

JBoss Messaging – present and future

Tim Fox
Messaging Lead, JBoss

Agenda

- What is JBoss Messaging?
- Is it supported?
- JBM 1.4 features
- JBM 2.0 – the next generation
 - In Jboss MC and embedded
 - Extended persistence support
 - Extended high availability (HA)
 - New transport
 - Improved configuration
- Road-map

What is JBoss Messaging?

- Generic reliable, well featured, asynchronous messaging system
- Fully compliant JMS 1.1 implementation
- Fully JEE 5 compliant
- Many features over and above JMS

Where is JBoss Messaging?

- 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 including investment banks.

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.

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
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
- Transparent fail-over
- Message redistribution
- Clustered temporary destinations
- 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, SQLServer

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 Q3/Q4 2008
- JBM 2.0 goal – To be the most performant, most full featured clustered reliable messaging implementation.
- Will produce performance comparisons against our competitors e.g. ActiveMQ
- A generic embeddable messaging implementation

What's new in JBM 2.0?

- Completely JMS independent generic core. (more later)
- Bootstrappable in JBoss Microcontainer or any other dependency injection framework.
- Can be embedded in third party applications – OEMs.
- Support for very fast local append only journalling store using Berkeley DB JE.
- Improved JDBC persistence support using Hibernate – supports any database that Hibernate supports.

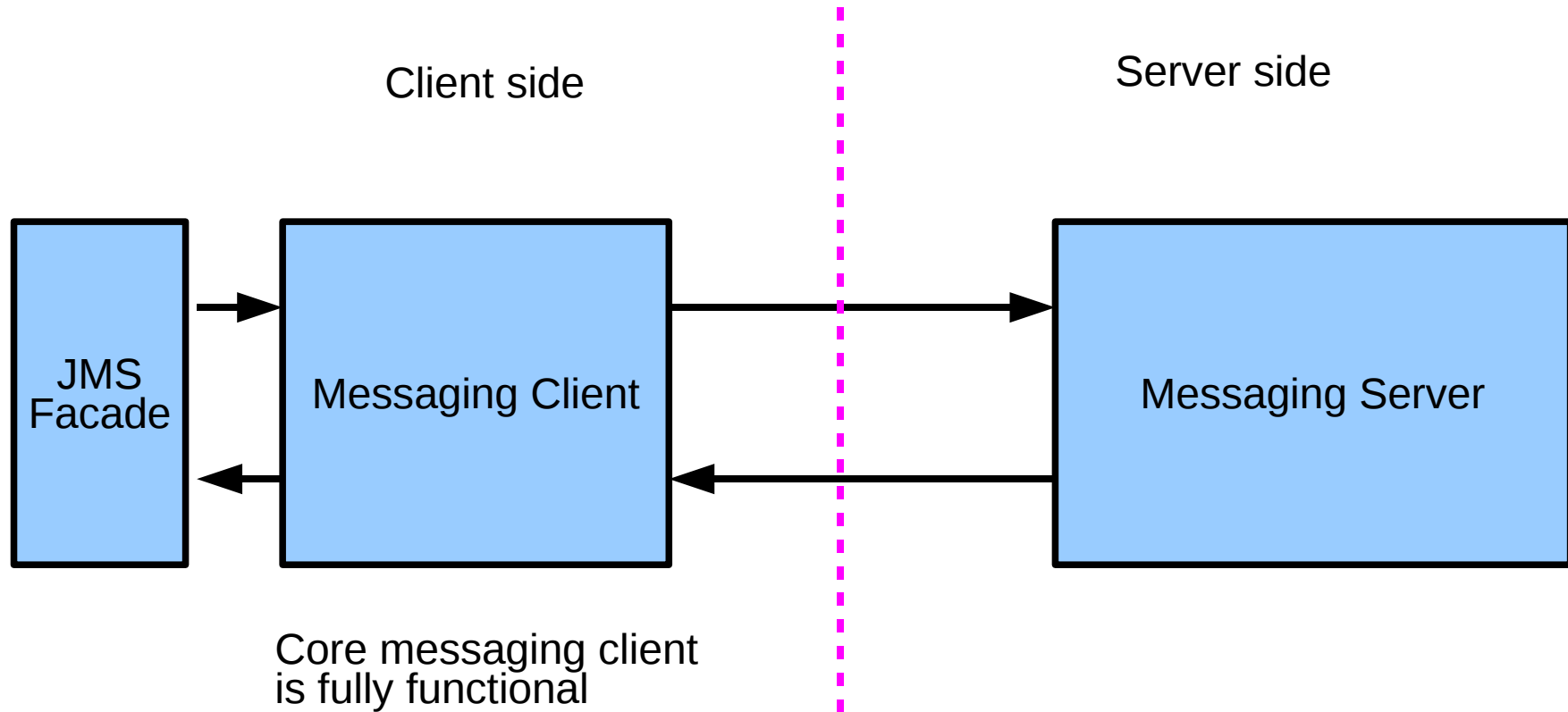
What's new in JBM 2.0 (continued)

