



JetStreamQ

Data Sheet
(Getting Started)

JetStreamQ

message broker for Java



- * Simplicity and productivity are most important
- * Reliable - and fast - messaging with no setup required
- * Supports JMS Pub/Sub (Topic) and Point-to-point (Queue) based messaging with persistence and 100% reliability
- * High-availability clustering is handled automatically, out of the box, without any configuration required

 JetStreamQ

Installation?

- * Designed to be as simple as it is reliable, JetStreamQ consists of just *one* JAR file
- * There is no installation required
- * Simply import JetStreamQ's JAR file and the message broker runs embedded within your application
- * Optionally, you can run JetStreamQ as a standalone process with the scripts included (for Windows, Linux, Solaris, and Mac OS X)

Configuration?

- * JetStreamQ requires no configuration
- * No configuration files or complicated administrative interface
- * All running JetStreamQ instances auto-discover each other and then self-configure
- * Each JetStreamQ node works as part of a cluster, again without any time-consuming configuration required

JNDI Server

* Initial Context Factory URL:

* `com.allure.JetStream.jndi.JetStreamInitialContextFactory`

* Sample code:

```
Hashtable env = new Hashtable();

env.put("java.naming.factory.initial",
        "com.allure.JetStream.jndi.JetStreamInitialContextFactory");

Context jndi = new InitialContext(env);

ConnectionFactory conFactory =
    (ConnectionFactory)jndi.lookup("

// Create a JMS connection
connection = conFactory.createConnection("user", "password");
session = connection.createSession(
    false, Session.AUTO_ACKNOWLEDGE);

topic = session.createTopic( topicName );

producer = session.createProducer(topic);
```

Message Persistence

- * Persisted messages are saved to an internal database
- * By default, the database journal files are stored in the current user's home directory
 - * For example: `~/jsjournal` on Unix
- * The default location can be changed via command-line:
 - * `java -DJetStream.persistfolder=/myfolder ...`

Automatic Failover



- * One instance per computer runs as the “server”
- * Other instances automatically serve as “stand-by”
- * If the server instance shuts down for any reason, one of the stand-by instances takes over seamlessly
- * All persisted messages are preserved
(must be the same user account or specified directory)
- * Reliability is guaranteed



