

# **JBoss Messaging – present and future**

Tim Fox  
Messaging Lead, JBoss

# **Agenda**

- What is JBoss Messaging?
- JBM 1.4 current features
- What's new in JBM 2.0?
- JBM 2.0 feature drill-down

# **What is JBoss Messaging?**

- Generic, reliable, well featured, asynchronous open source messaging system
- Fully compliant JMS 1.1 implementation
- Fully JEE 5 compliant
- Many features over and above JMS
- 100% LGPL code.

## **What is JBoss Messaging (continued)**

- JBM is the default JMS provider in JBoss Enterprise Application Platform 4.3.
- JBM is the default JMS provider in JBoss SOA Platform.
- JBM will be the default JMS provider in JBoss AS 5.0 and later
- Current production release is JBM 1.4.1
- Currently used in production by many customers including several household names and investment banks.
- Active user community.
- Next major release JBM 2.0

## **Can I get support for JBM?**

- Full production support is available for JBM as part of the JBoss Enterprise Application Platform (EAP) subscription, and as part of the JBoss SOA Platform subscription.
- The support engineers are JBM experts and contributors, and the core development team are JBoss employees, so we can provide a very high level of support – direct from the experts.

# Who is JBoss Messaging?

- JBoss maintains a team of core engineers and support engineers to work full time on JBoss Messaging. JBM team has expanded recently.
- Four core engineers

Tim Fox - Project lead  
Andy Taylor – ex Arjuna and HP  
Clebert Suconic  
Jeff Mesnil - ObjectWeb/JonAS/JOTM

- Three support engineers

Jay Howell  
Mike Clark  
Tyronne Wickramaratne

## **JBM 1.4 (current version) features**

- Fully JMS 1.1 and JEE5 compliant JMS implementation
- Clustered queues and topics
- Clustered durable subscriptions
- Message redistribution
- Transparent fail-over
- Fully functional JMS message bridge
- Delayed redelivery
- Limited redelivery attempts

## **JBM 1.4 features continued**

- Expiry Queues
- Dead Letter Queues
- Scheduled delivery
- JMX interface
- Pluggable JAAS security
- Transports: TCP, SSL, HTTP
- JDBC persistence via shared database – supported databases: Oracle, MySQL (InnoDB), PostgreSQL, Sybase ASE, MS SQL Server

## **JBM 1.4 features continued**

- Full XA implementation and integration with JBoss Transactions
- Message statistics
- Very large queue support

## **JBM 2.0 - the next generation**

- Team currently working on JBM 2.0
- GA release towards end of year. (Provisional)
- Builds on JBoss Messaging 1.4
- JBM 2.0 goal – To be the best performing, fullest featured open source clustered reliable messaging implementation.
- Will produce performance comparisons against our competitors. We don't want to make claims we can't substantiate.
- A generic, embeddable messaging implementation

## **What's new in JBM 2.0?**

- Completely JMS independent generic core.
- Boot-strappable in JBoss MicroContainer or any other dependency injection framework.
- Will be embeddable in third party applications – OEMs.
- Fast journal store using Java NIO
- Fast journal store using Linux asynchronous IO
- Improved JDBC persistence support using Hibernate – supports any database that Hibernate supports.

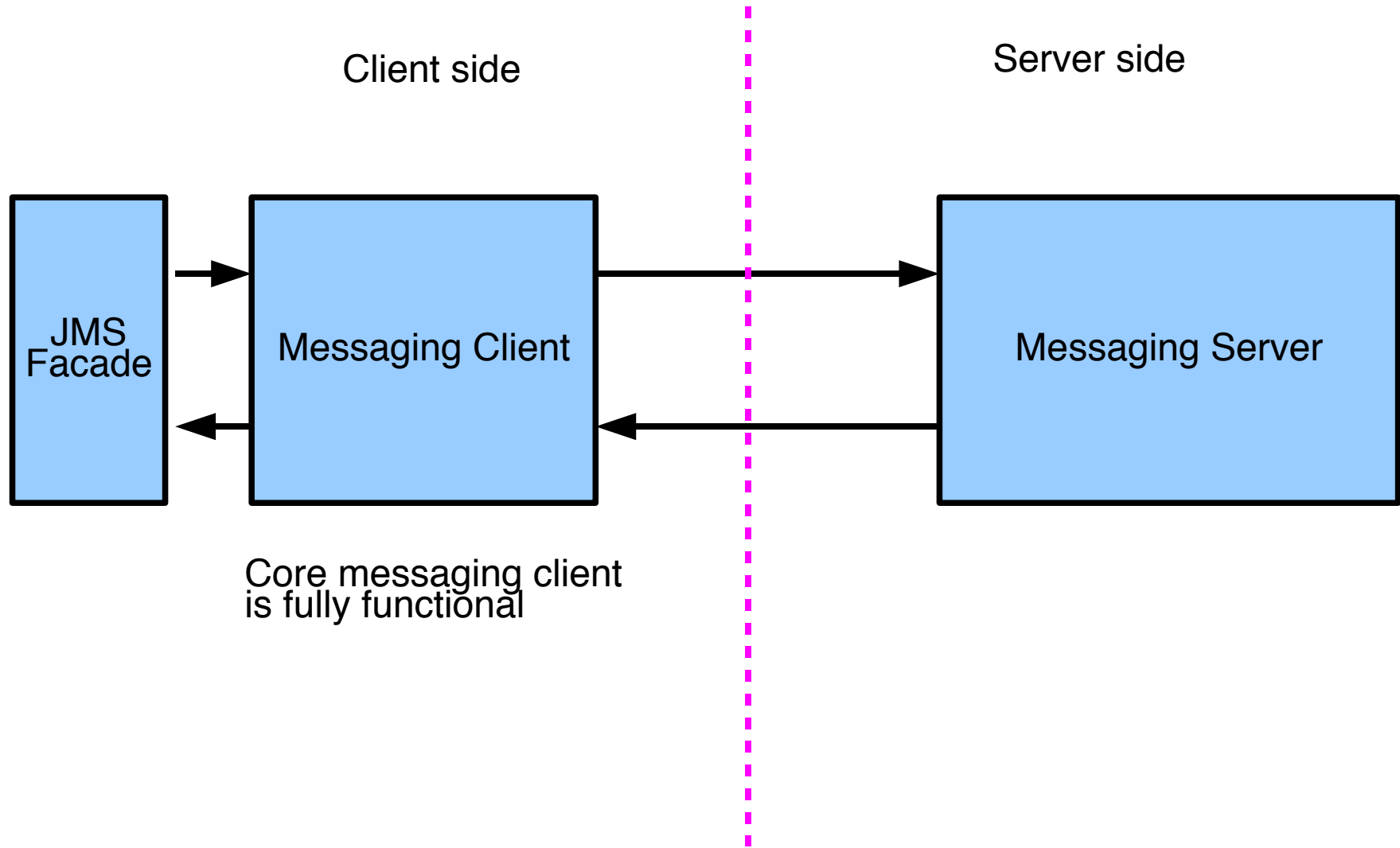
## **What's new in JBM 2.0 (continued)**

