management.queue-2"/>
          <queue physicalName="global.>"/>
          <topic physicalName="global.>"/>
       </staticallyIncludedDestinations>
 </networkConnectors>
```

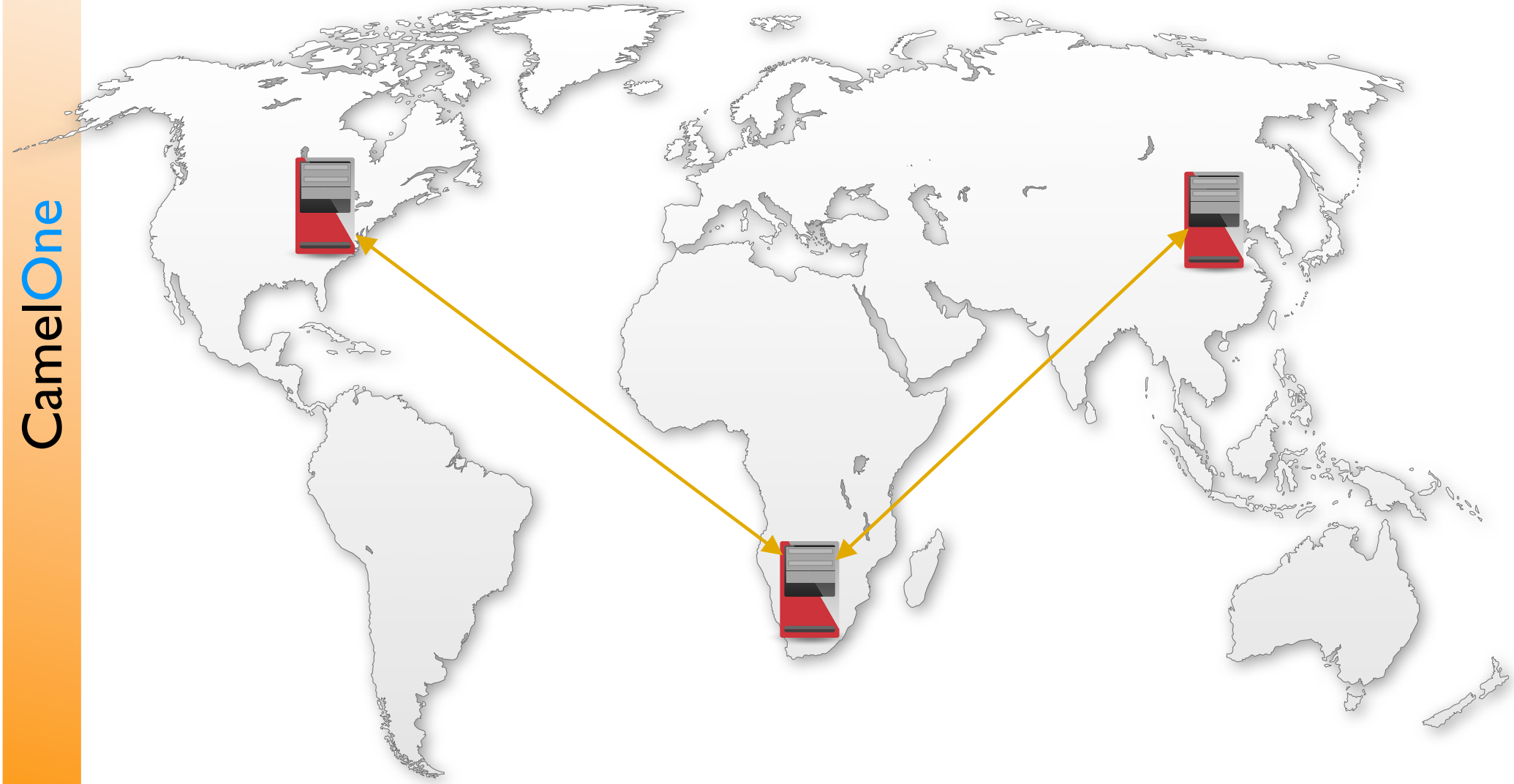
ActiveMQ – Broker Topologies

Networks – Configuration – networkTTL=2



ActiveMQ – Broker Topologies

Networks – Configuration – networkTTL=2



ActiveMQ – Geographically dispersed data centers:redundant links – Topic support

Enable duplicate subscriptions over the network:

```
<networkConnector uri="static:(tcp://brokerB:61617)" name="A-B"
   networkTTL="2" suppressDuplicateTopicSubscriptions="false"/>
<networkConnector uri="static:(tcp://brokerC:61618)" name="A-C"
