Open Source

in the Corporate World

JBoss Application Server

State of the Art: 2005

Aaron Mulder

CHARIOT SOLUTIONS

Agenda

- JBoss Basics
- J2EE Features
- Caching & Clustering
- Non-J2EE Applications
- Compared to the Competition

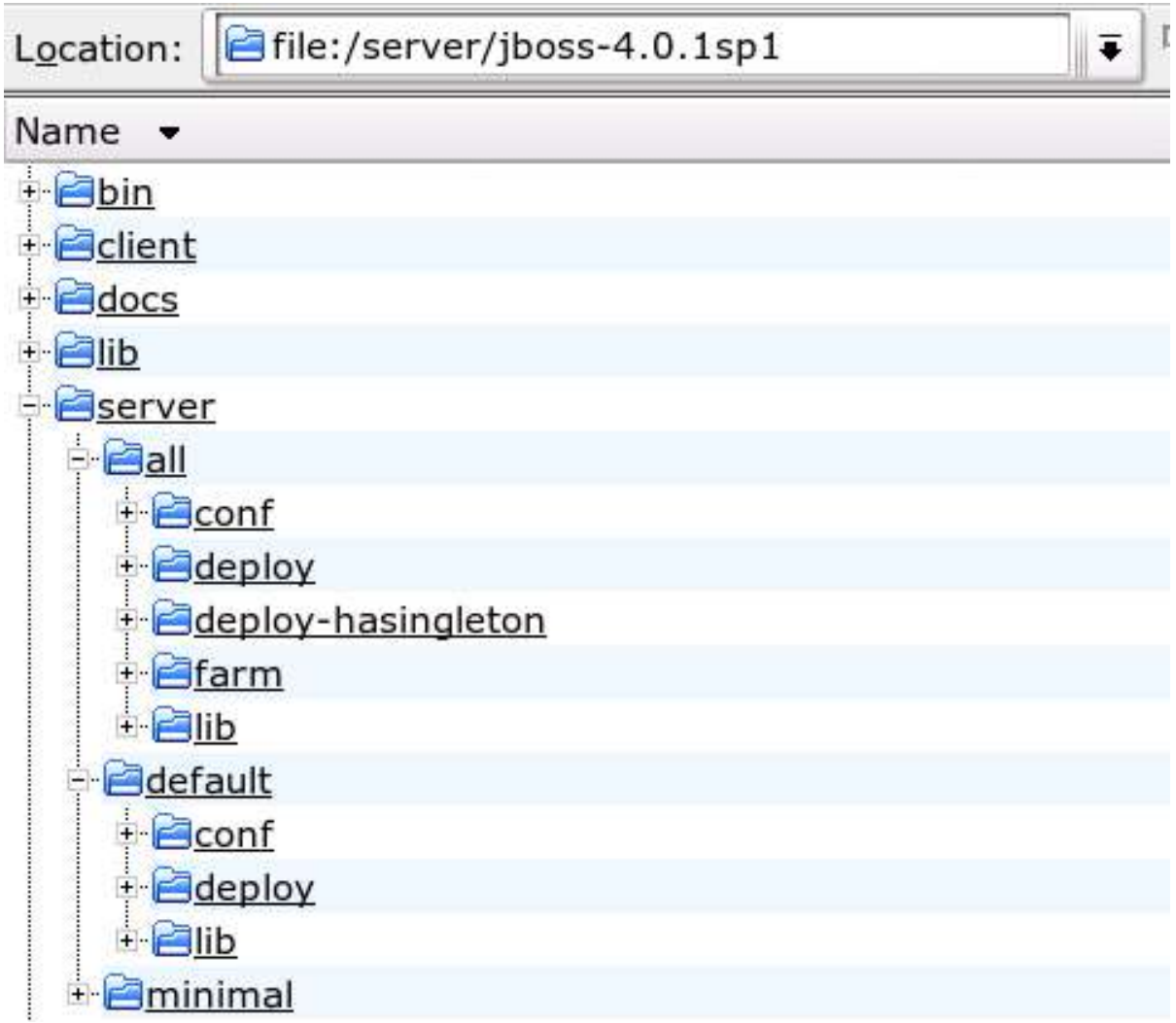
Introduction to JBoss

- Pure-Java Application Server
- Certified for J2EE 1.4 (some tweaks necessary)
- LGPL License
  - Extensions like security realms *may* need to be LGPL
- Online documentation (NYT-style)
- Trivial installation
- Primarily file-based configuration (w/ many files)
- Easy to develop for