- Extended and more flexible HA.
- Brand new NIO transport using Apache MINA. Supports TCP, SSL, HTTP and APR.
- Improved and more flexible queue configuration and security.
- Producer flow control.
- Topic hierarchies (wild-card routing)
- Improved management api
- Very large message support
- Many other features

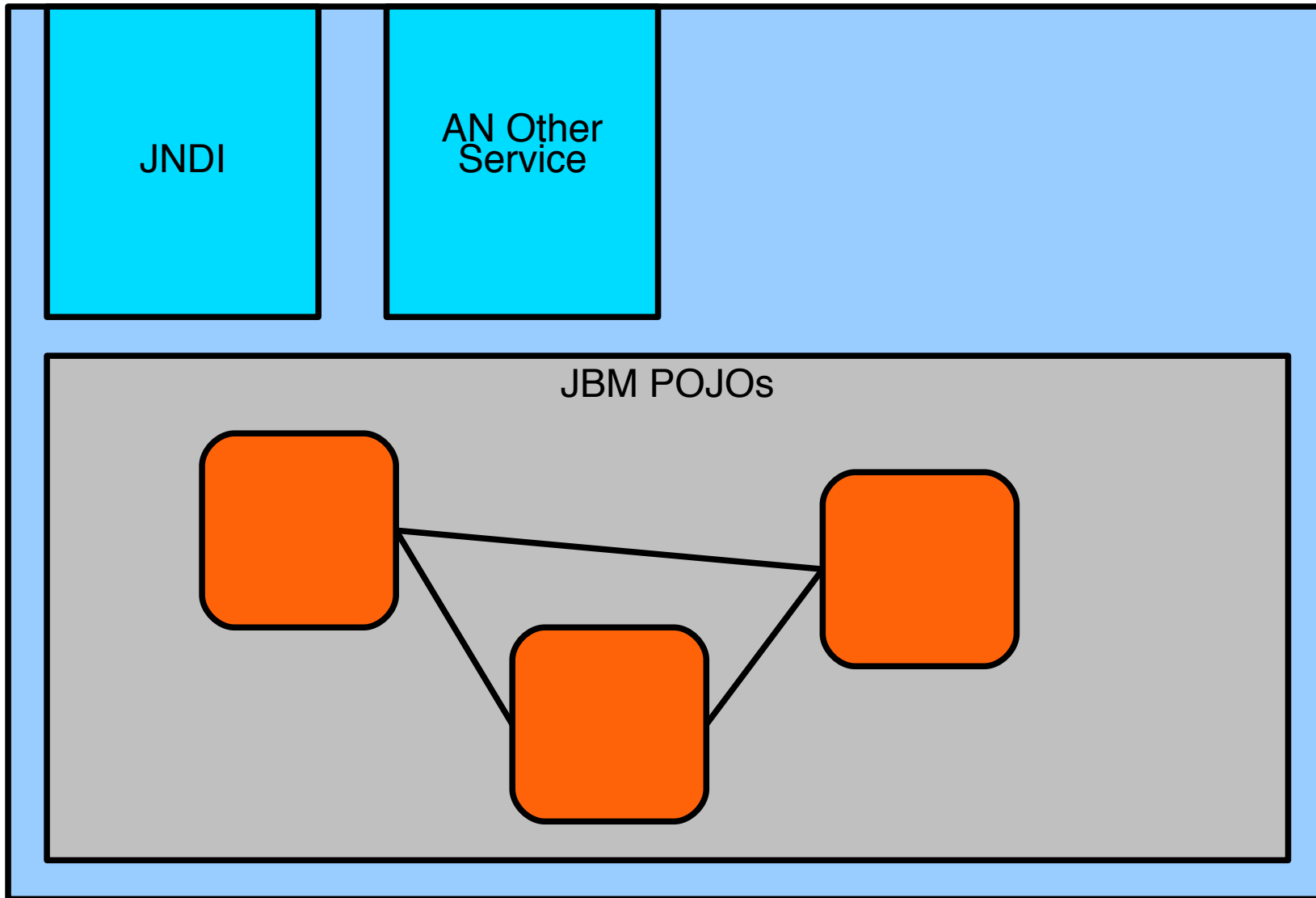
## **Elegant generic architecture**

- Fully functional JMS agnostic messaging system
- No dependencies on JMX, JNDI, JCA, etc.
- Just a set of simple POJOs
- JMS functionality applied as thin facade on the client side
- Superset of JMS – supports asynchronous send acknowledgements, has its own filter language etc.
- Can be embedded in an application that requires messaging internally.