- Extended and more flexible HA.
- Brand new NIO transport using Apache MINA. Supports TCP, SSL, HTTP and native APR.
- Improved and more flexible queue configuration and security.
- Producer flow control.
- Topic hierarchies
- Message grouping.
- Many other smaller features

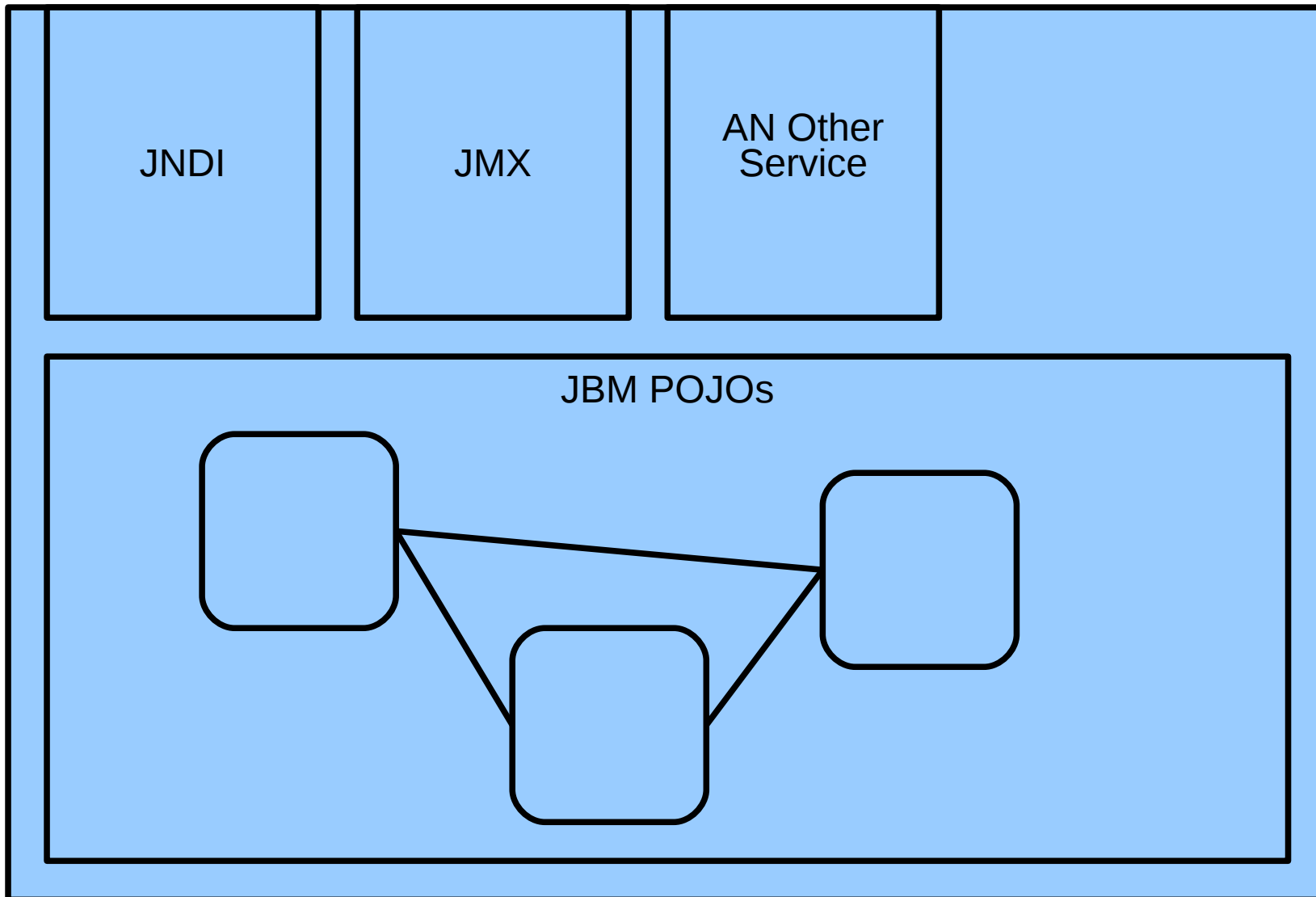
Elegant generic architecture

- 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 amongst other things, has its own filter language etc.
- Can be embedded in an application that requires messaging internally.