Automatic High-Availability

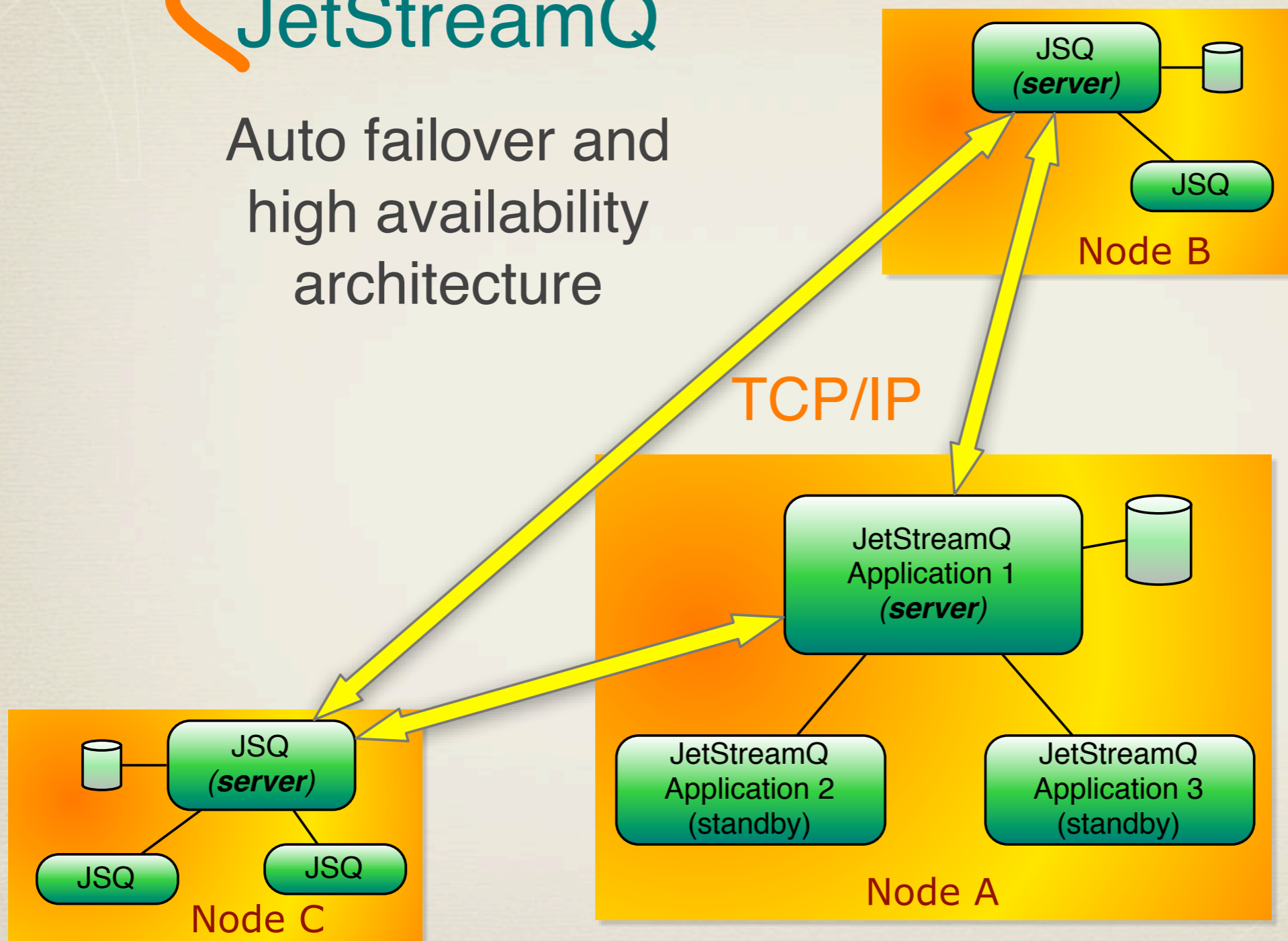


- * Across computers, persisted messages are replicated automatically (note: this can be disabled)
- * All computers running JetStreamQ instances are automatically clustered (note: this can be disabled)
- * If one computer fails, one of the remote JetStreamQ instances automatically recovers and delivers its messages
- * No lost or duplicated messages upon failure or recovery