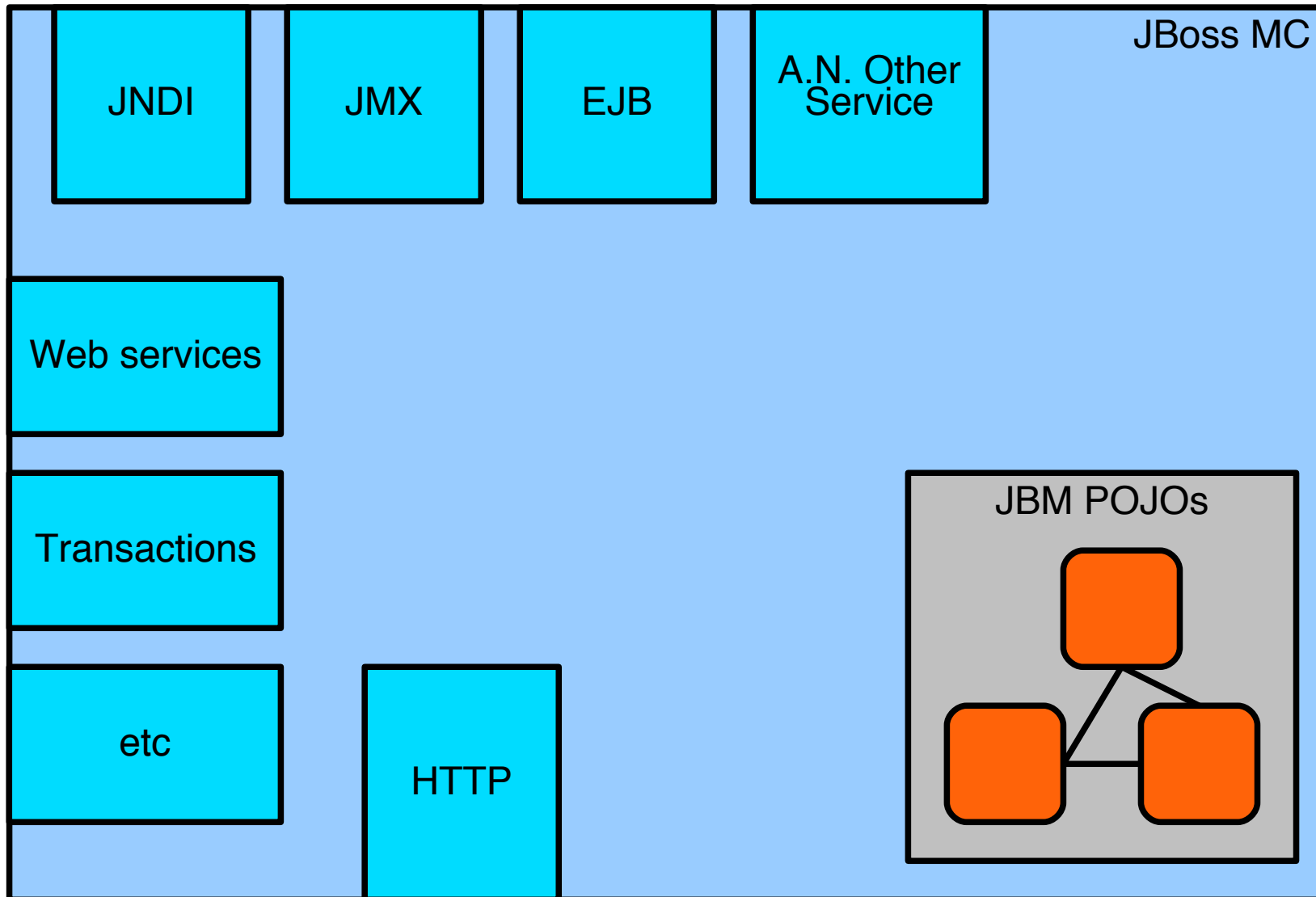
# Generic core and JMS facade



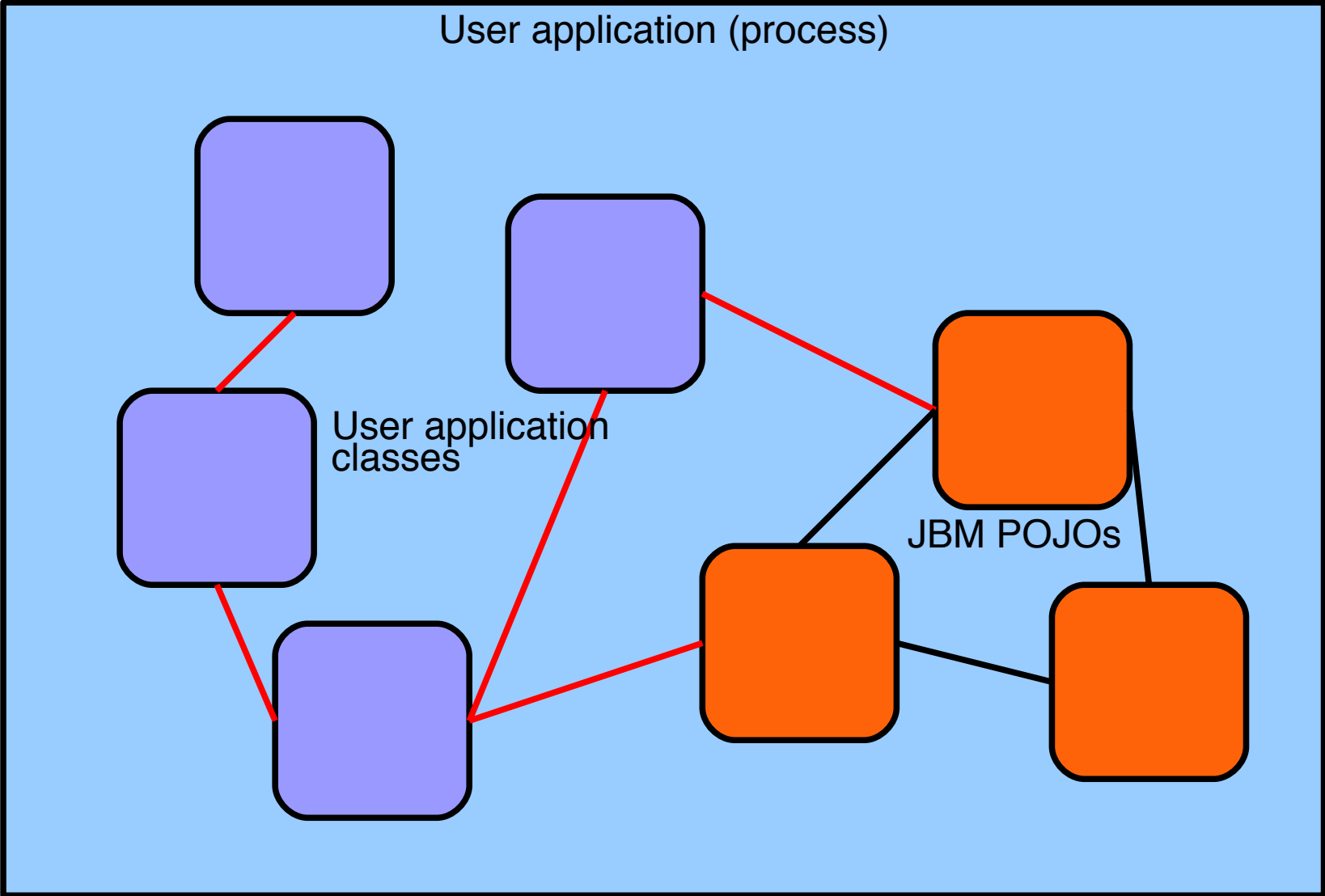
# JBM stand-alone deployment using JBoss Micro-container