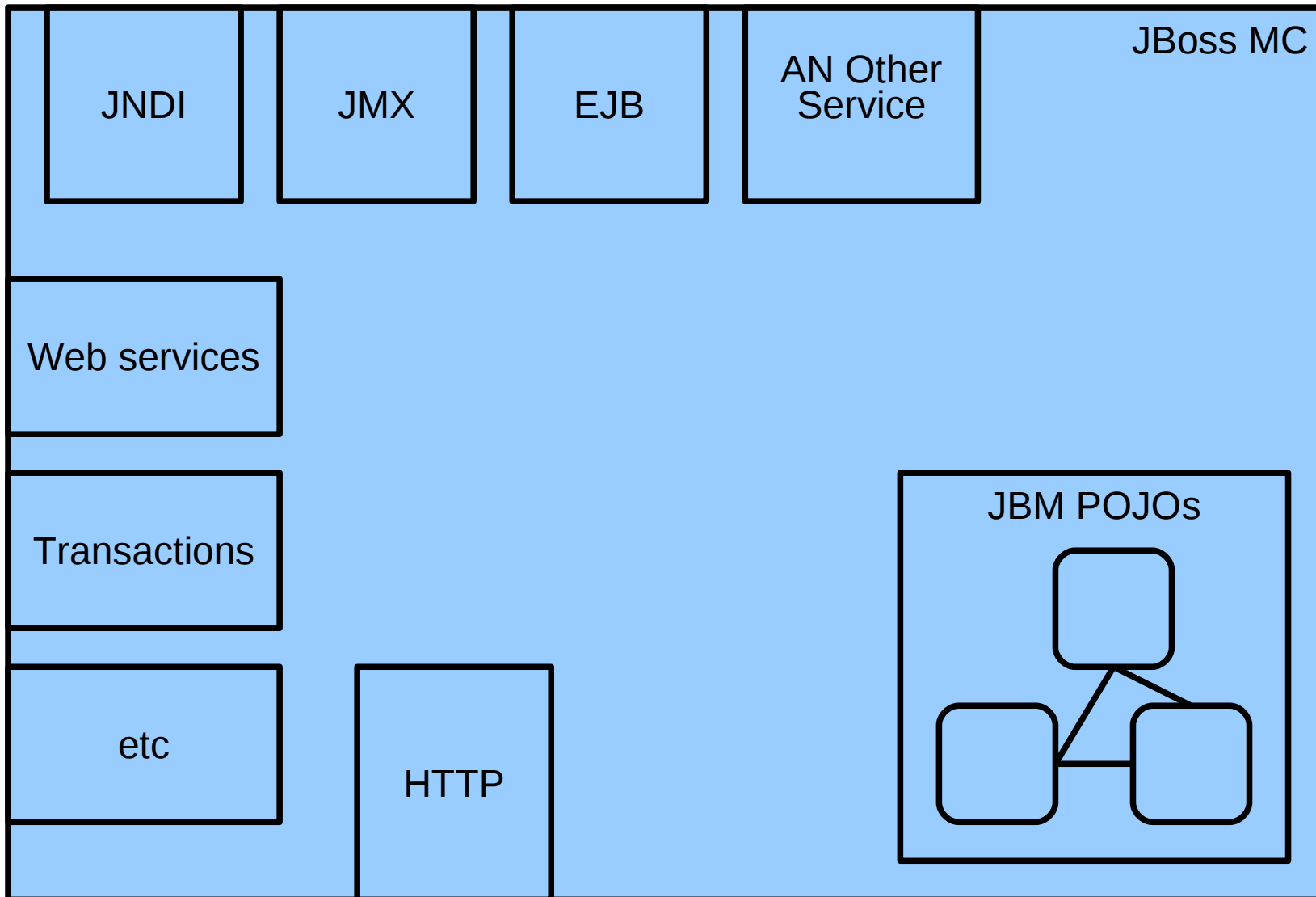
Generic core and JMS facade



