IBM WebSphere® Application Server V7 vs. JBoss® Enterprise Application Platform V5 TCO Analysis

Authors:
Jason Armstrong, Jeff Howell, Alex Wang (Summa Technologies)

Summa Technologies, Inc.
925 Liberty Avenue
Pittsburgh, PA 15222, USA
(412) 258-3300

summa
Technology + Business

December 29, 2010

Table of Contents

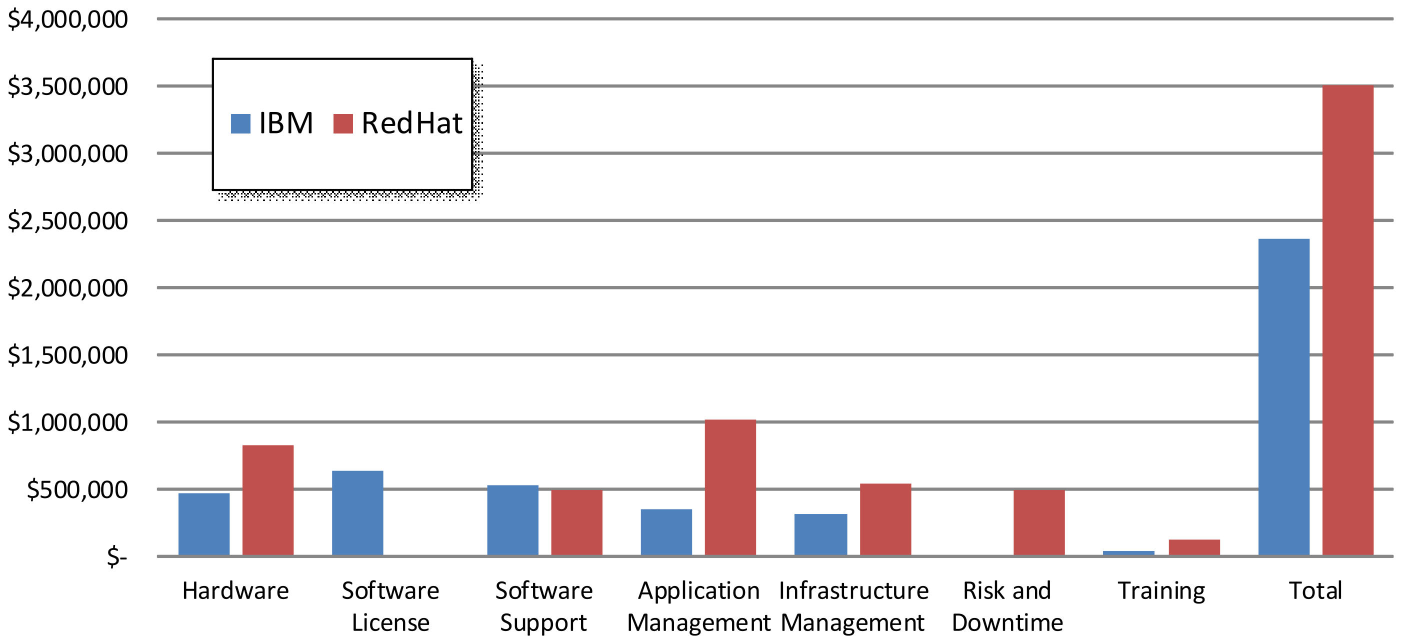
Executive Summary

To stay competitive in today's economic environment, more and more companies are looking to open source software as a means of reducing costs. In the application server market specifically, open source technology has grown in maturity and reliability, and is becoming an increasingly viable solution for many companies.

When considering open source, in most cases it is not the ability to modify source code that is attractive to customers, but the potential cost savings that open source can offer. While the easiest and most obvious costs to calculate are acquisition costs, often overlooked are the far-reaching effects that your choice of application server can have on other costs such as management, administration, training, hardware and opportunity costs due to unexpected downtime.