# JBM inside JBoss AS 5.0



# JBM embedded in 3<sup>rd</sup> party application



# **JBM embedded pseudo-code example**

```
MessagingServer server = new MessagingServerImpl();
```

```
ConnectionFactory cf = new ConnectionFactoryImpl();
```

```
Connection conn = cf.createConnection();
```

```
Session sess = conn.createSession(...);
```

```
Message message = new MessageImpl();
```

```
Consumer cons = session.createConsumer("Queue1");
```

```
sess.send(message);
```

```
Message received = cons.receive();
```

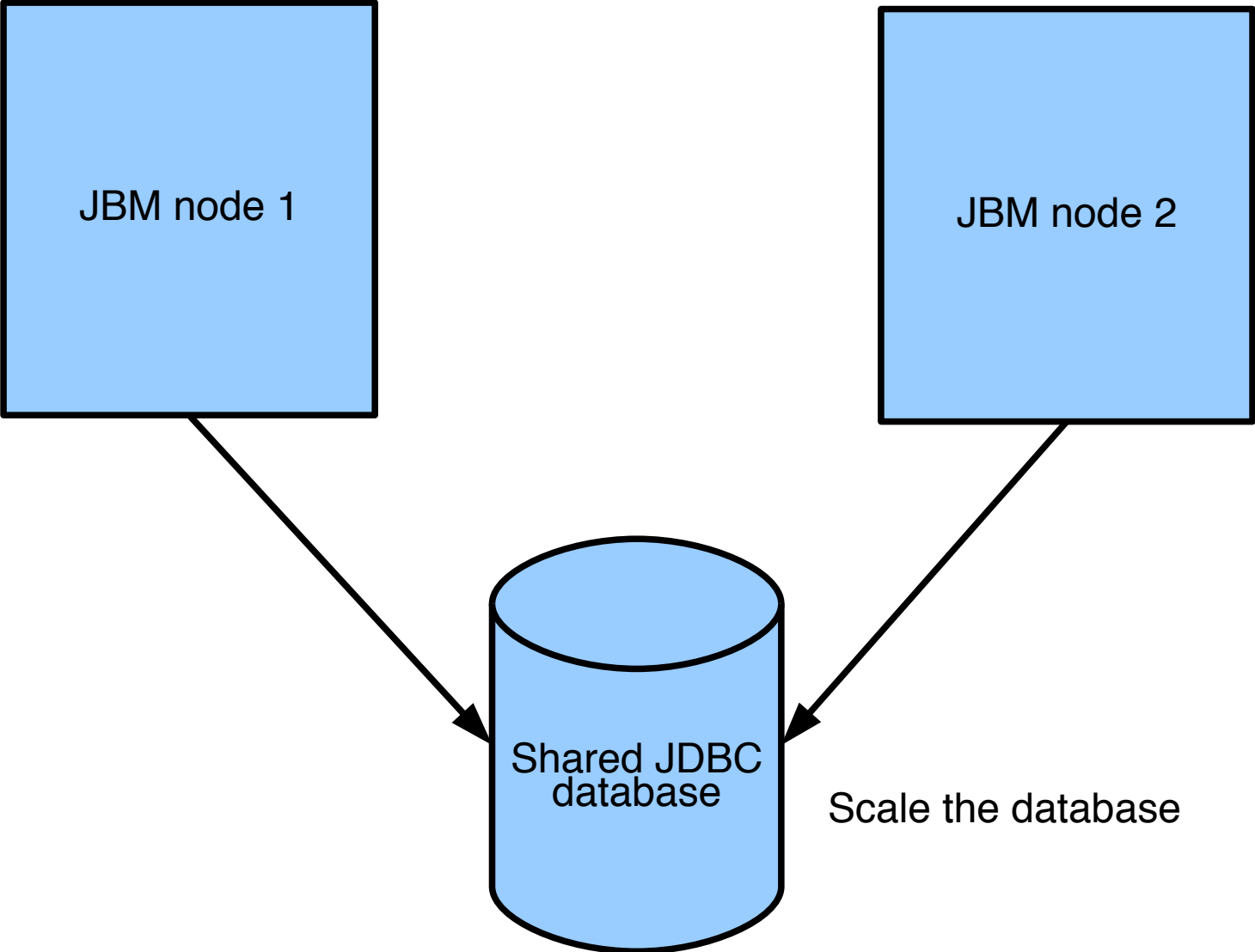
## **JBM 2.0 persistence options**

- JDBC shared database
- JDBC database per node
- Java NIO Journal per node
- Linux Asynchronous-IO store per node
- Null persistence