   networkTTL="2" suppressDuplicateTopicSubscriptions="false"/>
```

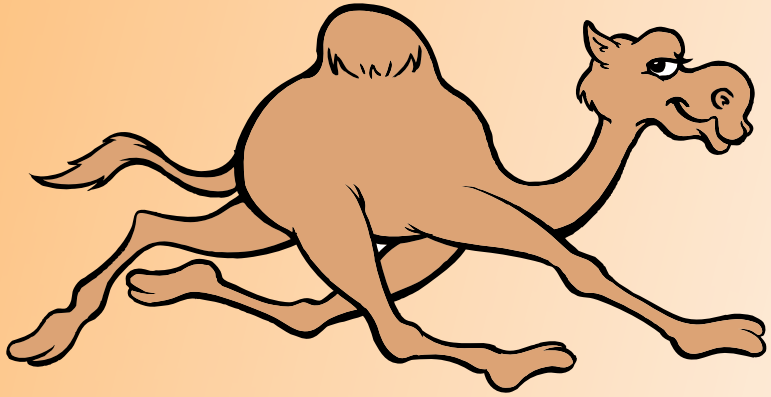
Ensure every Topic message is only sent through one network connection - the one with the highest priority:

```
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry topic=">" enableAudit="true">
        <dispatchPolicy>
          <priorityNetworkDispatchPolicy/>
        </dispatchPolicy>
      </policyEntry>
    </policyEntries>
  </policyMap>
</destinationPolicy
```

CamelOne 2013

June 10-11 2013

Boston, MA

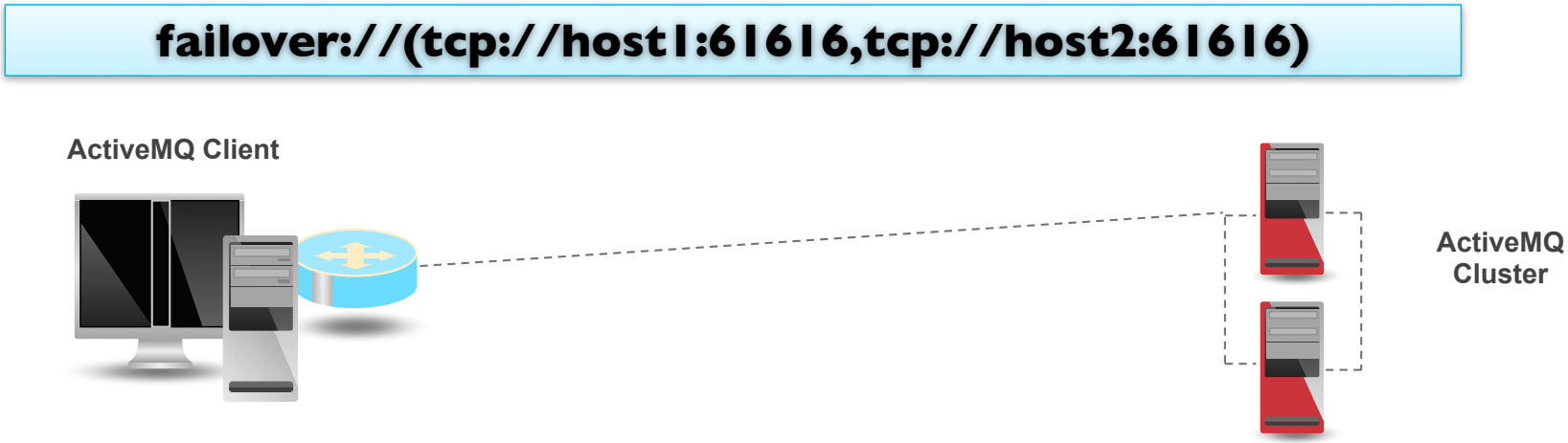


Apache ActiveMQ High Availability

ActiveMQ – Enterprise Features

Seamless Client Failover Support in

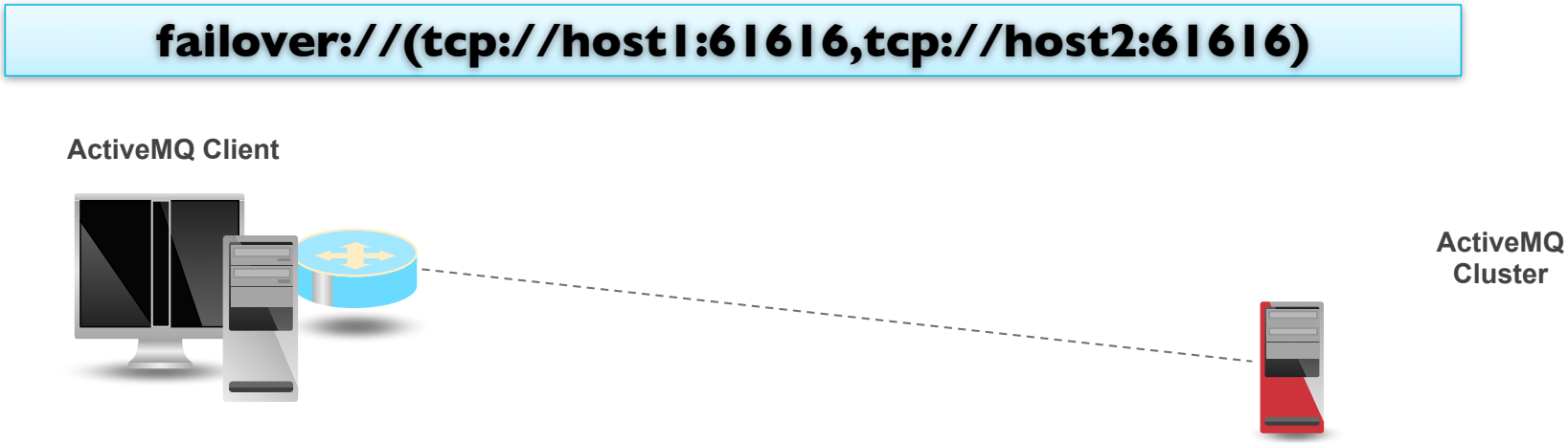
- Java
- C++
- C#