According to a number of independent industry studies, the cost of software system maintenance in the past two decades grew to be over 90% (see Appendix F). Your choice of application server should not be based on acquisition costs alone, but the Total Cost of Ownership (TCO) associated with a given product. To quote Gartner research note G00165072, March 2009: *"Products available for free (such as open source), or those that are a "same cost" swap out, can cost more during three- to five-year period than a first-time commercial purchase costing thousands of dollars. To understand operation life cycle costs, a number of key inputs are necessary to provide a more realistic assessment of the total costs of management products."*

This study provides an in-depth comparison between IBM WebSphere Application Server Network Deployment V7 (WAS ND) and JBoss Enterprise Application Platform V5 (JBoss EAP), measuring and quantifying the various costs associated with each product and providing a detailed comparison of their respective Total Costs of Ownership. Furthermore, this document provides a transparent and comprehensive presentation of the extensive testing that Summa Technologies performed for its analysis. The conclusion reached in this study is that in most environments, while acquisition costs for JBoss EAP are lower, IBM WAS ND provides lower overall TCO due to its advantages in stability, manageability, documentation, and performance. In medium configuration over a five year period considered in this study **WAS ND had 49% total cost of ownership advantage over JBoss** as shown on the diagram below:



This paper will help you determine whether your company is among the majority that can realize savings with WAS ND, or if an open source application server such as JBoss Enterprise Application Platform or other alternatives like Apache Geronimo, and IBM WebSphere Application Server Community Edition (WAS CE) would be more appropriate.

Key Findings

While JBoss has an initial cost advantage with zero software license acquisition costs, our results indicate that over the course of a five-year period the TCO clearly favors WAS and WAS ND. JBoss is also missing critical functionality as discussed later in the paper. Projecting over five years, our study results demonstrate a 49% total cost advantage in favor of WAS ND for medium and large environments, which can be broken down as follows:

TCO Category	IBM	RedHat	RedHat as % of IBM
Hardware	$ 470,074	$ 831,610	177%
Software License	$ 643,962	$ -	0%
Software Support	$ 530,579	$ 492,648	93%
Application Management	$ 348,873	$ 1,016,844	291%
Infrastructure Management	$ 324,090	$ 541,869	167%
Risk and Downtime	$ -	$ 502,281	N/A
Training	$ 46,875	$ 131,528	281%
Total	**$ 2,364,454**	**$ 3,516,779**	**149%**

Note: *In October 2010 version of the paper the TCO advantage of WebSphere over JBoss was determined to be 43%, however Red Hat has changed its license policy for JBoss and effectively doubled support cost for multi-core processors when they changed the licensing from socket based to be core based as of November 2010. This study uses this new core based JBoss license model, which is now similar to IBM's.*

For full disclosure of the recorded steps (Camtasia screen captures) and associated time measurements, server and licensing assumptions, as well as formulas used in this study please contact Summa or your IBM representative. Below is the description of some of the findings of the study:

- **Hardware Costs -** Based on the performance benchmark we performed in the lab, the hardware cost for JBoss V5 is 177% of the cost of IBM WAS ND V7 due to performance differences between the environments. This leads to a larger number of servers required to run JBoss and a more complex environment, which in turn requires more software support subscriptions and increased cost of administration as shown in the following sections. Please note that in a 2007 version of this study the performance difference between WebSphere V6.1 and JBoss V4.2 was only 39% in favor of WebSphere. In the current study (WebSphere v7.0.0.11 and JBoss EAP 5.0.1), we have seen the gap widen to up to 290% for certain application types. Moreover, looking at widely accepted industry benchmarks, IBM was the first vendor to publish and currently holds the record for the SPECjEnterprise2010 benchmark, measuring performance for JEE5 applications in terms of scalability (7,903.16 EjOPS) and also for performance (226 EjOPS /core). JBoss has never published results to the SPECjEnterprise2010 benchmark nor to the previous industry benchmark, SPECjAppServer2004.

- **License, Subscription, and Support Costs -** an annual subscription must be purchased to obtain JBoss EAP software. Annual subscription costs for JBoss are 42% of IBM WAS ND combined license and support costs over a five-year period. Based on the performance benchmark completed during this study, JBoss requires higher count of CPU support subscriptions to run the same workload as WebSphere. Another contributing factor is the fact that IBM bundles JDK, LDAP, Cache, WLM servers with WAS ND at no additional charge while JBoss customers must purchase license or subscription for this software separately (unless there is already existing LDAP, WLM and other software in place that can be reused). The difference in license and support cost is not a surprise as commercial software, such as WebSphere does initially cost more, but the benefits of it are realized over the longer period thanks to improved user experience.