## **JDBC shared database**

- Single database instance is visible by all nodes.
- Contention point – need to scale the database to scale the system if the system uses persistent messages heavily.
- Will use Hibernate for persistence – will work with any database Hibernate works with

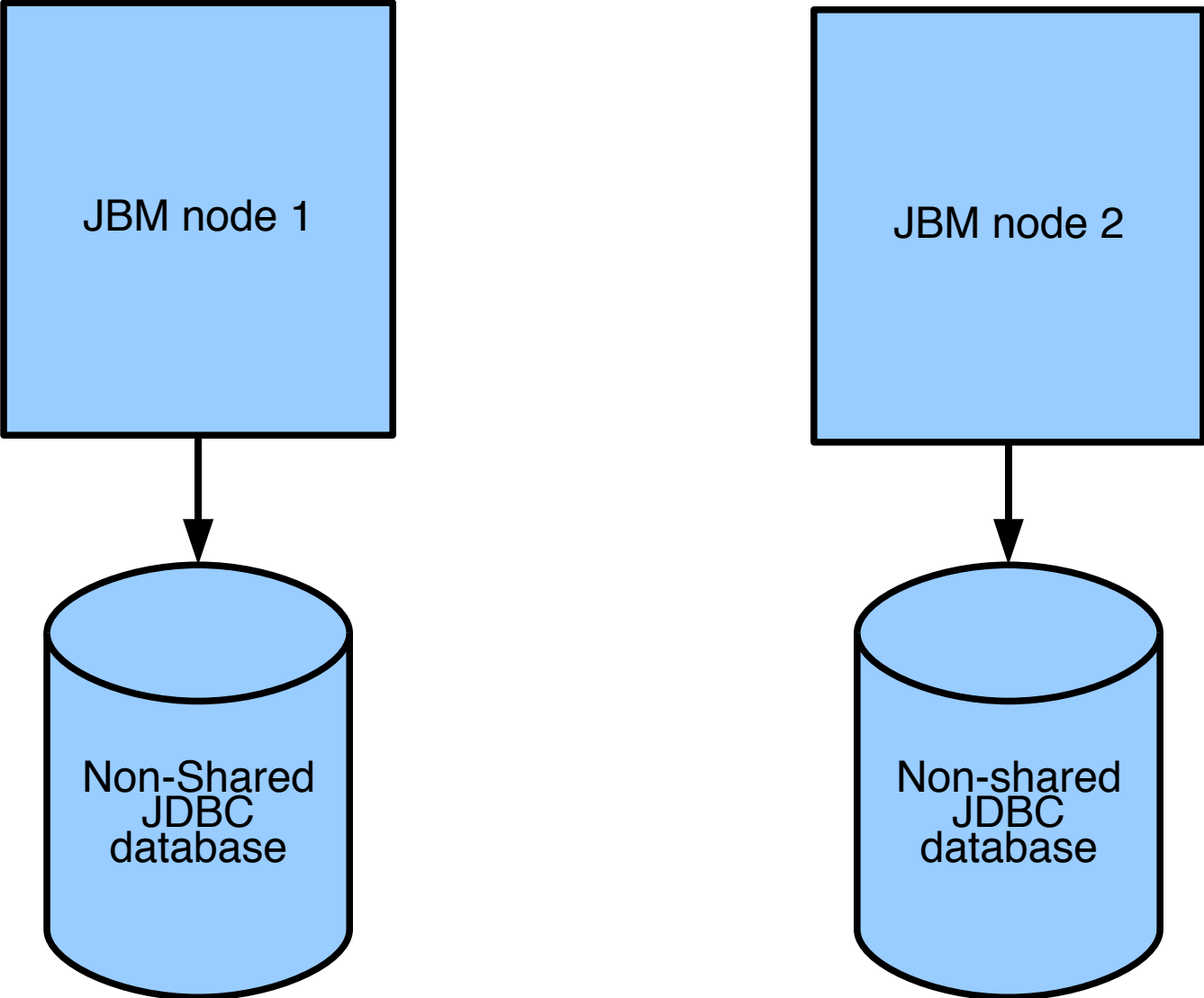
# JDBC shared database



## **JDBC database per node**

- Each node has its own database instance (either local or remote)
- Easier to scale than shared database
- Will use Hibernate for persistence – so will work with any database supported by Hibernate