Installation & Layout

- Simply unzip to install
- Start/stop scripts and so on in `bin/`
- Separate configuration directories under `server/`
- Each configuration has its own services, libraries, config files, deployments, logs, etc.
- Multiple configurations can be run in separate JVMs, though ports conflict by default

Directory Layout



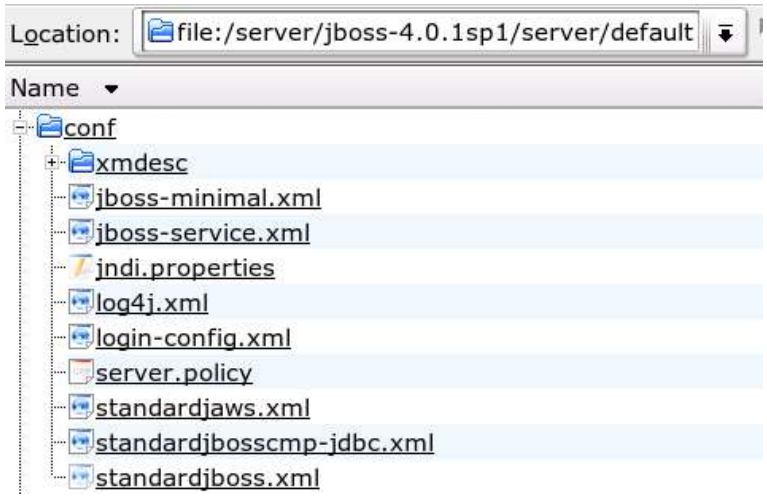
Startup & Shutdown

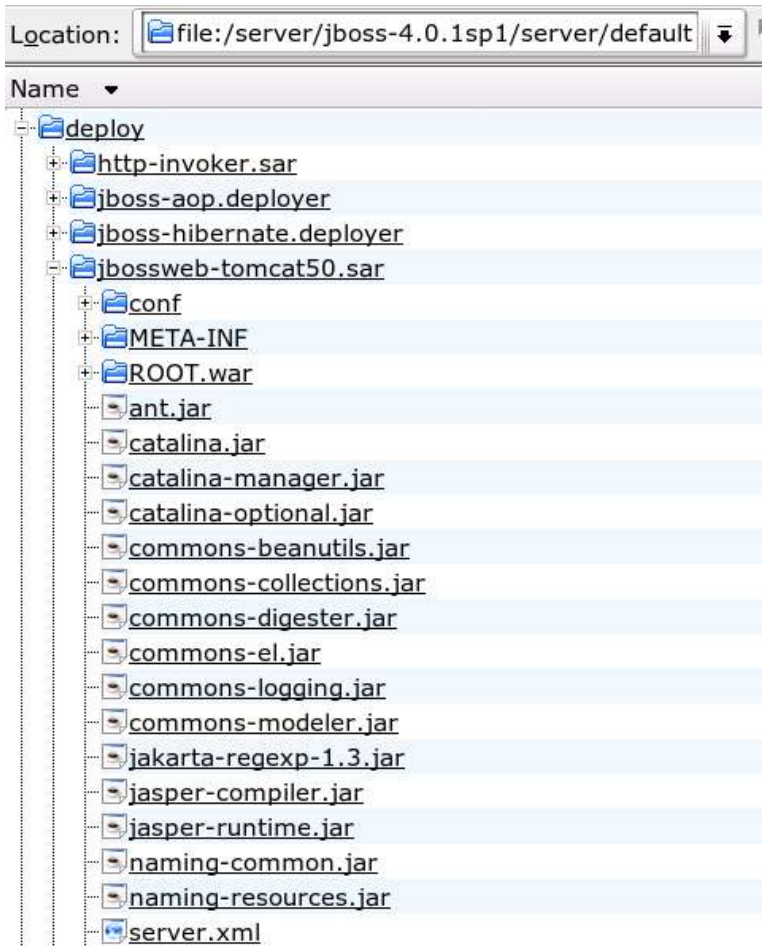
- To start, run `bin/run.sh -c configName`
- This runs JBoss in the current console, with both console output and a log file
- Easy stop: `Ctrl-C`
- (Remote) shutdown script: `bin/shutdown.sh`
- `bin/` directory also includes UNIX init scripts for automated startup
- Can customize startup scripts with memory settings, JVM tuning, System properties, etc.