- **Application Administration Costs –** The application specific administration labor cost for support of JBoss is 291% of the cost for IBM WAS ND. This is due to differences required in administrator skill level, the time required to perform similar administrative tasks, and the larger size of the JBoss environment that must be managed. While JBoss has enhanced its management capabilities with the latest version of JON, our research found that the new functionality contained significant quality and reliability issues. Meanwhile, WAS continues to have significant advantages in this area. As a result, overall, initial software investment favors WAS, as its additional license costs are made up for with its lower administrative labor costs.[1]

- **Infrastructure Administration Costs -** The infrastructure administration labor cost for JBoss support is 167% of the cost for IBM WAS ND. The production installation and management of the core application server is more difficult with JBoss and similar to the application administration described above, requires higher skill level and more effort to maintain. Additionally, the additional hardware that is required to run JBoss workload brings additional cost of the OS management.

- **Relative Risk and Downtime Costs -** The study found that JBoss does not provide capabilities such as XA transaction recovery, secure role separation between administrators, auditing capabilities, full featured administrative scripting framework and other important enterprise qualities of services. The lack of these features introduces risks and potential downtime. The impact of risk and downtime can vary significantly based on the type of the business and application. In our study we made the assumption that the annual business value of the project is $9,000,000 over 5 years. The approximate cost of additional risks and downtime for JBoss was conservatively calculated to be $502,281 higher than for WebSphere.

In addition to quantifying the total cost of ownership we also evaluated several qualitative factors that influence the overall user experience for WAS and JBoss. These factors may have an indirect impact on the TCO for an organization and should be considered when evaluating either product. Significant findings are:

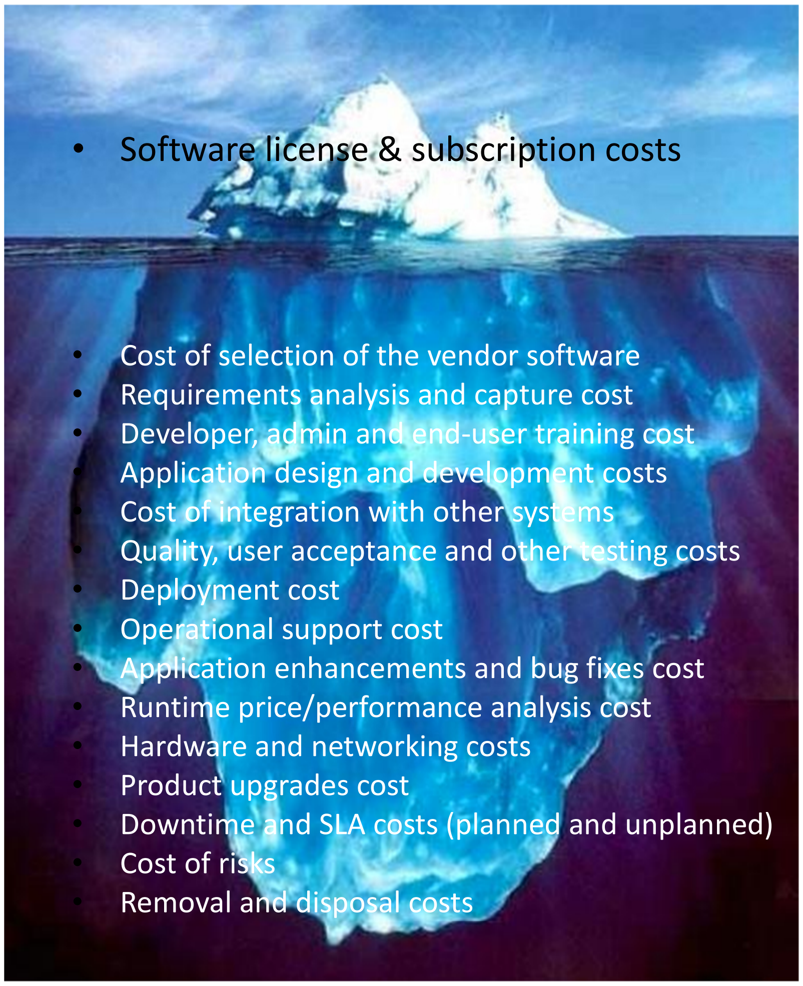
- **Performance** – Our performance tests indicate a performance advantage favoring WAS over JBoss by a factor of between 1.4:1 and 2.9:1. This difference in performance will increase hardware, software, and administrative costs for JBoss when attempting to achieve comparable performance results. We have conservatively used a performance factor of 170% (factor 100% would mean identical performance).
- **Documentation** – JBoss documentation is geared primarily towards developers. It contains many errors and in many cases requires administrators to identify the correct settings by trial-and-error. This can be a costly process, especially if an issue is not encountered until after a system enters production.
- **Failover Support and Transactional Integrity** – While JBoss has improved in some areas of failover support such as HTTP session replication, it still has issues with multi-node failures. In addition, IBM testing has shown that there are transactional integrity issues with JBoss as well.
- **Out-of-the-box Security** – Default shared installations of JBoss attempt to form clusters automatically. Extra work is required to segregate JBoss application servers into discrete clusters. In some instances, disabling the default mechanism causes other components, such as the web server integration, to not function correctly. A number of other security issues are also described later in this paper.
- **Compatibility Issues Between Admin Tools** – JBoss has added a new web-based graphical administration console, and an updated version of JBoss Operations Network (JON). Unfortunately, these suffer from a variety of quality issues, and a fundamental compatibility issue with the more traditional method of managing JBoss via editing of configuration files on the file system. Our experiences while testing this new administration console and JON indicates that administrators will still need to edit over half of the configuration settings via the file-system.
- **Unplanned Server Restarts** – When using the JBoss plugin (mod_cluster) we ran into issues where we had to restart the web servers if we restarted the application server, which would have resulted in outages that were visible to customers in a production environment. We encountered this in our testing