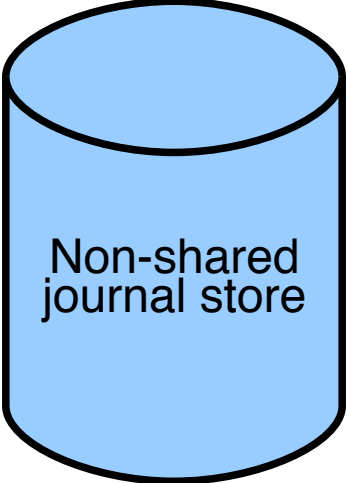
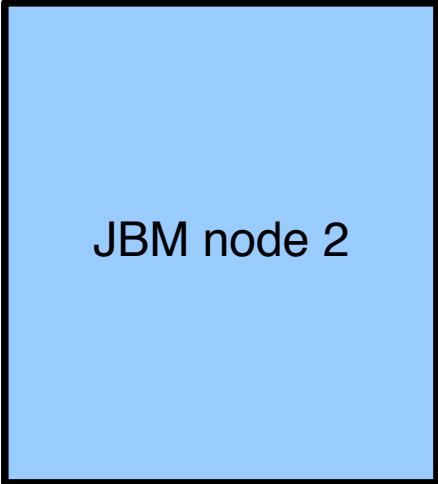
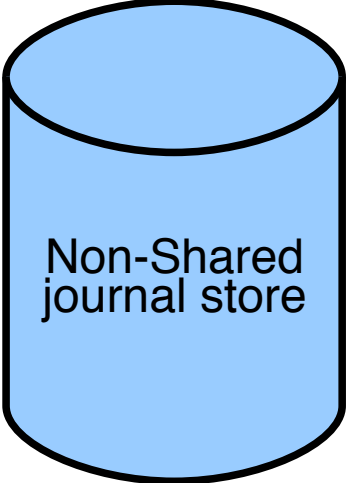
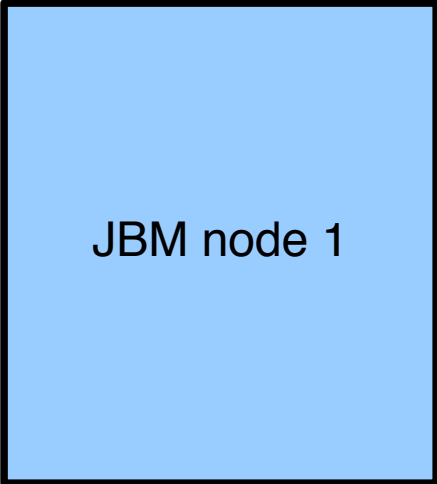
# JDBC non-shared database



## **Java NIO journal store**

- Very fast storage that uses an append only journal approach. This minimises disk seeks, since aim to keep entire log file on single cylinder of disk.
- 100% Java – uses Java NIO.
- Each node has its own store
- Store can be local (on same box), or on a shared file system e.g. GFS on a SAN.

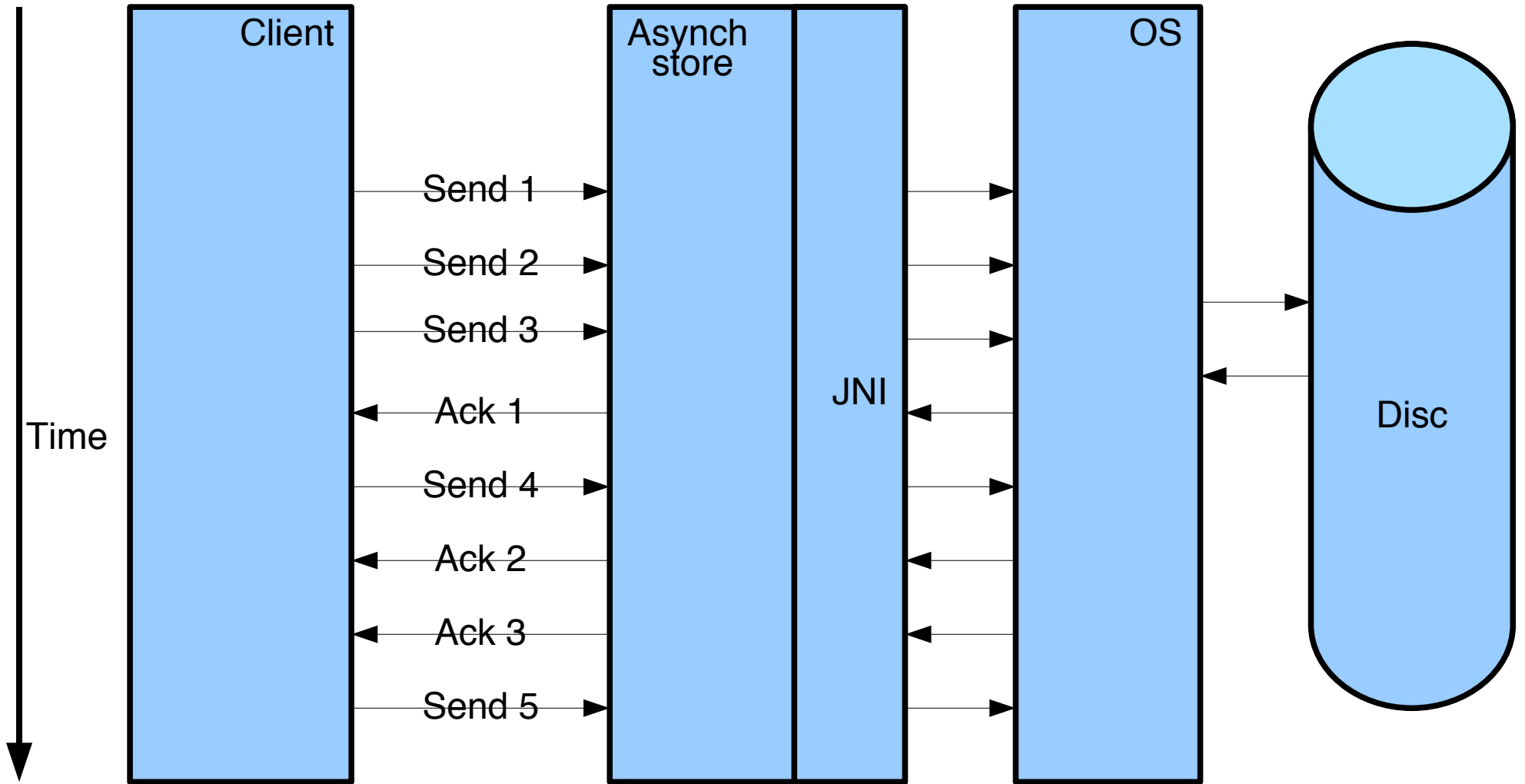
# Journal Persistence



## **Asynchronous IO journal store**

- Very fast journal store using Linux asynchronous IO support.
- JNI interface to aio library, encapsulated in Java package.
- Majority of store written will be written in Java so can be easily extended to work with other OS's (just need to change the thin C++ layer)
- Message sends do not block until message is stored
- Acknowledgement of persistence is received later in a separate stream.
- Throughput not limited by latency of network.
- Will be available using JMS API
- Can easily saturate disk write throughput