ActiveMQ – Enterprise Features

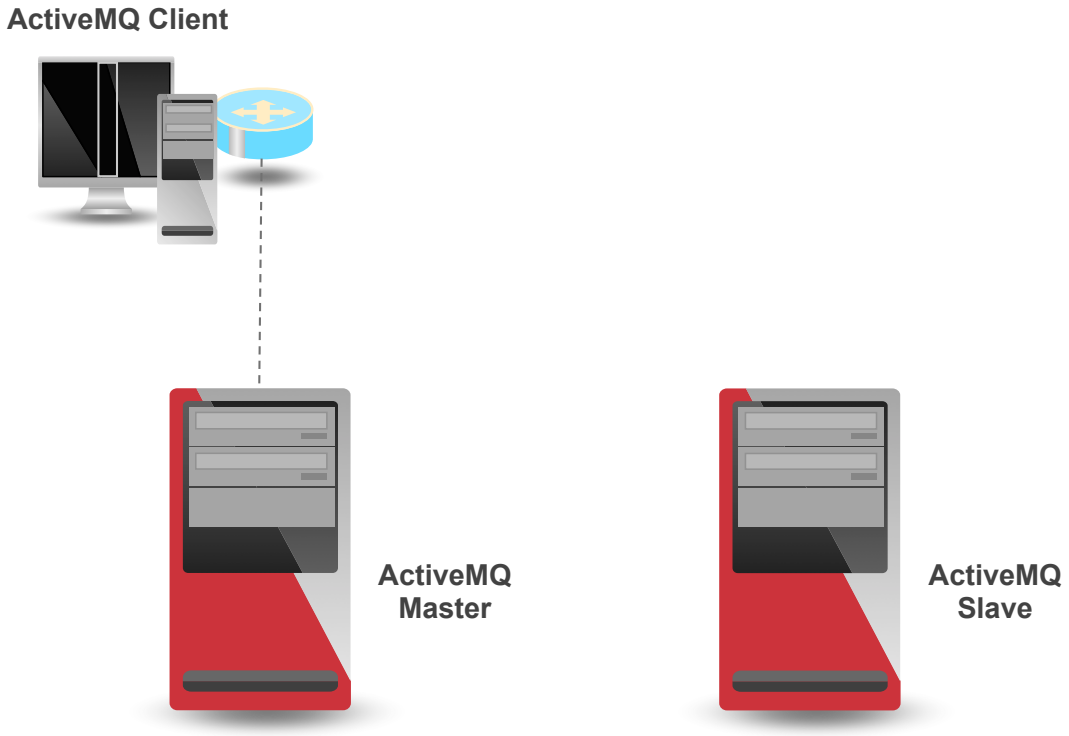
Seamless Client Failover Support in

- Java
- C++
- C#



ActiveMQ – High Availability

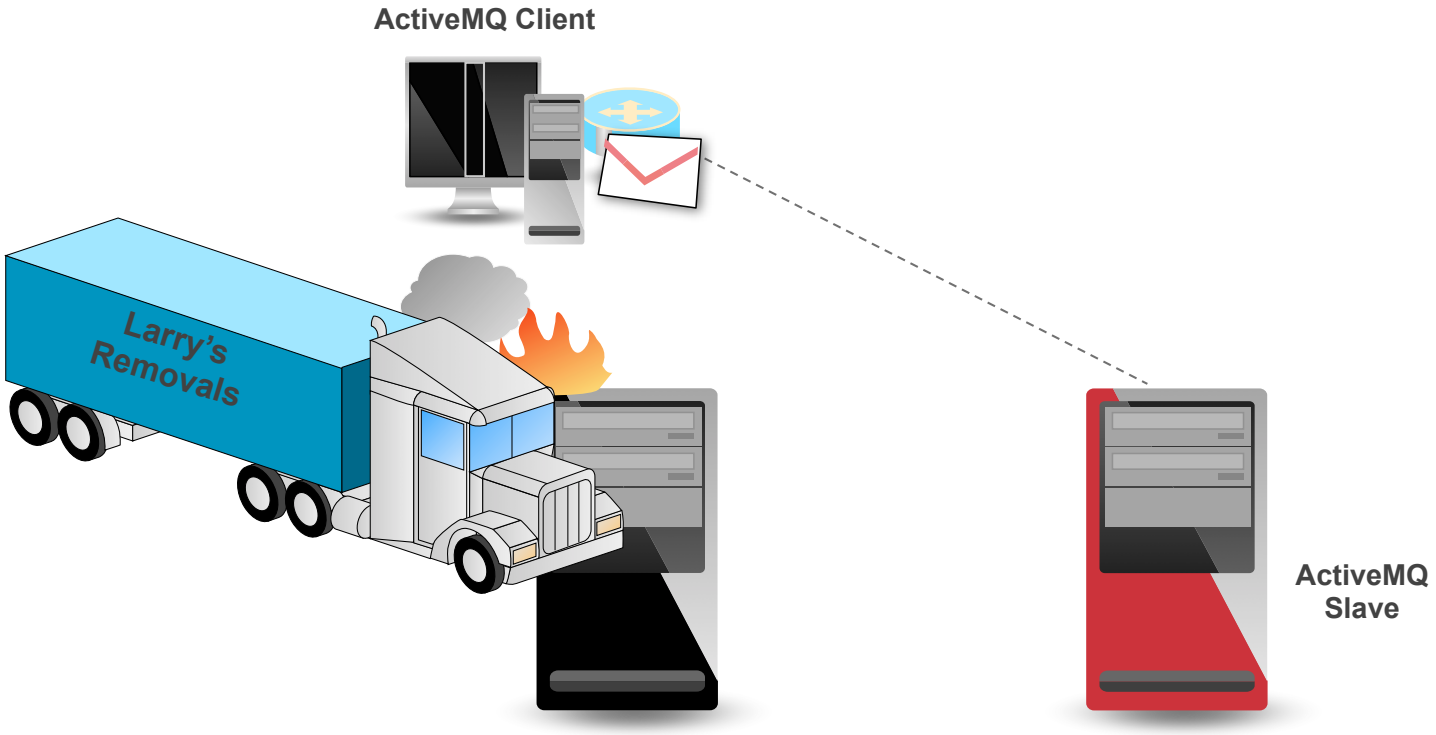
Master/Slave Configuration



CamelOne

ActiveMQ – High Availability

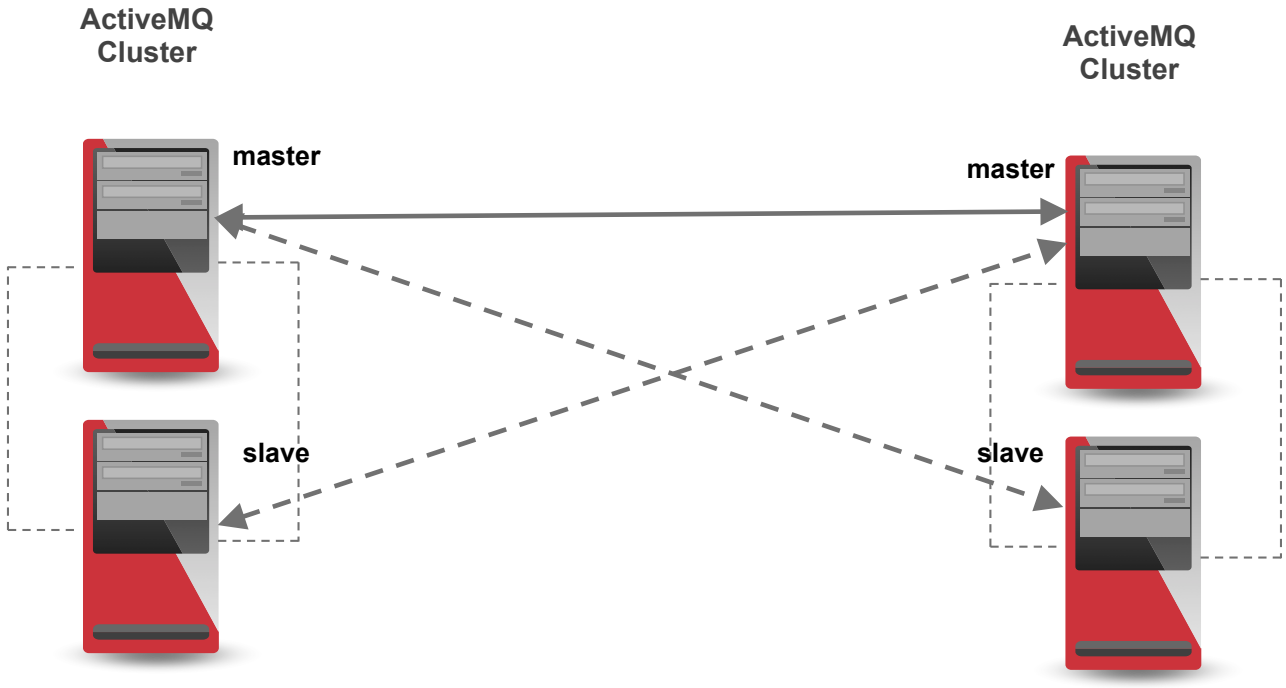
Master/Slave Configuration



CamelOne

ActiveMQ – Broker Topologies

Combining HA and Networks



CamelOne

ActiveMQ – Broker Topologies

Configuration for master/slave (from 5.6)

```
<networkConnectors>
    <networkConnector uri="masterslave:(tcp://master,tcp://slave)"/>
</networkConnectors>
```

Which is the same as:

```
<networkConnectors>
    <networkConnector uri=”static:failover:(tcp://master,tcp://slave)?