[1] The calculations in this study were done using US based labor costs.

for both clustered and standalone installations. Our testing of WAS found that restarting an application server did not require a corresponding restart of the web servers.

- **Backwards Compatibility** – Testing indicated issues with backwards compatibility between JBoss EAP 5 and JBoss AS 4. Additionally, we found backwards compatibility issues between the community-supported edition of JBoss AS 5.1 and the commercially-supported JBoss EAP 5. This latter issue is expected as Red Hat does not support the community version of JBoss and makes no claims as to its suitability for production or compatibility with any of their supported software.

When choosing an application server, organizations should carefully examine the costs and risks associated with a given product in the context of their staff capabilities, application and environment characteristics, and overall risk posture to determine which solution will have the lowest total costs in the long run. IT organizations must frequently launch software implementations and select products based on limited information. The assessment of cost is frequently based exclusively on initial product acquisition costs when the total costs to a business over the lifetime of an application are often difficult to measure and can include many other factors shown below:



- Software license & subscription costs

- Cost of selection of the vendor software
- Requirements analysis and capture cost
- Developer, admin and end-user training cost
- Application design and development costs
- Cost of integration with other systems
- Quality, user acceptance and other testing costs
- Deployment cost
- Operational support cost
- Application enhancements and bug fixes cost
- Runtime price/performance analysis cost
- Hardware and networking costs
- Product upgrades cost
- Downtime and SLA costs (planned and unplanned)
- Cost of risks
- Removal and disposal costs

In addition to failing to account for these costs, other pitfalls in calculating costs include the fact that direct labor cost estimates typically fail to account for the actual skill levels required for a task, especially for advanced configuration and administration tasks. Managers also frequently fail to account for the development of additional non-functional support code needed to maintain deployments, along with the ongoing care and maintenance needed for complex systems.

Cost Factors

The relative impact of TCO factors can vary widely between projects depending on the environment and type of application being deployed. The TCO factors considered in this study are described in the table below.

TCO Area	*Example Costs*
Software Licensing and Software Support	▪ Initial license and subscription acquisition costs for all elements of a solution, including application server, operating system, web server, load balancer, LDAP, HTTP server, JVM, etc. ▪ Ongoing product support and subscription costs
Hardware	▪ Server hardware and support costs ▪ Cooling and power costs
Application Management	▪ Installation and configuration costs associated with software development or implementation projects ▪ External system interface integration, support and testing costs ▪ Performance tuning and monitoring of application server environment ▪ Troubleshooting time for applications and development environments
Infrastructure Management	▪ Deployment, monitoring and routine maintenance of application software ▪ Application of fix packs and patches to the OS and Application Server ▪ Upgrade of servers to newer versions ▪ Backup and restore of the OS and Application Server
Risk and Downtime	▪ Automated recovery from failures, including heuristic XA transactions ▪ Failover time of JMS queues and destinations, HTTPSessions and other stateful services, server restarts and memory leaks ▪ Consistency checks on configuration to prevent illegal settings ▪ Security tools for separation of administrative roles and auditing ▪ Impact of encryption of intra-cluster communications
Training	▪ Training/education for administrators and developers
Additional Considerations	▪ Costs to extend products to support service level agreements (SLAs) ▪ System integration costs for development and deployment of integrated services such as packaged applications (e.g. SAP), messaging products (e.g. WebSphere MQ, TIBCO) and LDAP and database servers ▪ Costs for custom development and testing of non-supported features

One factor not included are software development costs, which we decided not to directly measure due to their high degree of variability based on an individual organization's software development practices, team sizes, and skill level. Based on our hands-on testing of the products, we have included notes and calculations where we believe the software development costs would be different based on different capabilities of the products, and where appropriate, we have extrapolated possible TCO implications from those differences.