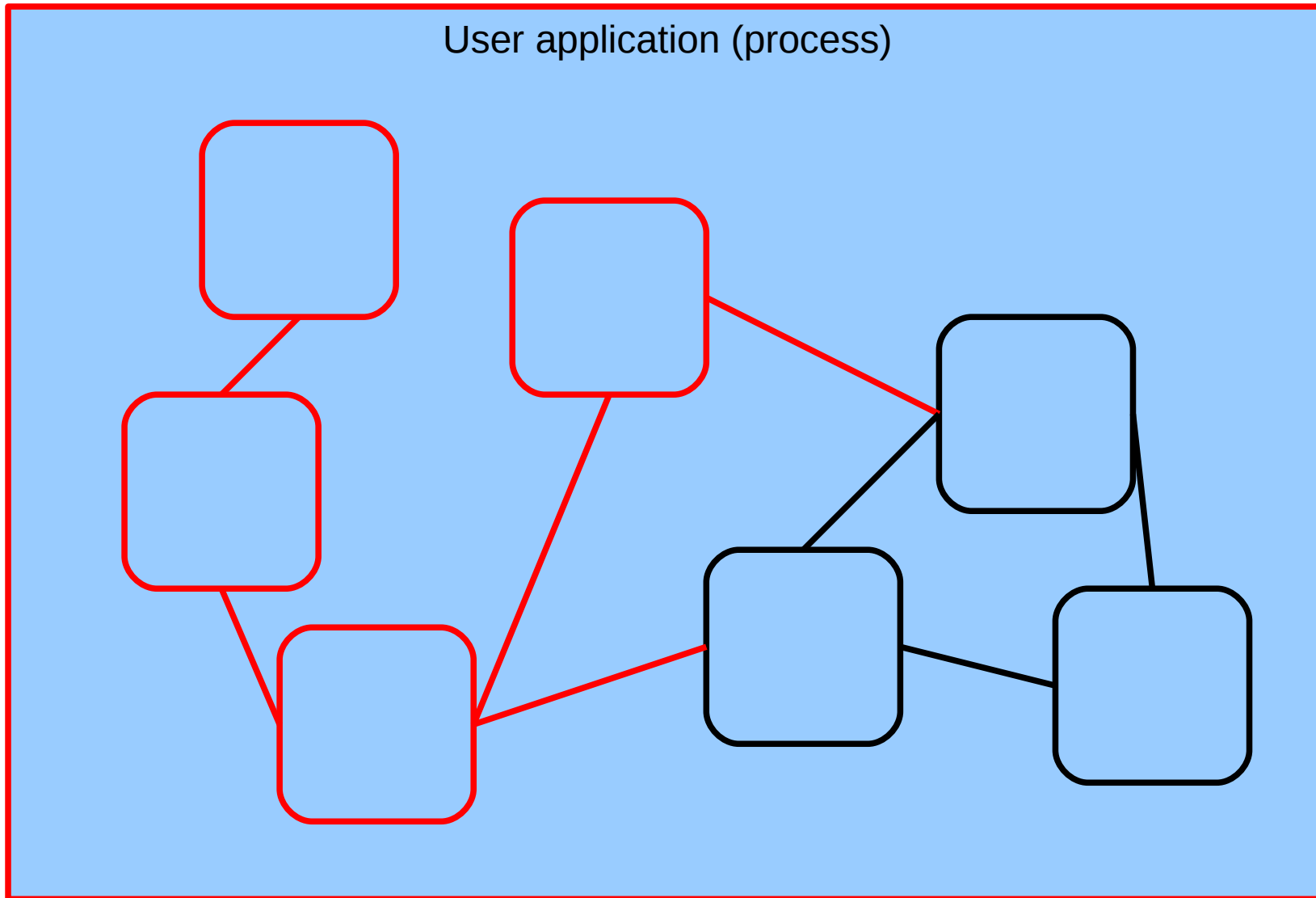
JBM stand-alone configuration using JBoss Micro-container



