APACHECON
DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

Connecting Tomcat to the World

Presented For The Apache Foundation By
LINUX FOUNDATION

What is a Connector?

- Tomcat's interface to the world
- Binds to a port
- Understands a protocol
- Dispatches requests

Tomcat Connectors

- Java Blocking I/O (BIO or sometimes JIO)
- Java Non-blocking I/O (NIO)
- Native / Apache Portable Runtime (APR)
- Java NIO.2

Types of I/O

- Polling
  - Straightforward API (peek)
  - CPU-inefficient
  - Thread loops while waiting for data
- Blocking
  - Straightforward API (streams)
  - CPU-efficient (blocking)
  - Thread stalls while waiting for data

Types of I/O

- Non-blocking
  - Complicated API (registration, event callbacks)
    - Channel
    - Buffer
    - Selector
  - CPU-efficient
  - Thread does not block: execution continues
  - When data is ready, the selector notifies observers

Common Connector Features

- Support for all protocols
  - HTTP, AJP, Websocket
- Support for all dispatch methods
  - Standard, Comet, Servlet 3.0 async
- Support for HTTPS (SSL/TLS)
- Acceptor thread(s) call accept() and hand-off
- Request processor thread pool

Blocking I/O Connector

- All I/O operations are blocking in processor thread
  - SSL handshake
  - Read request line (e.g. GET, POST, etc.)
  - Read request body
  - Write response
  - Read next request (HTTP keep-alive)
- Simple, stable, mature

Blocking I/O Connector

- Request throughput limited by thread count
- Clients can waste threads
  - Slow request line (mobile)
  - Aborted keep-alive stalls thread (default=20sec!)
- Unfair: accepted connections get priority for keep-alive requests

Blocking I/O Connector

- Single thread handles request after accept
- Uses Java Secure Sockets Extension (JSSE) for SSL/TLS

Non-blocking I/O Connector

- Single thread handles request after request-line
- Poller thread(s) manage non-blocking Selector
  - Read SSL handshake
  - Read request line
  - Wait for next keep-alive request

Non-blocking I/O Connector

- Block poller simulates blocking
  - Request header/body reads
  - Response writes
  - Processor thread sleeps during sim-blocking
- Uses JSSE for SSL/TLS
- Supports sendFile

Non-blocking I/O Connector

- Allows huge number of parallel requests
  - Not limited by request-processor threads
- Slow clients do not stall threads
- Aborted keep-alives die in the poller queue
- Simulated blocking adds overhead

Native Connector (APR)

- Single thread handles request after accept()
- Poller thread(s) handle certain I/O reads
  - Wait for next keep-alive request
- Some I/O operations block processor thread
  - SSL handshake
  - Read request line
  - Read request body
  - Write response

Native Connector (APR)

- Uses OpenSSL for SSL/TLS
- Supports sendFile

Native Connector (APR)

- Request throughput limited by thread count
- Slow clients can stall threads
- Aborted keep-alives die in the poller queue
- OpenSSL offers performance advantage
- Native code risks JVM instability

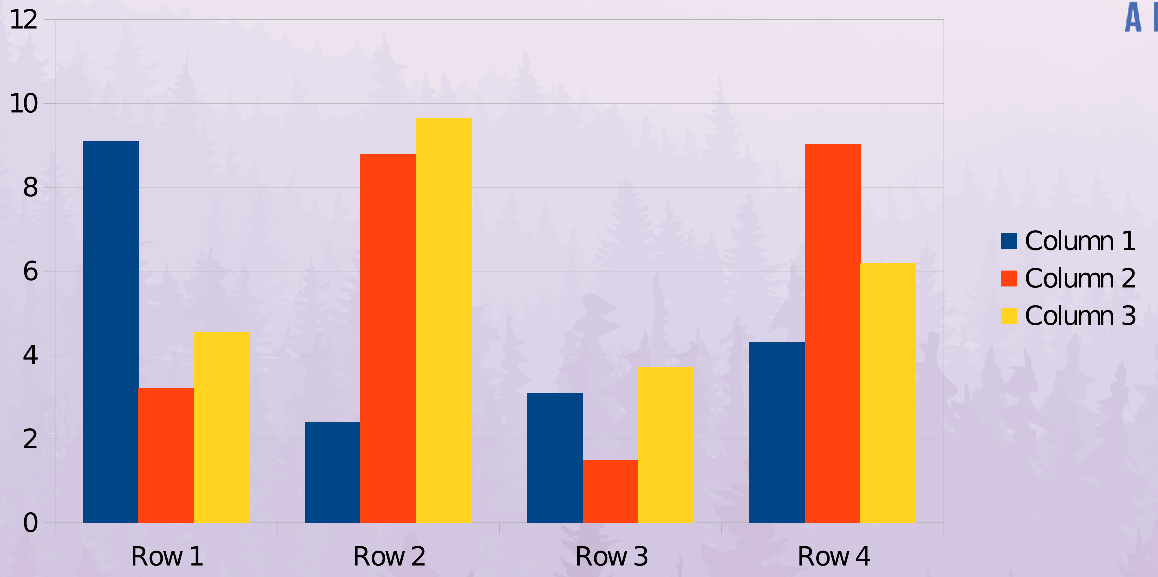
NIO.2 Connector

- like the NIO connector but uses the NIO2 framework.

Practical Notes

- Don’t try bother using non-blocking protocols with blocking connectors (BIO+Websocket = bad)
- AJP can be thought of as 100% keep-alive
- AJP doesn’t support HTTP upgrade
- Use of sendFile is highly recommended for any static-content (all but BIO)

Performances


Row 1
Row 2
Row 3
Row 4
Column 1
Column 2
Column 3