randomize=false&maxReconnectAttempts=0"/>
</networkConnectors>
```

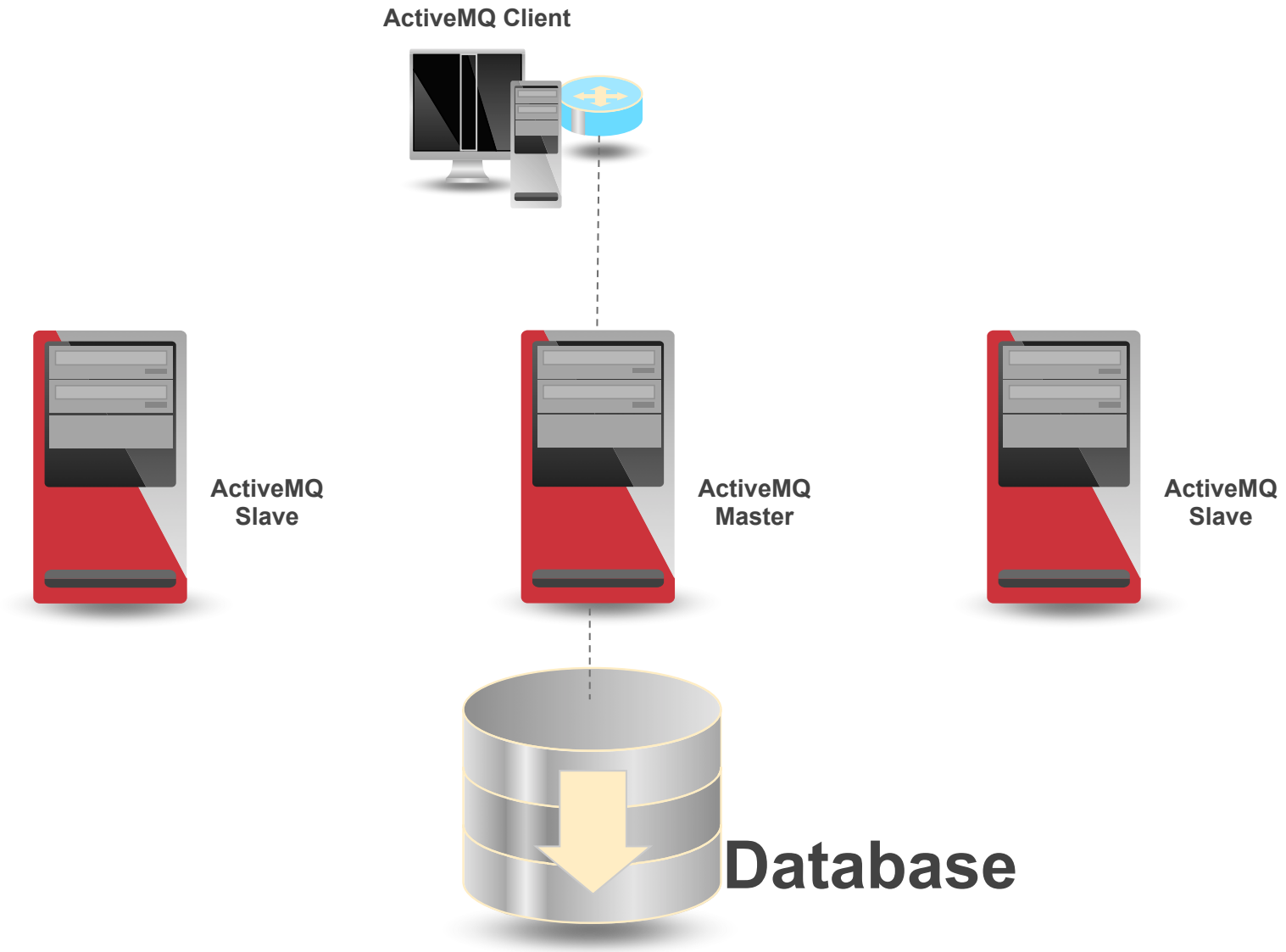
ActiveMQ – High Availability

Three types of Master/Slave.