JBM inside JBoss AS 5.0



JBM embedded in 3rd 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();
```

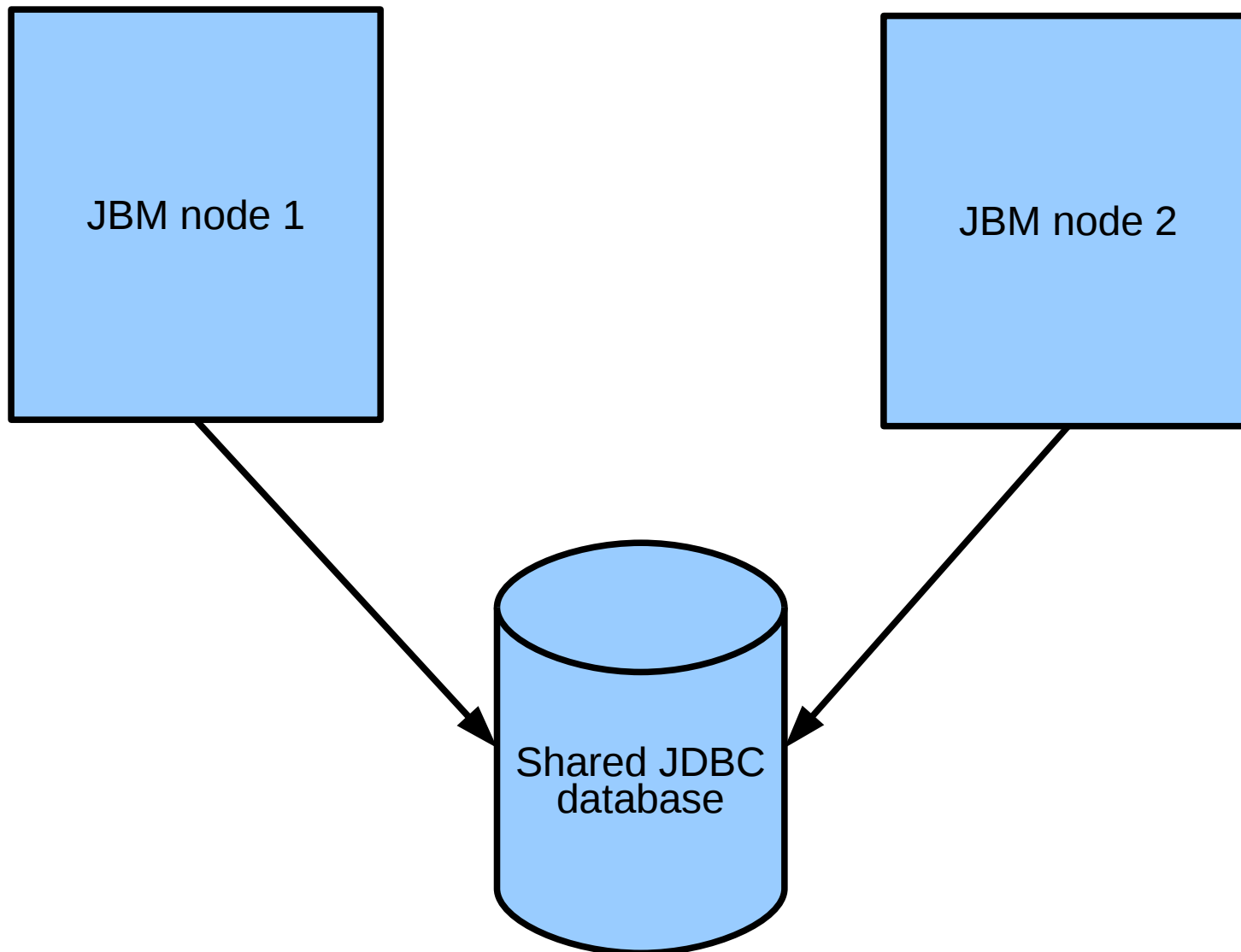
Extended Persistence support

- Classic JDBC shared database
- JDBC local database per node
- BDB file store per node
- Null persistence
- New persistence configurations has implications for HA

JDBC shared database

- Single database instance is visible by all nodes.
- Single point of bottleneck – need to scale the database to scale the system.
- Uses Hibernate for persistence – so works with any database
Hibernate works with
- Not as fast as journal based persistence
- Approach used for JBM 1.x

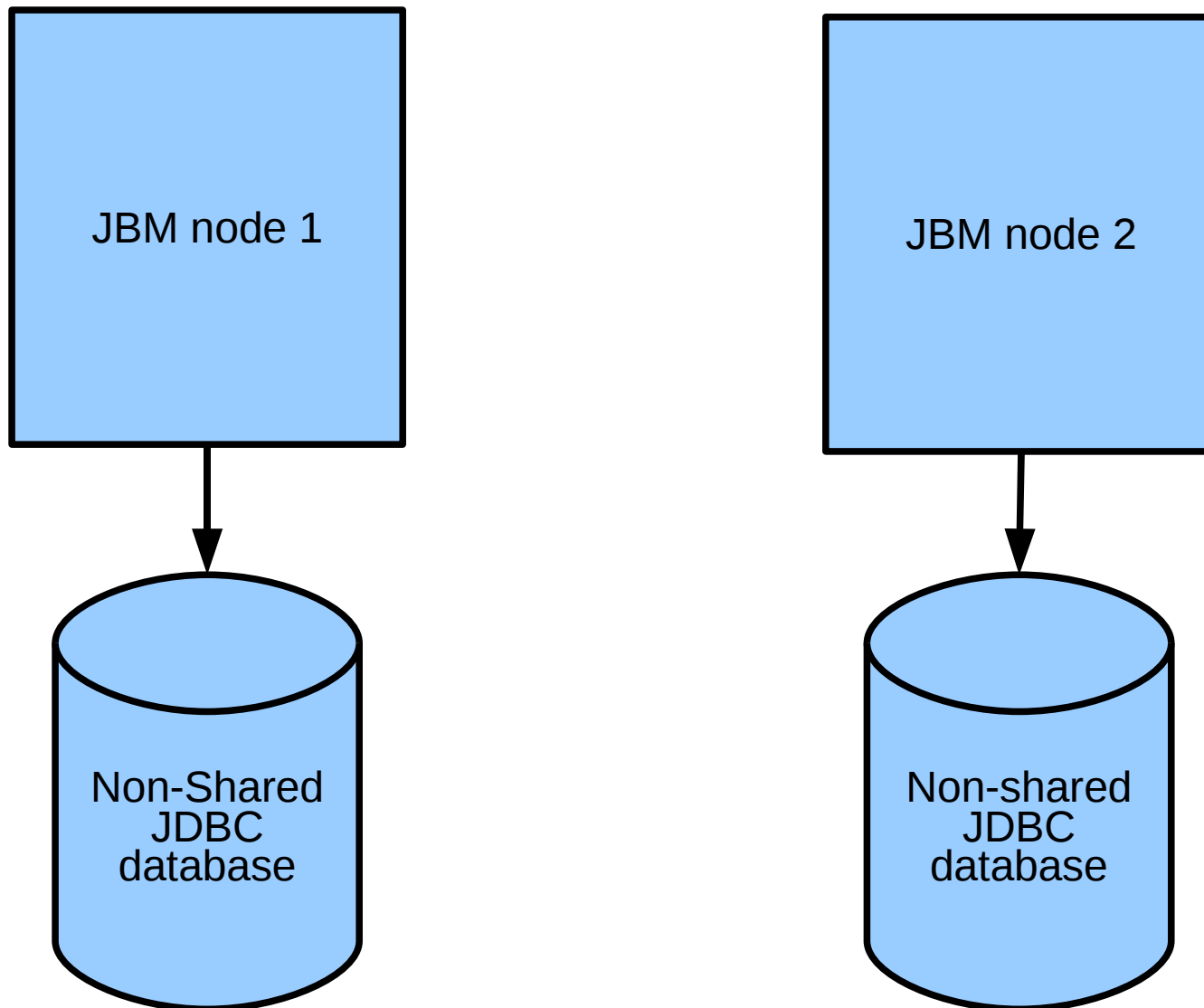
JDBC shared database



JDBC non shared database

- Each node has their own database instance (either local or remote)
- Better scalability then shared database