# Asynch-IO Store



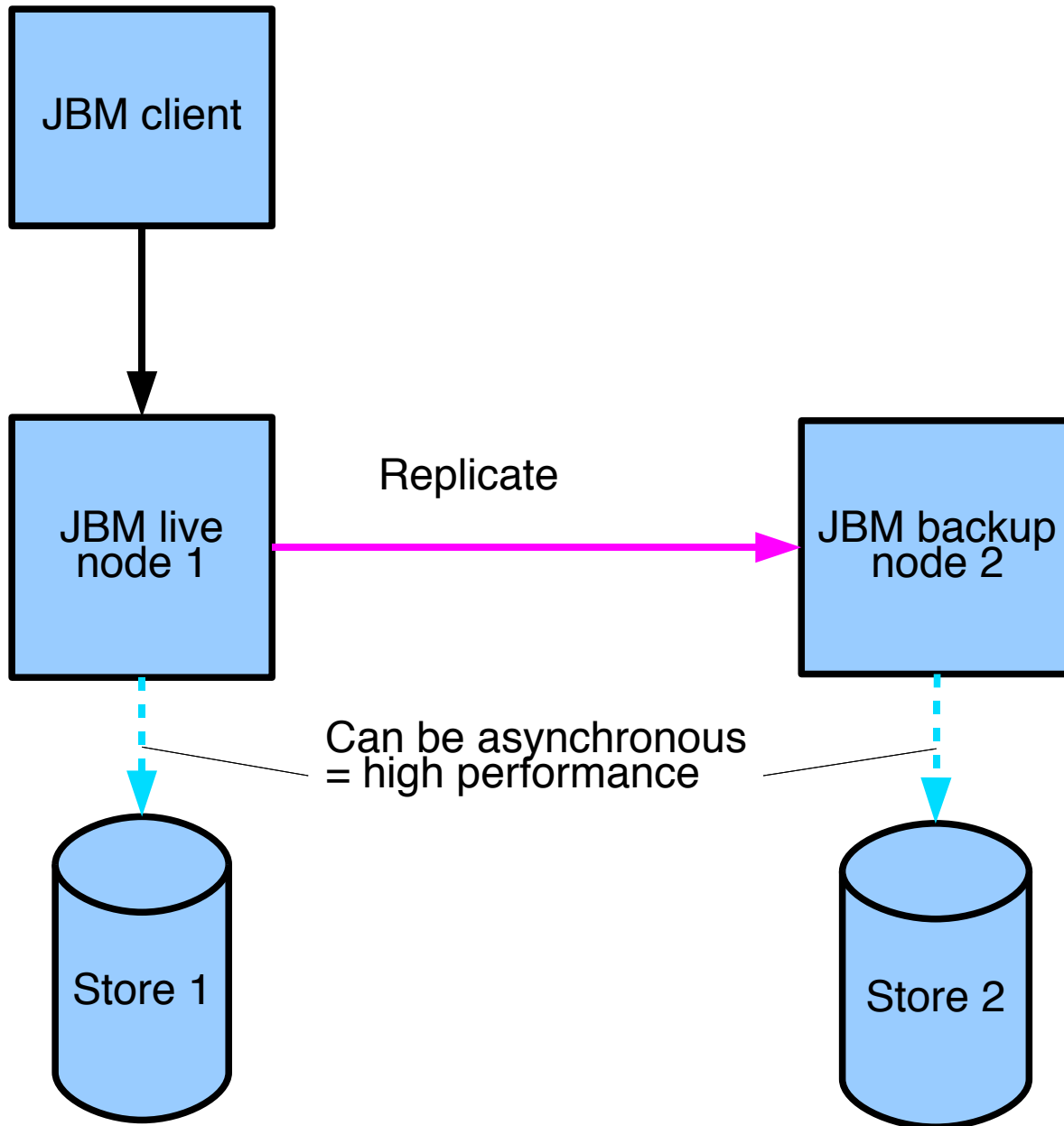
## **Extended high availability support**

- For non shared store (journal or JDBC) – if we want hot HA, then we need to replicate the data. This is a “shared nothing” approach in replication terminology.
- For shared store, we can also replicate, but may choose to fail-over via the shared store.

## **Replication approach (shared nothing)**

- Each live node in the cluster is twinned with a non-live backup node.
- A non-live node does not service any client requests unless its master fails.
- As work is done on a live node, it is replicated synchronously to the backup.
- On event of live node failure, the backup already has the state to carry on where the live node left off. ==> Fail-over is very quick and transparent.
- Can actually be even faster than a single node – since don't need to persist synchronously on live and backup!