- JDBC Master/Slave
- Shared File System Master/Slave
- Replicated LevelDB Master/Slave

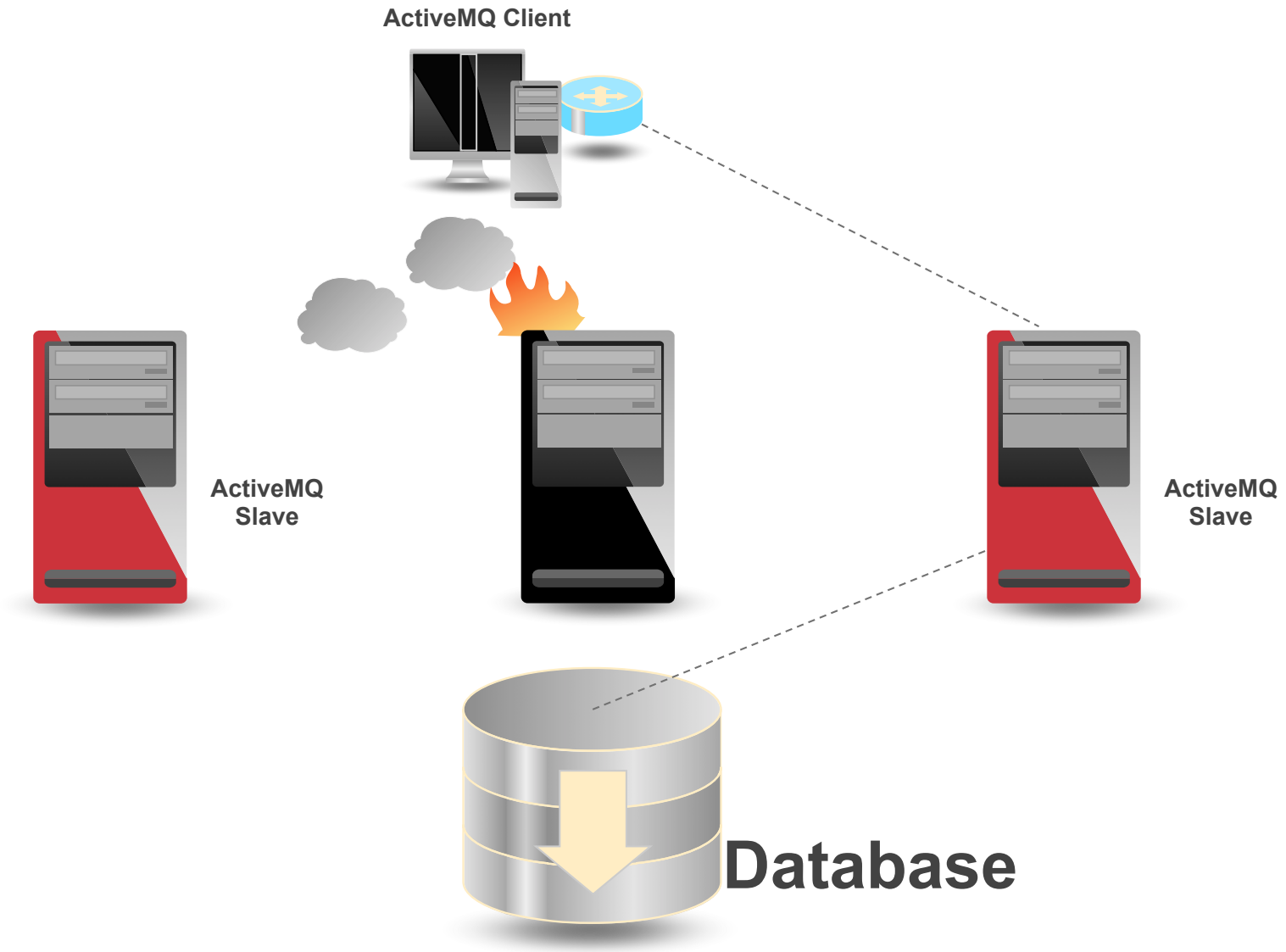
ActiveMQ – High Availability

JDBC Master/Slave



ActiveMQ – High Availability

JDBC Master/Slave



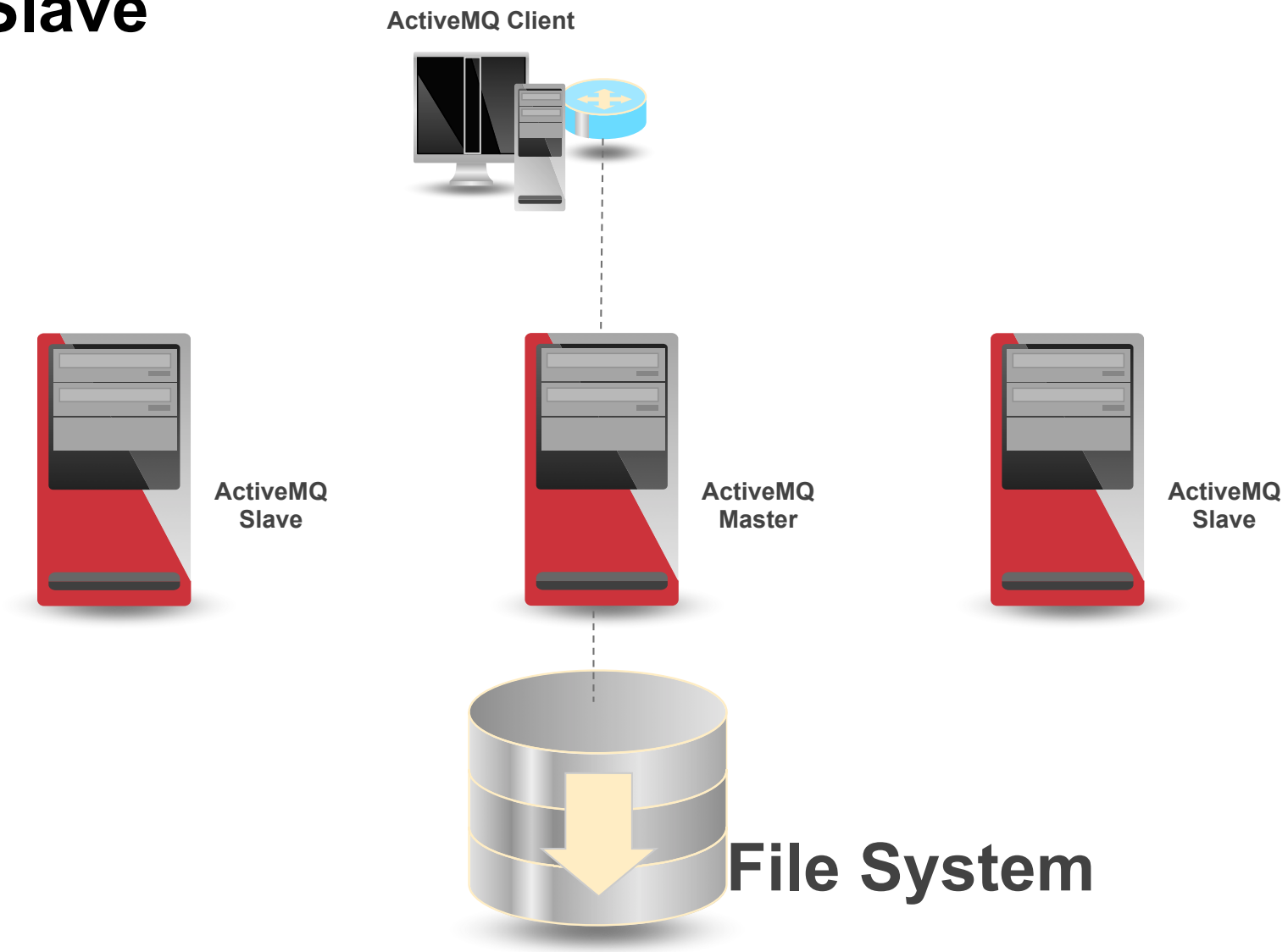
ActiveMQ – High Availability

JDBC Master/Slave

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

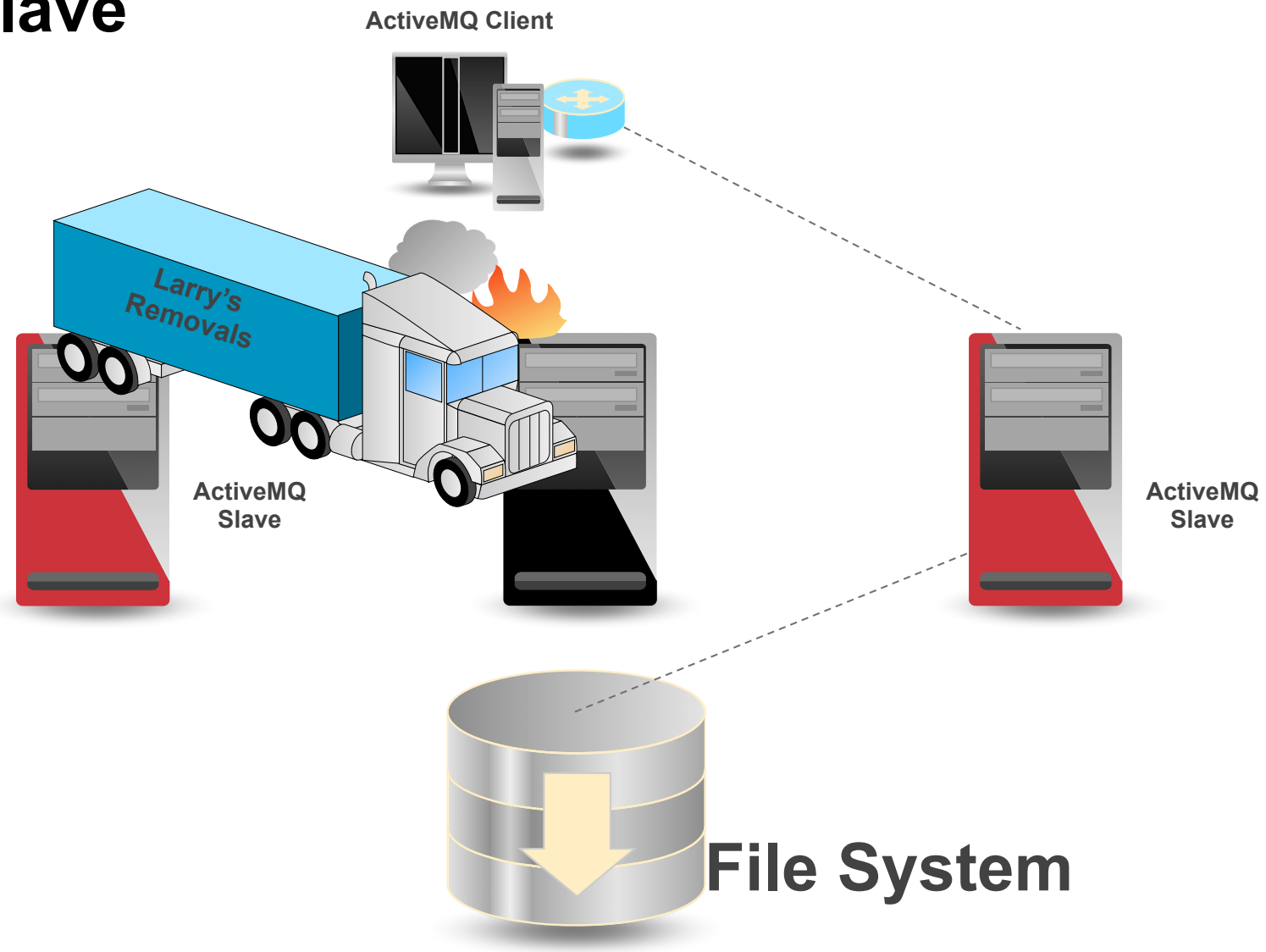