Configuration Files

- Security, logging, basic product configuration files in `server/configuration/conf/`
- Feature configuration files (web server, JMS server, etc.) in `server/configuration/deploy/feature`
- Options like network ports are split across all these locations
- All configuration accessible programmatically via JMX, making scripting straightforward (though the syntax can be cumbersome)

Configuration Layout





Network Ports

- Numerous ports used by default
  - 1098: JNDI
  - 1099: JNDI
  - 4444: RMI
  - 4445: RMI
  - 8009: AJP (native web server connector)
  - 8080: Web application service
  - 8083: HTTP Class Loader
  - 8093: JMS
- More for clustered configuration, etc.

Logging

- Very early server logging is not really controllable, but there's not too much either
- As soon as the Log4J service is started, further messages are controlled by the log4j.xml configuration file, including all the features of Log4J (good and bad)
- Default logging levels typically produce 50-100 MB logs per day (for a small application), including accounts and passwords, etc.
- *You must trust your administrator!*

Services

- Includes a global JNDI space
- Includes JDBC Connection Pools
  - Normal or XA drivers
  - Added features like ID generator, exception interpreter, etc.
- Includes a JMS server
  - Supports normal or XA transactions,
  - Supports in-VM or remote JMS transports
- Includes JavaMail

Security

- No security on the server itself, JDBC, JMS, etc.
- Can configure multiple security realms for applications, using JAAS realm configuration syntax
- Ships with useful set of default realms, including properties files, JDBC, LDAP, etc.
- Different applications (or application modules) can be configured to use different security realms
- Configuration can be hit or miss – errors are not reported and some options are subtle

Management

- Includes two web management interfaces
  - JMX console
  - Web console
- Neither are very useful to non-programmers
- Typically better to hook up a third-party JMX management tool, such as MX4J
- Still no proper management interface that compares to (for example) WebLogic/Websphere web consoles
- SNMP features available, if you have the mojo

Agenda

- JBoss Basics
- J2EE Features
  - Web Applications
  - EJBs
  - J2EE Connectors
  - J2EE Compliance
- Caching & Clustering
- Non-J2EE Applications
- Compared to the Competition

Web Applications

- Uses Tomcat 5.0.x
- Supports JSP 2.0
- JSP Expression Language
- Expression language functions (which call Java methods)
- Custom Tags without Java Code
- Can use JSTL 1.1

JBoss Web App Config

- Saved in `WEB-INF/jboss-web.xml`
- Includes settings for:
  - Class loading configuration
  - Security realm, role mapping, run-as principal, etc.
  - Assign web app to virtual host
  - Session replication configuration
  - Web services configuration (client & server)
  - Resource mapping
  - Service dependencies
- Defined in `docs/dtd/jboss-web_4_0.dtd`

EJB Features

- Includes custom JBoss EJB container
- Supports EJB 2.1
- Enhanced EJB-QL (with ORDER BY!!!!)
- EJB Timer Service
- Session Beans as Web Services
- Message-Driven Beans for non-JMS messaging systems (and other JMS configuration enhancements)
  - Used in conjunction with Inbound J2EE Connectors