JetStreamQ

Auto failover and high availability architecture

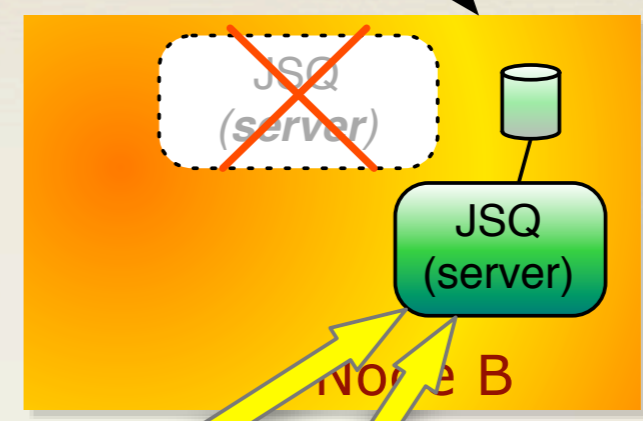


JetStreamQ

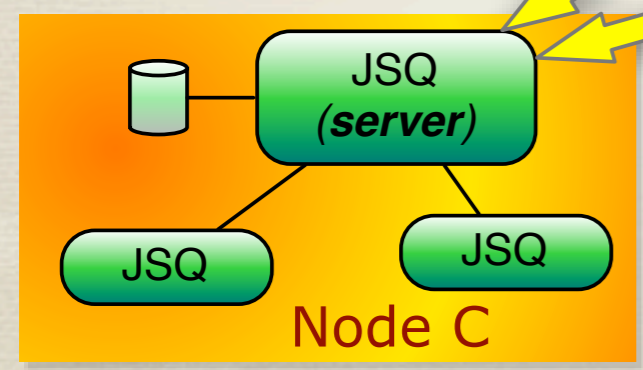
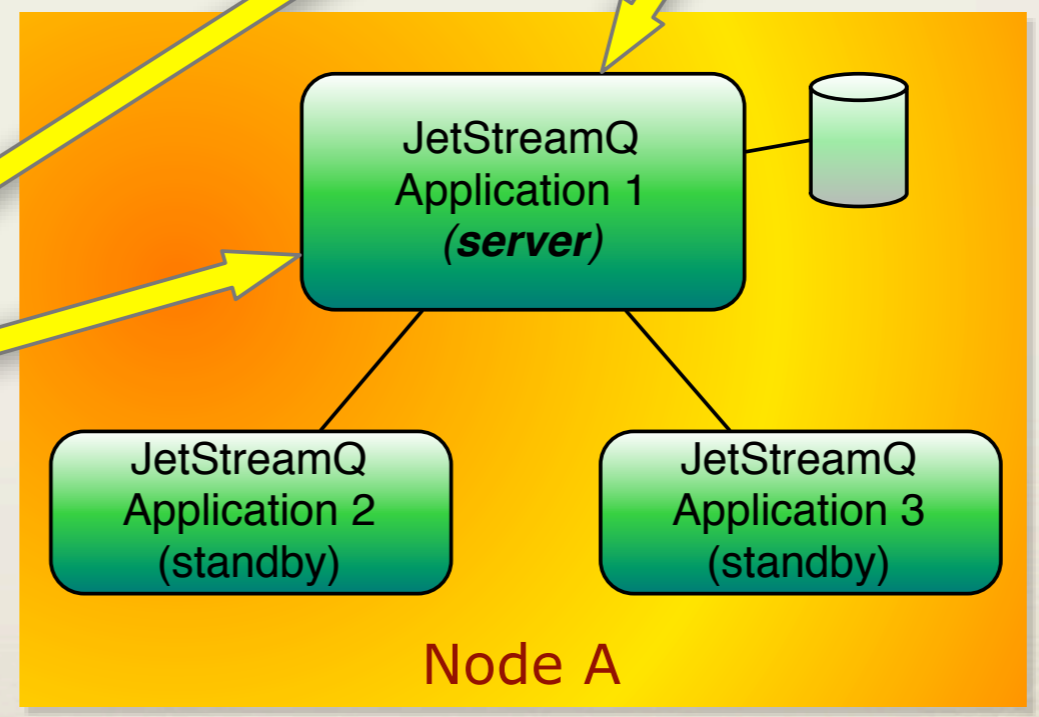
JetStreamQ

Auto failover and high availability architecture

Server instance fails
Standby instance recovers



TCP/IP



JetStreamQ

Scalability

- * Designed to scale to multi-core/processors
- * Designed to never starve a single queue or topic
 - * Fast senders for one Destination do not starve other Destinations
- * Large messages are automatically compressed
 - * This is tunable

Scalability

- * Designed to work well with modern garbage collectors
- * Low heap overhead even when stressed
- * Major GC events are rare even when 100% utilized
- * Low-memory: senders are halted, receivers continue
- * Measured with CMS collector - small pauses

Tuning Parameters

- * Although tuning is not required, it may be desirable in some cases. The following pages list the available command-line parameters
- * Repeatedly add them to the Java command-line as such:
 - * `java -DJetStream.queue senders=16
-DJetStream.log to stdout=true ...`

Tuning Parameters



(page 1 of 4)