ActiveMQ – High Availability

Shared File System Master/Slave



ActiveMQ – High Availability

Shared File System Master/Slave



CamelOne

ActiveMQ – High Availability

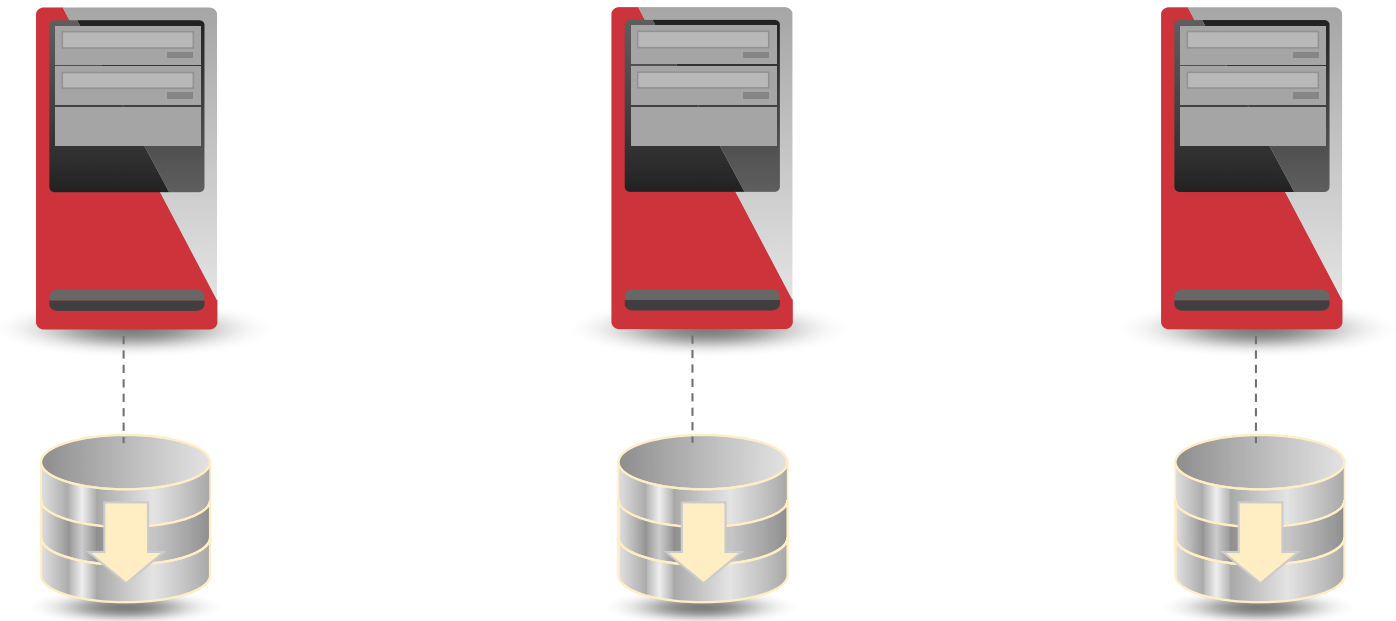
Shared File System Master/Slave

- Recommended if you have a SAN, or DRDB
- No restriction on number of slaves
- Simple configuration
- Ensure file locking works – and times out – NFSv4 good!