- Uses Hibernate for persistence – so works with any database
Hibernate works with
- Not as fast as journal based persistence

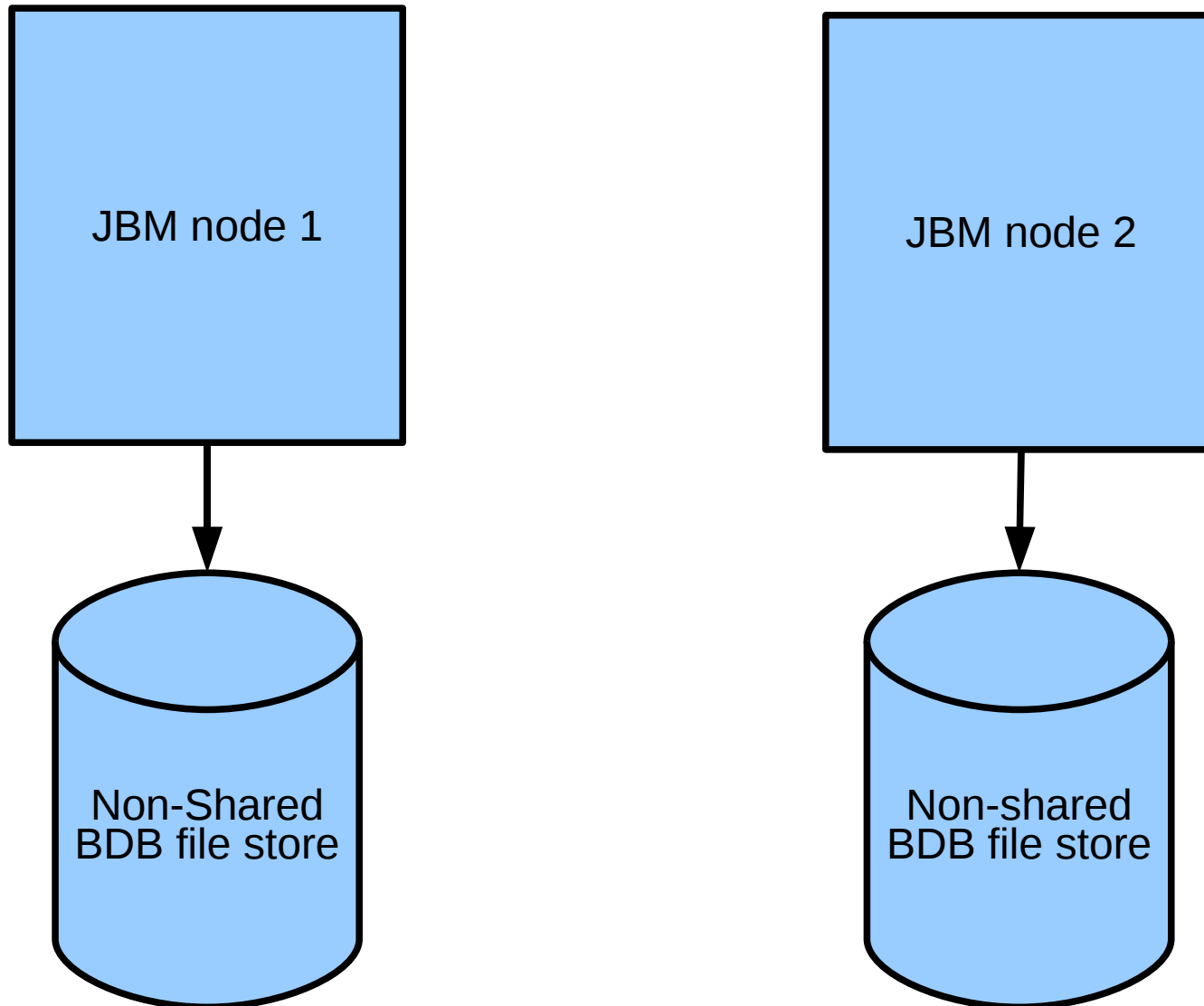
JDBC non shared database



Berkeley DB JE local database

- Very fast persistence manager that uses an append only journal approach. This minimises disk seeks, since aim to keep entire log file on single cylinder of disk.
- 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.

BDB JE Persistence



Extended high availability support

- For non shared store (BDB 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.

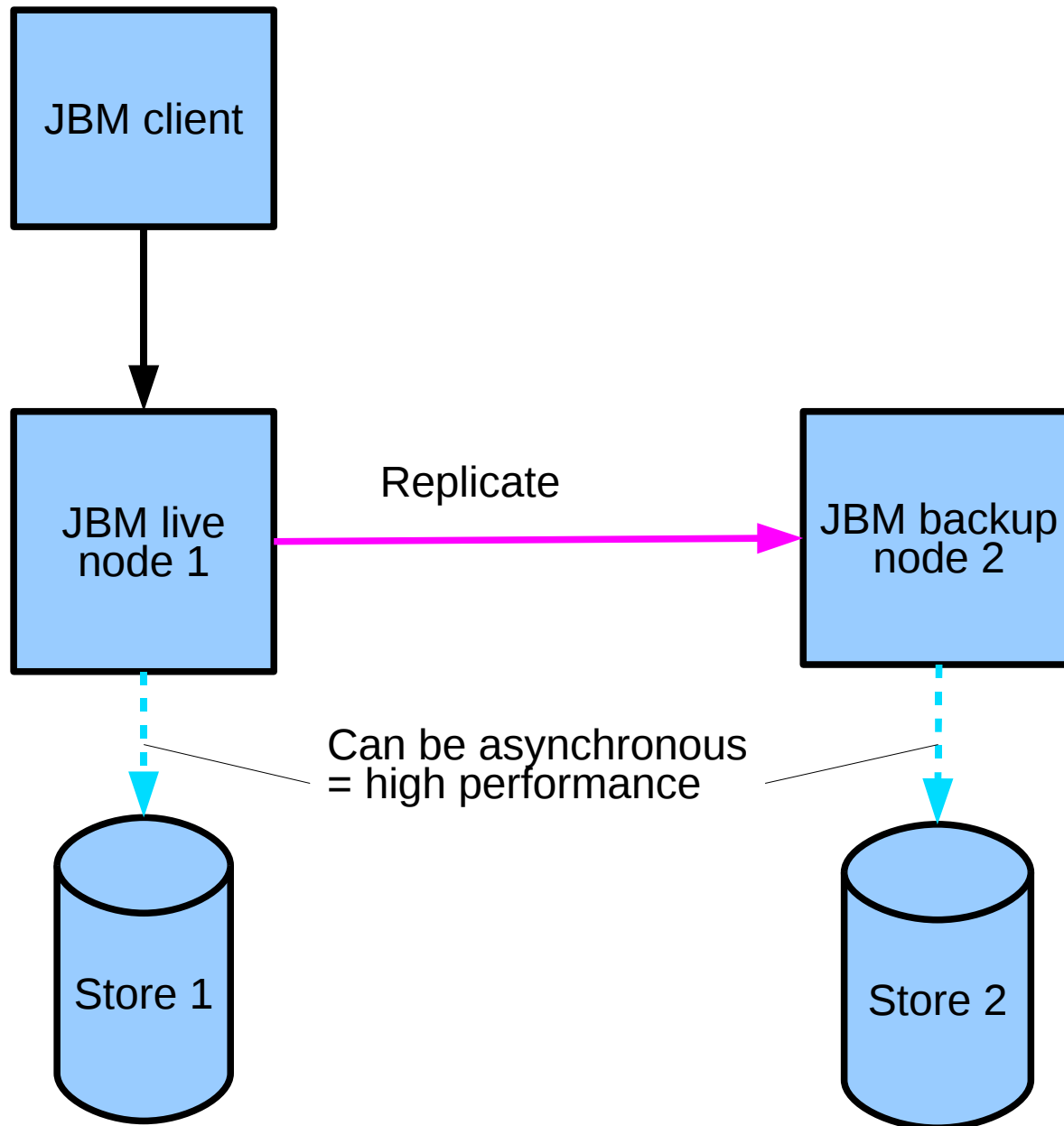
Replication approach (shared nothing)

- Each live node in the cluster is twinned with a non-live backup node.
- The non-live nodes do not service any client requests unless it's 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 approach (continued)