Parameter name	Default	Description
<code>-DJetStream.logtofile</code>	TRUE	By default, all log messages are saved in the file <code>jetstreamX.Y.log</code> . When set to FALSE, log messages aren't saved. The value <i>X</i> is a unique value if multiple instances are started. Value <i>Y</i> is the file count if rollover occurs.
<code>-DJetStream.logtostdout</code>	FALSE	When set to TRUE, internal log messages are replicated to standard out.
<code>-DJetStream.logall</code>	FALSE	When set to TRUE, verbose logging is turned on. WARNING: This option greatly impacts performance
<code>-DJetStream.logappend</code>	TRUE	When set to TRUE, new log messages will be appended to the existing log file upon start up of JetStreamQ.
<code>-DJetStream.logfilesize</code>	10,000,000	The maximum size of the log file before it gets rotated. The latest log file is always <code>jetstream0.0.log</code> .
<code>-DJetStream.logfilecount</code>	30	The maximum number of rotated log files stored. The file with the largest number before ".log" is the oldest.
<code>-DJetStream.logfilepath</code>	./	The directory (with path) used to store the log files. By default, it's the same directory JetStreamQ is started in.

Tuning Parameters



 JetStreamQ

(page 2 of 4)

Parameter name	Default	Description
<code>-DJetStream.ha</code>	TRUE	By default, high-availability mode is on. As a result, all persisted messages are replicated to each remote JetStreamQ instance on the network. Setting this to FALSE turns off replication, and hence disables high-availability. In cases where HA is not desired or needed, setting this parameter to FALSE will increase performance and reduce network traffic.
<code>-DJetStream.persistfolder</code>	<code>~/jsjournal</code>	You can change the default directory for persisted message storage by specifying a path here. Regardless of the path you specify, the text “/jsjournal” will be appended.

Tuning Parameters



(Page 3 of 4)

Parameter name	Default	Description
-DJetStream.serverallowed	TRUE	By default, each instance of JetStreamQ on a computer can run in server mode. This enables automatic failover between instances. If you don't want a particular JetStreamQ application to run in server mode, set this to FALSE. Warning: this will impact message delivery if there are no other instances available to run in server mode.
-DJetStream.qsenders	8	Point-to-point message delivery is handled by a pool of dedicated internal worker threads. Use this parameter to change the default number of threads.
-DJetStream.topicsenders	8	Publish/subscribe message delivery is handled by a pool of dedicated internal worker threads. Use this parameter to change the default number of threads.
-DJetStream.psthrottle	1,000,000	To ensure a single fast publish/subscribe message producer cannot consume all server resources, the sender will be blocked when the total number of undelivered messages equals this value. It will be unblocked when the message count decreases below this value, plus a margin of 50

Tuning Parameters



(Page 4 of 4)

Parameter name	Default	Description
<code>-DJetStream.compress</code>	TRUE	By default, messages will be compressed when they exceed a specific size (see <code>-DJetStream.compressthreshold</code>). This saves network bandwidth, which can quickly become a bottleneck in message delivery performance. To turn off message compression, set this to FALSE.
<code>-DJetStream.compressthreshold</code>	1024	If message compression is turned on, message data is compressed for messages that exceed the specified size in BYTES.
<code>-DJetStream.compressSpeed</code>	TRUE	By default, compression speed is favored over size (the amount of compression). To compress messages as much as possible, trading off some performance, set this to FALSE. Use this only if you've determined that network bandwidth is a bottleneck, not CPU.

Scripts and Requirements

- * For Unix-based systems (Solaris, Linux, Mac OS X):
 - * jetstream.sh
- * For Windows-based systems
 - * jetstream.bat
- * Java SE 5 or 6 required. Java SE 6 recommended

Contact



* For downloads:

* www.alluretechnology.com

* For sales quotes:

* sales@alluretechnology.com

* For other information:

* info@alluretechnology.com