ActiveMQ – High Availability

Replicated LevelDB Master/Slave

ZooKeeper Cluster

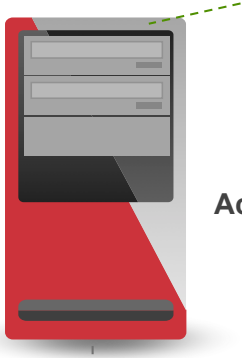


Local File System | Local File System | Local File System

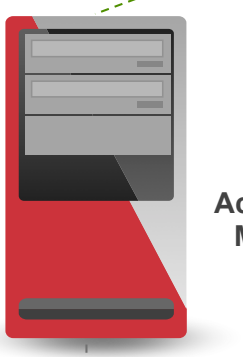
ActiveMQ – High Availability

Replicated LevelDB Master/Slave

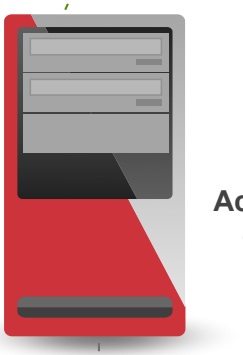
ZooKeeper Cluster



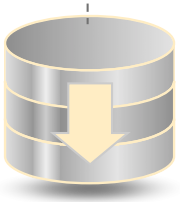
ActiveMQ Slave



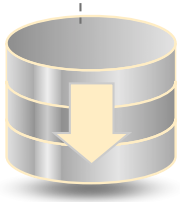
ActiveMQ Master



ActiveMQ Slave



Local File System



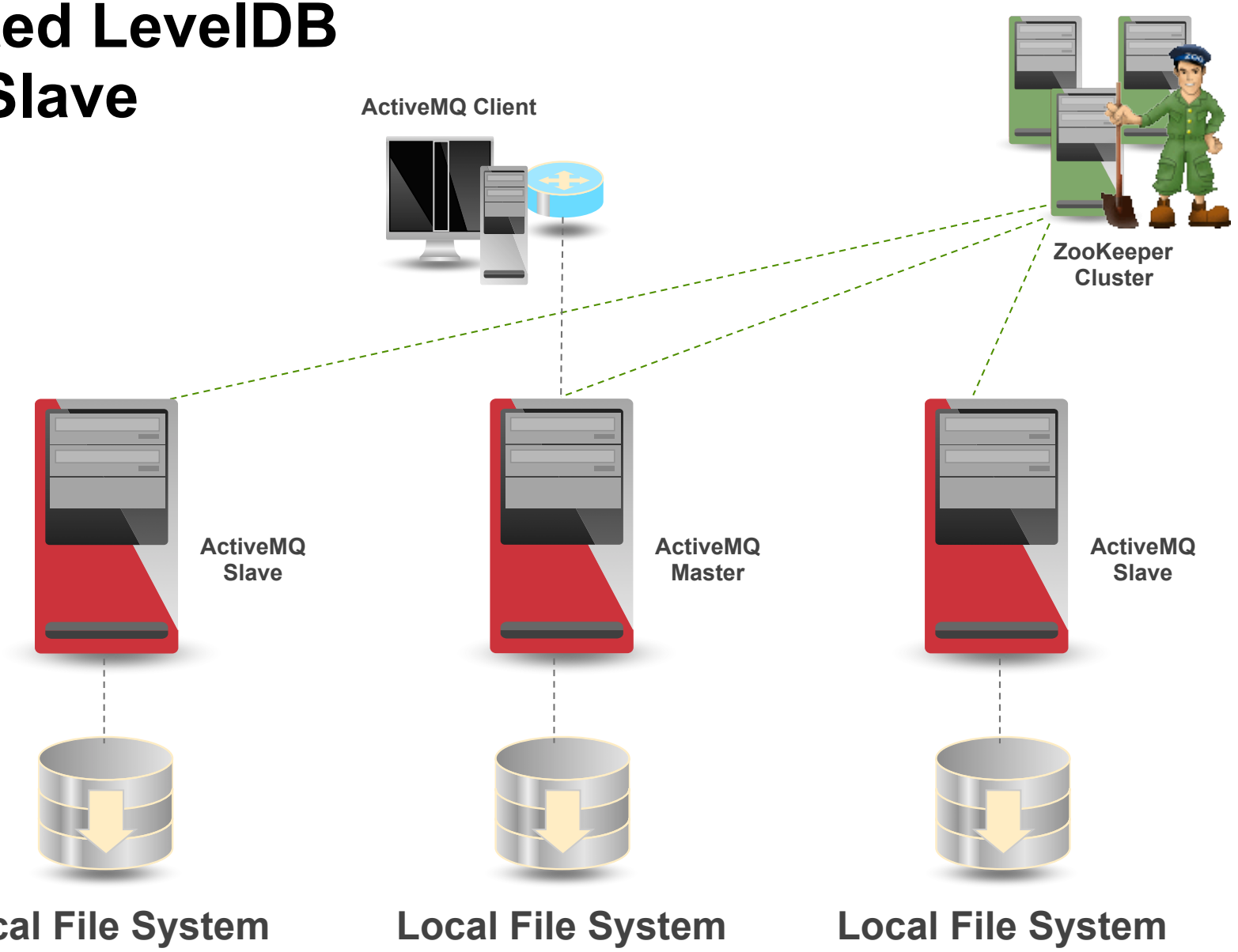