- When master fails, new backup can be brought up and backup can be promoted to master.

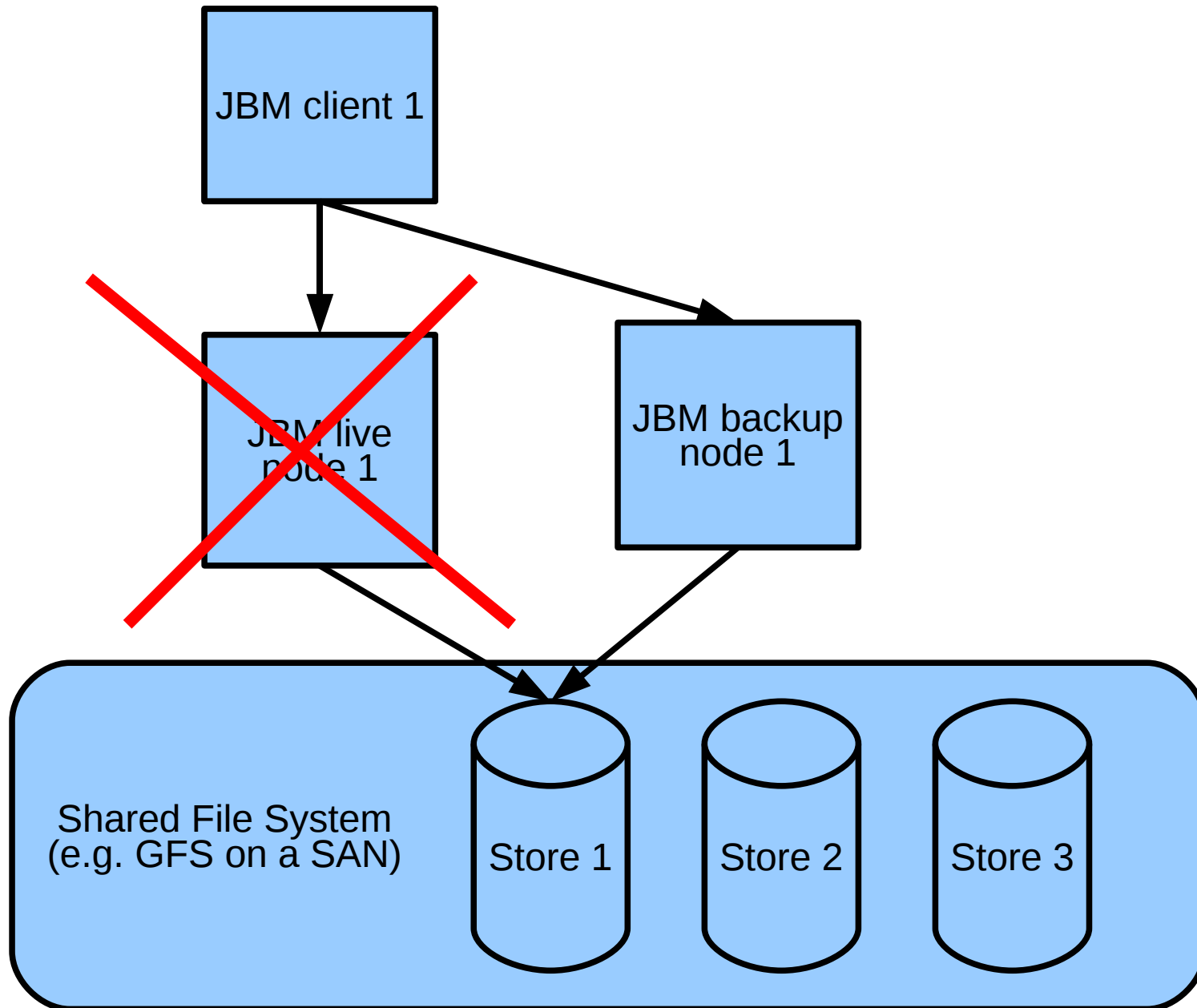
Replication for HA – Shared nothing



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
- Typically GFS over SAN on fibre channel (or other high speed layer) for high performance.
- No replication is performed between nodes.
- If asynchronous persistence on shared nothing this can be used for highest performance with guaranteed persistence to disk.
- 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 lead) is now JBoss employee
- Scales to many thousands of concurrent connections.
- Support for TCP, SSL, HTTP and Apache APR.

New security configuration

- Highly flexible declarative security for queues. Supports wildcards. 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 wildcard 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>
```

JBM – the future

- Support for non Java clients. C++, .NET etc
- AMQP – Messaging specification being worked on by other Red Hat groups. The specification is currently immature (beta) and is currently not rich enough to support even JMS functionality or other basic messaging functionality (no selectors, security etc). When it matures (reaches 1.0) then it is likely we will implement it, but it is shifting too much now.

Roadmap (provisional)

- JBM 2.0 alpha (non clustered) – end of Q1 2008
- JBM 2.0 beta (full features) – end of Q2 2008
- JBM 2.0 GA – end of Q3/Q4 2008

The future looks bright!

- JBM 1.4 is currently available supported.
- 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 can be run as a standalone messaging server, can be run integrated in JBoss AS and SOA platform, and can be run 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?