Analysis Findings

We executed a number of hands-on tests to calculate administrative costs and to gain a better understanding of the qualitative aspects of those costs for the products. The following is a list of the test scenarios executed:

Standalone Tests

- Install and Configure Application Server
- Upgrade Software from Previous Release
- Manual Application Deployment
- Configure LDAP for Administrative Security
- Scripted Application Deployment
- Apply Fix Packs and Hot Fixes
- Web Service Performance Tests
- JEE5 Performance Tests
- Heuristic Transaction Recovery Tests

Clustered Tests

- Install and Configure Application Server
- Upgrade Software from Previous Release
- Add Additional Server to the Cluster
- Configure XA and Transaction Logging
- Manual Application Deployment
- Configure LDAP for Administrative Security
- Scripted Application Deployment
- Apply Fix Packs and Hot Fixes
- HTTP Session Replication Failover Tests
- JMS Failover Tests

The TCO areas measured as a part of the study are discussed in the following sections.

Performance

TCO Advantage: IBM WebSphere Application Server

In this version of the study, we elected to perform several variations of standalone performance tests, instead of long-running clustered performance tests. This shift in testing strategy was due to feedback from conversations with customers who indicated that very few of them were deploying JBoss in a clustered environment. Instead, they were deploying it for a purpose driven application under standalone conditions. We believe that by using the results for a standalone server we can extrapolate a best case scenario for clustered operations. Our performance testing strategy included the following test scenarios:

- 48-hour, 12 hour, and 1 hour tests of WAS and JBoss using Apache DayTrader
    - Single application server machine (with multiple JVMs if needed)
    - Single, remote web server machine
    - Remote database server running IBM DB2
    - Used EJB3 for the persistence tier
- 2-hour web service test with moderately complex payload (SOAP/HTTP, 4 kB in, 2 kB out)
    - Single application server machine with multiple JVMs if needed
    - Single, remote web server machine
    - No database
    - JAX-WS annotated POJO service

As part of the performance testing, we spent significant time tuning both JBoss and WAS. The software and hardware configurations used in the test are described in the “Appendix C”. Average server CPU utilization was greater than 90% with JBoss and WebSphere.

Prior to each of the performance tests being executed we tuned the configurations for each environment (JBoss and WAS) to achieve optimal performance. There are a number of interesting discoveries that were made as part of the tuning process.

Tuning of WebSphere was a relatively straightforward process, and was completed in less than one day. WebSphere built-in Tivoli Performance Viewer, even with its non-Web 2.0 user interface, was an important tool throughout the tuning and monitoring process because we were able to review relevant stats quickly and easily including changes in values over time. We also found that the tuning advisor in WebSphere recommended more conservative settings and in our case were only used as a starting point for the tuning process.

The process of tuning JBoss took two weeks to complete and required a deeper understanding of the inner workings of JBoss. The Red Hat JBoss Community and best practices were referenced to identify tunable parameters that would provide the biggest impact to performance. For detail performance tuning recommendations, refer to the "Best Practices for Performance Tuning JBoss Enterprise Application Platform 5". In addition to tuning JVM heap size, thread pools and many other settings, we have also considered a number of Red Hat Performance Lab tuning recommendations specific for DayTrader (as presented during several sessions at the JBoss World conference in Boston in June 2010), including, but not limited to:

- Utilize JBoss Message Resource Adapter and JMS connection pooling
- Use StrictMaxPool for EJB3 Stateless Session Pool
- Disable tracking of unclosed connections in jca-jboss-beans.xml
- Start multiple JBoss JVMs to work around the bug in the JBoss Transaction Manager locks
- Use EJB3 local interface to avoid serialization of remote interface calls

We also identified several qualitative issues were also identified during the course of performance tuning and testing which will have an impact on TCO. We found that tuning JBoss is a more labor intensive task than tuning WebSphere and required a higher level of skill resources. Several of the reasons are highlighted below:

- **Little Actionable Information in Documentation** - The JBoss website (including documentation, wikis, and forum postings) yielded little actionable information. We had to use third-party websites to gain better understanding of what the various configuration options were and what they meant.
- **Incorrect Information in Documentation** – The JBoss website indicated a series of configuration changes at the EJB layer which would control EJB pool size. These changes required updates to either the source code or the JBoss specific deployment descriptors. After making the changes in the deployment descriptors and troubleshooting difficult to interpret error messages, we were able to implement these changes, only to find out that JBoss EAP 5 does not actually use these settings.
- **Issues with Performance Monitoring Tools** - Due to quality issues with JON monitoring tools, we found that the best approach for tuning JBoss involved using native OS tools, reading log files, and working with the JMX admin console. Unfortunately, the JMX admin console is unable to show statistics for more than one component at a time, so we had to constantly switch back and forth between component views.
- **mod_cluster Scalability Issues** - During the tuning process we found that JBoss' mod_cluster was a bottleneck. We added a second web server for the JBoss tests. We ultimately removed the web servers for the 12-hour tests to reduce the number of test variables and provide a more direct comparison of application server performance.
- **JBoss Restarts** – We found that after most configuration changes were made, including those made through the JMX console, JBoss had to be restarted. Two notable instances related to server restarts included:
    - **Increasing JDBC Connection Pool Size** - Increasing the database connection pool via the JMX console resulted in the application failing due to database connection errors. The only way to eliminate these errors was to restart JBoss.

  - **Restarting JBoss Required a Web Server Restart** – Due to issues with mod_cluster, we found that when the application server had to be restarted, we also had to restart the web server, due to an invalid checksum during the server handshake process.

Our performance tests indicate a performance advantage favoring WAS over JBoss by a factor of between 1.4:1 and 2.9:1. This difference in performance will increase hardware, software, and administrative costs for JBoss when attempting to achieve comparable performance results. We have conservatively used a performance factor of 170% - a WebSphere advantage (factor 100% would mean identical performance). For the complete documentation of the performance test, project files, load scripts and data files, please contact your IBM representative.

Hardware Costs

TCO Advantage: IBM WebSphere Application Server

We executed several performance tests for JBoss EAP and WebSphere Application Server in single server configurations using the Apache DayTrader test application and a simple web service implementation running under a simulated load. Details of the tests can be found in the performance testing section of this paper.

Based on the results of the performance test, a JBoss-based solution will need between 40% and 290% more hardware to handle the same load as a WebSphere-based solution. Our TCO estimate uses a conservative 170% for the cost calculations in the report.

The TCO components for software licenses and support/subscriptions costs favor WAS ND over the five-year period by 177% ($470,074 vs. $831,610). These numbers reflect the additional subscription costs needed to support an equivalent load on JBoss along with modest growth predictions of 5% transaction growth per year and the creation of 4 new applications of similar size and complexity to the DayTrader application each year.

Software License, Maintenance and Subscription Costs

TCO Advantage: IBM WebSphere Application Server

The mantra of commercial open source has long been, “If you don’t need support, don’t pay for support. If you need support, buy support.” For some open source products and use cases this may be true. However, our testing indicates that a JBoss subscription is essential.

Through the course of our analysis we compared the community supported version of JBoss AS to that of the commercially supported JBoss EAP product. Of the 32 open source projects, which make up both products, we found that JBoss EAP uses newer versions for 14 of the components. To get the same version levels of these projects, one would need to manually integrate these projects. Manually integrating these 32 open source projects into an equivalent “free” package, similar to what CentOS has done for RHEL, would be very time consuming and costly.

For the TCO study we obtained list pricing for both JBoss EAP[2] and IBM WebSphere Application Server.[3] We decided to use Red Hat’s premium support pricing for TCO calculations since it provides the closest match to IBM’s basic 24x7 support. The hardware configuration used for our license cost analysis assumed 2 CPU sockets per machine with each socket having 4 cores, thus bringing total number of cores on a machine to 8. For this configuration IBM requires the purchase of 560 PVUs per machine while JBoss requires 0.5 bundles per machine of their 16 core annual subscription license bundle.