Local File System

Local File System

CamelOne

ActiveMQ – High Availability

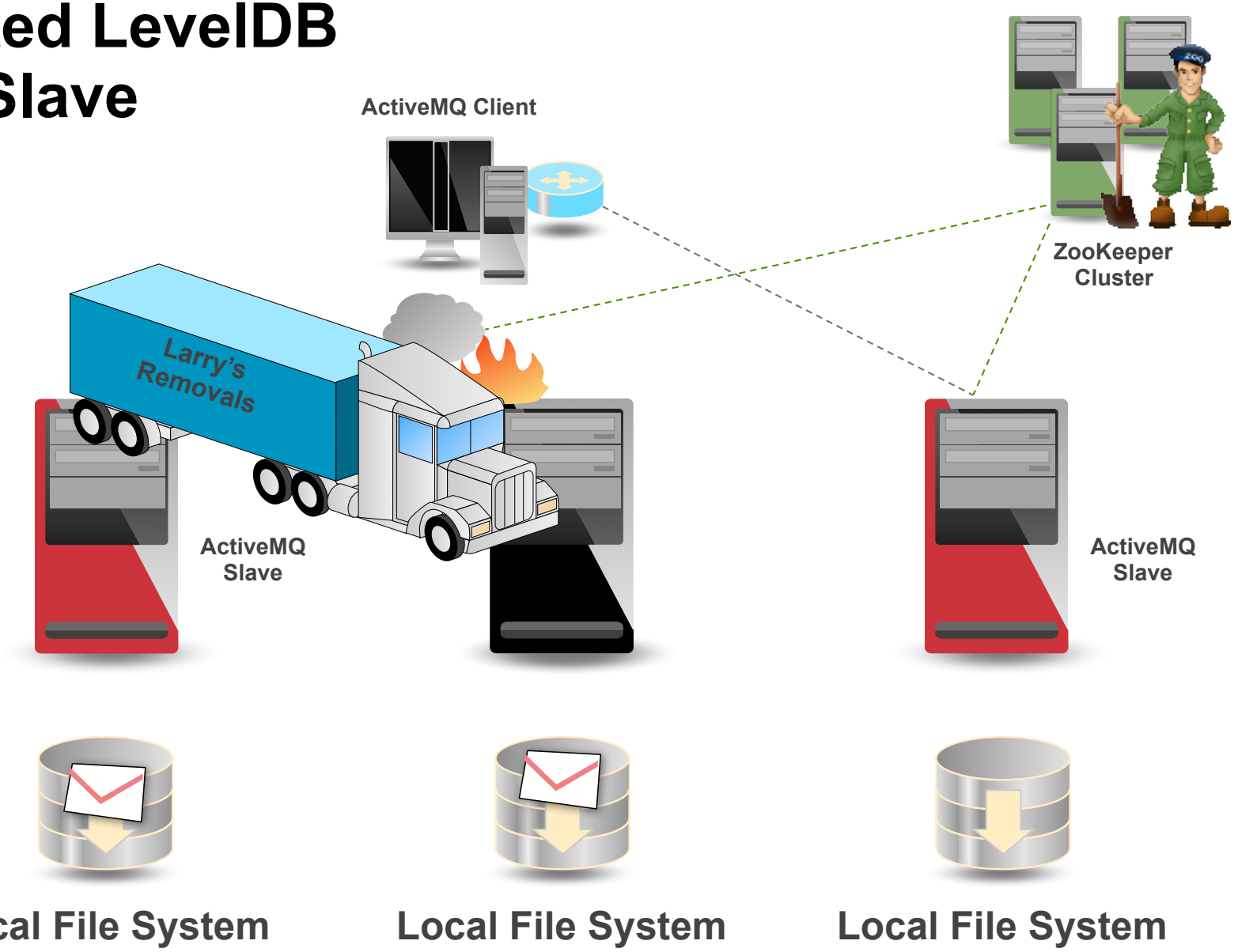
Replicated LevelDB Master/Slave



CamelOne

ActiveMQ – High Availability

Replicated LevelDB Master/Slave



CamelOne

ActiveMQ – High Availability

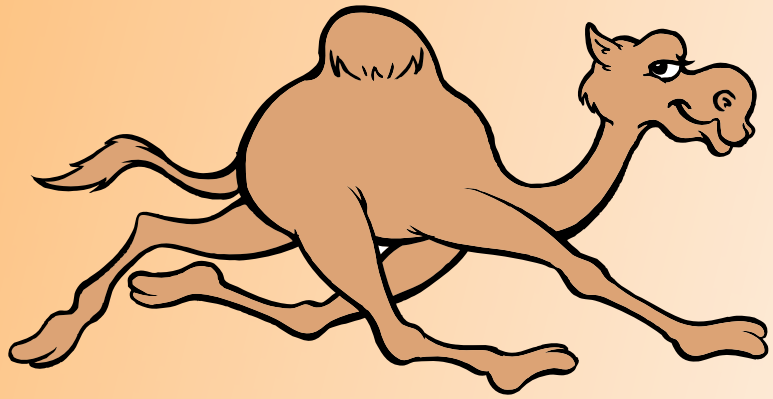
Replicated LevelDB Master/Slave

- Requires a HA Apache ZooKeeper Cluster
- No Single Point of Failure
- Dynamic number of slaves
- Simple configuration
- Sync or Async Replication

CamelOne 2013

June 10-11 2013

Boston, MA



Questions?