JBoss EJB Config

- Saved in `META-INF/jboss.xml` and `META-INF/jbosscmp-jdbc.xml`
- Includes settings for (among others):
  - Security realm, role mapping, run-as principal, etc.
  - JNDI names for client access
  - Cluster configuration
  - Web services configuration (client & server)
  - Resource mapping
  - Pool/cache tuning
- Defined in `docs/dtd/jboss_4_0.dtd` and `docs/dtd/jbosscmp-jdbc_4_0.dtd`

JBoss EJB CMP Features

- Can handle DDL for you (create/drop tables on start/stop, etc.)
- Can generate unique IDs in a generic way or specific to multiple RDBMSs
- (Very) extensive tuning options
  - Eager/Lazy load groups
  - Batch loads
  - Load on finder invocation or load on first access
  - CMR and nested CMR loads
  - Commit options, read-only, and read-mostly
  - Multiple EJB-QL overrides & fully dynamic queries

Cost of CMP Tuning

- Many options to consider
  - Commit option
  - Read only beans & methods
  - Instance per transaction
  - Optimistic locking
  - Row locking
  - Reading data outside transactions
  - Load strategies & tuning
  - CMR tuning
- No clear guidance on high-performance applications (particularly when clustered)

J2EE Connectors

- Includes custom JBoss J2EE-CA container
- Supports J2EE Connector Architecture 1.5
- Connector lifecycle management
- Work management
- Inbound connectors (messaging)
- Inbound transactions & recovery
- *Inbound features are undocumented*

JBoss J2EE-CA Config

- Saved in `deploy/*-ds.xml` (not in the RAR file!)
- Includes settings for specific instances of outbound connectors:
  - JNDI name for client access
  - Configuration values for options defined in standard DD
  - Security (login) settings
  - Connection pool settings
- Includes custom MBeans for inbound features
- Defined in `docs/dtd/jboss-ds_1_5.dtd`

J2EE Compliance

- The default installation favors performance and ease of development over compliance
  - All calls in the VM go by reference, whereas the spec indicates calls should go by value in some areas
  - A single “unified” class loader loads classes across all applications, to support call by reference and eliminate some class loader errors
- Certain settings need to be made to enable full J2EE compliance
  - Enable call-by-value for naming service
  - Turn off “unified class loader” and enable call-by-value between applications
  - Configure web application class loader for compliance

Agenda

- JBoss Basics
- J2EE Features
- Caching & Clustering
- Non-J2EE Applications
- Compared to the Competition

Clustering & Caching

- JBoss first built a clustering layer on JGroups
- As of 4.0, a new distributed cache package is available called JBossCache (also built on JGroups)
- JBossCache is currently used for HTTP session replication, but not fully integrated into EJB clustering

Clustering Features

- HTTP session replication for web apps
- Clustered JNDI
- “Smart stubs” for EJBs
  - All Stateless Session Bean calls load-balanced by default
  - Stateful session bean calls are “sticky”, but state is replicated for fail-over
  - Entity beans calls are “sticky”, and the database is used for concurrent access or fail-over
    - CMP engine supports SELECT ... FOR UPDATE
- Supports auto-discovery of cluster nodes

JBossCache Features

- Anything saved to the cache will be replicated to other nodes
- Can surround cache writes with transactions
- Can manage cache size (LRU, FIFO, etc.)
- Can hook the cache up to a persistent store (file, JDBC, etc.)
- Not currently integrated with EJBs (for example, to cache entity bean state in a cluster with the database as the persistent store)

EJB CMP Cache Features

- Completely separate from JBossCache (this slide just here to confuse you)
- When CMP optimizations are enabled, there is a cache of data read from the database
- When EJBs are accessed, they first read out of the data cache, and if that's exceeded, another block of rows is fetched from the database
- Data cache used for reading database data only; EJB instances in memory store EJB state for certain commit options

Caching Digression

- Some products include a cluster-wide data cache
  - Used for CMP Entity Bean state, reads and writes
  - Transaction-local cache space for concurrent access
  - Synchronized across a cluster
  - Includes an event model, with notifications on cache updates
- Not clear if JBoss will go this way
- Hibernate can hook into JBossCache, and Hibernate appears to be a likely JBoss CMP engine down the road...
- Performance and locking issues are not trivial