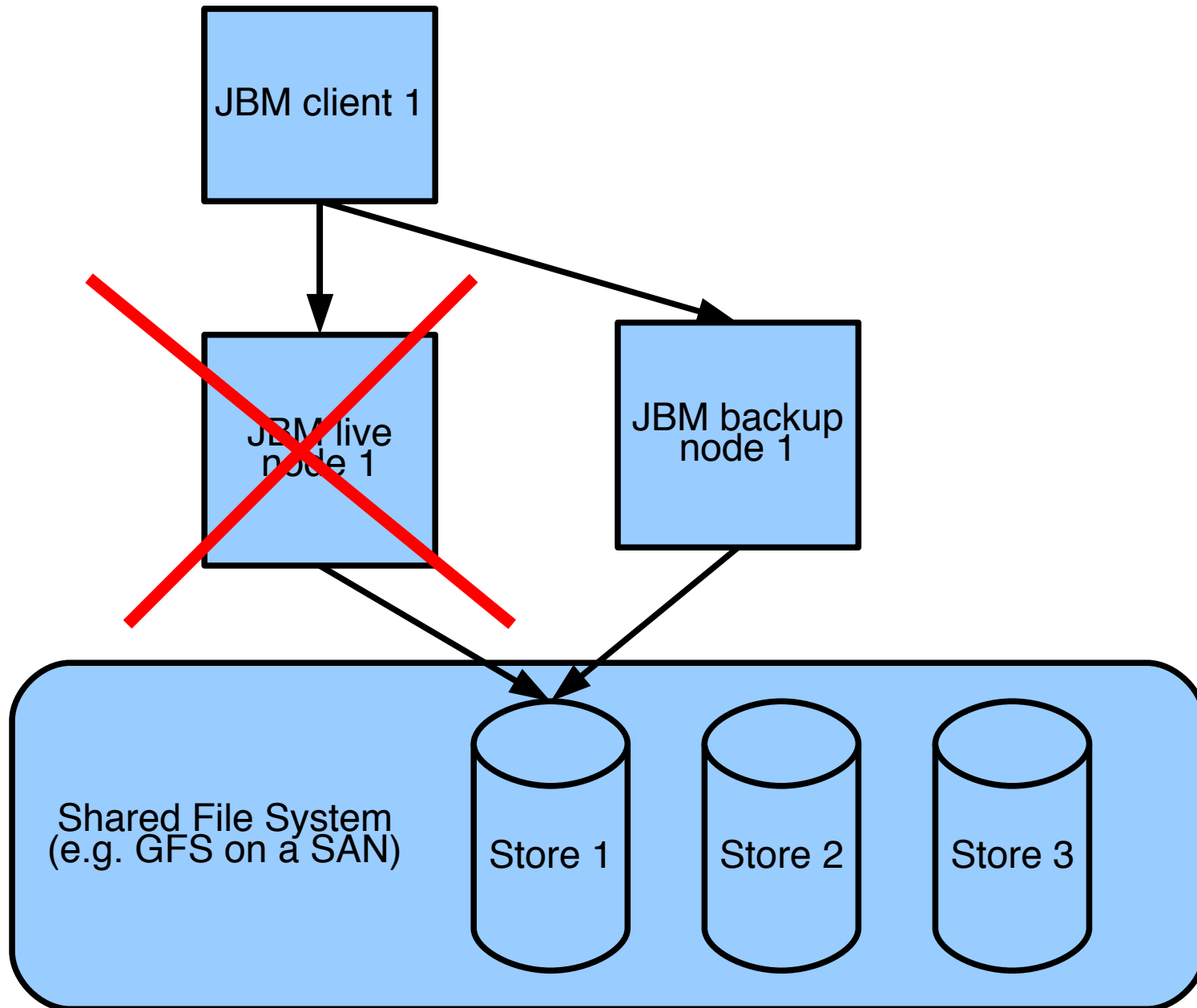
# Replication for HA - Shared nothing



## **Fail-over – shared storage area**

- Each node can have its own store
- Each store is persisted on a shared file system which is accessible by each node in the cluster e.g. GFS over SAN.
- No replication is performed between nodes.
- Typically fail-over slower – since need to load queue state into backup on fail-over

# Fail-over using shared storage



# **Brand new high performance transport**

- Brand new transport using Apache MINA.
- Trustin Lee (MINA project lead) is now JBoss employee
- Scales to many thousands of concurrent connections.
- Support for TCP, SSL, HTTP and Apache APR.
- Can easily saturate a gigabit LAN.

# New security configuration

- Highly flexible declarative security for queues. Supports wild-cards. Queues created on the fly according to permissions.

```
<security match="*">
  <permission type="create" roles="admin"/>
  <permission type="read" roles="admin"/>
  <permission type="write" roles="admin"/>
</security>

<security match="queues.userqueues.*">
  <permission type="create" roles="admin, guest"/>
  <permission type="read" roles="admin, guest"/>
  <permission type="write" roles="admin, guest"/>
</security>
```

# New queue configuration

- Can use wild-card matching to apply sets of properties to queues

```
<queue-settings match="">
```

```
  <clustered>>false</clustered>
```

```
  <dlq>DLQ</dlq>
```

```
  <expiry-queue>ExpiryQueue</expiry-queue>
```

```
  <redelivery-delay>0</redelivery-delay>
```

```
  <max-size>-1</max-size>
```

```
</queue-settings>
```

```
<queue-settings match="queues.publicqueues.*">
```

```
  <max-size>10000</max-size>
```

```
</queue-settings>
```

## **Beyond JBM 2.0**

- Native support for non-Java clients. C++, .NET, etc. (note we have users using StompConnect with JBM currently)
- AMQP – Messaging specification being worked on by other Red Hat groups. The specification is currently quite immature. If the specification matures and we see sufficient demand for it then it is likely we will implement it.
- Running JBM on real-time Linux and real time Java, for guaranteed latency.

## **Road map (subject to change)**

- JBM 2.0 alpha (non-clustered) – a few weeks time
- JBM 2.0 beta (full features) – end of Q2 2008
- JBM 2.0 GA – Q4 2008

## **The future looks bright!**

- JBoss Messaging is the future of messaging at JBoss. JBM is the JMS provider in all JBoss platforms.
- JBM 2.0 will offer un-rivalled levels of performance and scalability as well as supporting an extended set of persistence, transport and HA configurations.
- JBM 2.0 will run as a standalone messaging server, or integrated in JBoss AS and SOA platform, and will be runnable embedded in a third party application.
- Our goal is for JBoss Messaging to be the premier open source messaging solution.

**Thanks for coming!**

**Any questions?**

**<http://jbossfox.blogspot.com>**