Agenda

- JBoss Basics
- J2EE Features
- Caching & Clustering
- Non-J2EE Applications
  - Hibernate Applications
  - AOP Applications
- Compared to the Competition

Hibernate Applications

- Can configure an MBean in JBoss to provide a basic Hibernate SessionFactory
- Can package all the Hibernate code and configuration into a single JAR file (a Hibernate ARchive, or `.har` file)
- Application code can look up the Hibernate session in JNDI, and load/save persistent objects from there
- JBoss can include Hibernate components in J2EE (JTA) transactions

Best Practices

- Package a `.har` file
  - Basic Hibernate config in `META-INF/hibernate-service.xml`
  - Classes laid out like normal in the JAR
  - Persistence mapping in `*.hbm.xml` files anywhere in the JAR
- For J2EE Applications
  - Include the `.har` in the EAR, using special `META-INF/jboss-application.xml` tags
  - Look up a Hibernate session using the HibernateContext class, for transaction integration
- Like J2EE apps, any database pools are available to all applications in the server

Costs & Benefits

- Hibernate uses POJOs, while CMP Entity Beans do not
- JBoss CMP implementation is extremely tunable, though Hibernate can use a JBossCache cache
- Both Hibernate and CMP can use XDoclet to generate anything more than the basic Java class
- Hibernate archives are not portable to other servers
  - Hibernate code still works, but basic Hibernate configuration, transaction integration, & session looking will be up to you

AOP Applications

- If you're not familiar with AOP...
  - It allows you to attach logic to various parts of your program (for example, attach some logging logic every time a getter is called – *any* getter)
  - The program and the attached logic (the “advice”) are stored separately
  - Can separate business logic from service (transaction/security/logging/metrics/etc.) logic
  - Code you're looking at no longer reflects everything that might happen in the system
- JBoss supports AOP features, alone or in combination with J2EE applications

Developing with AOP

- JBoss includes predefined aspects for transactions, security, notification, etc.
  - These can be configured and applied with annotations, e.g. “this method requires security role X”
- Write custom advice in plan Java classes
- Can use source code annotations (JavaDoc or J2SE 1.5) and/or XML config files to specify where particular advice should be invoked
  - Syntax is unique to JBoss
- Can precompile applications to incorporate the advice, or enable runtime handling of AOP

Costs & Benefits

- Separating different types of code makes each individually easier to maintain
- Runtime behavior is somewhat indeterminate: defined by two separate chunks of code, a mapping file, and whether you ran a precompiler or how you configured the JBoss AOP runtime
- AOP archives are not portable to other servers, though precompiling aspects with the JBoss tools and syntax results in normal classes that can be run anywhere
- Runtime mangling has a performance impact

Agenda

- JBoss Basics
- J2EE Features
- Caching & Clustering
- Non-J2EE Applications
- Compared to the Competition

JBoss Disadvantages

- Configuration is cumbersome
  - Limited help via comments in config files
  - Too many config files, settings spread out
  - Limited web management interface
- Uses too many network ports
- Configuration inappropriate by default
  - Super-verbose logging (also hard to tune well)
  - Too much running, not enough security
  - Not configured for J2EE compliance
- No security on the server itself

JBoss Advantages

- All the app server features you need, plus more
- Easy start/stop/deploy/redeploy for developers
- Easy to backup/restore, or copy configurations from one server to another
  - Multiple configurations for the same installation
- XDoclet support
- Free documentation (with registration)
  - Schema/DTD text included with the product
- Source code available

Summary

- Great for developers
  - Would be nice to have easier configuration, better defaults
- Mediocre for administrators (best for UNIX types)
  - Must be comfortable with the command line & XML files
  - Basic package needs extensive configuration
  - A number of security weaknesses
- Outstanding product for the price
  - Hard to beat for low-cost deployments
- Will be interesting to see where value-added products take it (BPM, Portal, etc.)

Questions?

CHARIOT SOLUTIONS