There are many factors that can affect software licensing costs. The following is a list containing some of the more important items to take into account when considering alternatives:

- Both Red Hat and IBM will discount pricing based on volume of business and application environment deployment, affecting individual cases very differently. The license and support cost analysis performed

[2]JBoss NA Channel SKUs, http://www.redhat.com/f/html/jboss_channel_skus.html.
[3] IBM WebSphere Application Server Pricing, http://www-01.ibm.com/software/webservers/appserv/was/appserv-was-pricing.html.

in the study is based on "street" prices assuming 27.0% discount off the list price for IBM and 27.0% discount for Red Hat. We believe both of these are appropriate for the configuration of the size used in the study, however there is some evidence that suggests that JBoss discounts do not go above 20%.

- JBoss pricing requires purchasing capacity in groups of 16 or 64 CPU cores (as opposed to their old way of counting CPUs by "sockets" before November 2010). If someone needed a configuration with fewer than 16 total cores, it is not clear how Red Hat will price their product.
- Red Hat does not longer provide JON subscriptions free of charge with JBoss support and now requires all customers to pay for JON separately (in the past JON was free for those clients who purchased over 32 or more CPUs of JBoss support).
- Both Red Hat and IBM have similar policies regarding sub-capacity licensing when using virtualization technologies such as VMware or other hypervisors.
- WAS-ND includes WebSphere Edge Components which have been included in our TCO analysis. Configurations that use the WebSphere Edge Components will have additional cost benefits over JBoss EAP for functions such as load balancing and edge caching; potentially eliminating the need for other hardware or software products and custom integration effort when using WebSphere.
- WAS-ND includes free licenses of the Tivoli® Access Manager, which can act as an LDAP server; JBoss EAP does not include a bundled LDAP server so one will be required. The cost of a Red Hat Directory Server was factored into the analysis for JBoss.
- Since June 2009 IBM offers IBM WebSphere Application Server for Developers Edition (WASDE) as either a no-cost community supported version or a paid support version. WASDE is the exact same code as WebSphere Application Server Base. Customers who have at least one full WebSphere license will also have access to the Rational Application Developer Assembly and Deployment Tool.[4] This tool is a subset of Rational Application Developer which can be used to develop and package applications for deployment on WebSphere.

The TCO components for software licenses and support/subscription costs favor WAS ND over the five-year period by 93% ($1,174,542 vs. $1,509,492). These numbers reflect the additional subscription costs needed to support an equivalent load on JBoss along with modest growth predictions of 5% transaction growth per year and the creation of three new applications of similar size and complexity to the DayTrader application each year.

Application and Infrastructure Administration Costs

TCO Advantage: IBM WebSphere Application Server

As part of the TCO study we executed a number of time motion scenarios to understand the steps involved in various administrative tasks, the relative complexity of the tasks and issues that were encountered while carrying out these tasks. Specific scenarios evaluated include the installation, upgrade, migration, configuration, clustering, and security hardening of an environment for a moderately sized application environment. For each test case, we calculated its impact on a five-year comparative TCO difference.

As part of the testing we also studied the effects of the learning curve associated with each of the products along with the skill level required for each product. The cost calculations use the best case results from each of the test cases to prevent over inflation of administrative costs associated with newer administrators coming up to speed.

With regard to the learning curve, our tests indicate that WAS ND has a lower overall learning curve than JBoss. We found that execution of a given task with JBoss can take between 122% and 321% longer to

[4] IBM WebSphere Application Server for Developers FAQ, http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/faq-wasdevelopers.html.

complete than with WebSphere. We also found that on average, the skill-level required to complete tasks with JBoss was higher than the skill-level required with WebSphere.

The remainder of this section describes the various scenarios tested, the results of each test, how the results correlate to TCO, and a discussion of notable findings encountered during the course of testing.

The study divides administration costs into two categories. The first is application administration, and the second is infrastructure administration.

Application administration costs are the costs most directly tied to managing an application during a software development project. These are the costs associated with standing up shared development environments, QA environments, and the initial production environment. Many organizations treat this category as either capital or planned work projects.

Infrastructure administration costs are focused on the day-to-day operations of keeping a production environment up and running, applying fix packs, or adding additional capacity to support increased application demand. Many organizations tend to categorize this work as expense work. After the 2007 study, we decided to split these costs into two categories to better reflect how many readers viewed administration costs.

When looking at application administration costs we found that over the five-year period, WAS has a 291% advantage over JBoss ($348,873 vs $1,016,844). Infrastructure administration costs favor WAS by 167% over the same period ($324,090 vs. $541,869).

Installation and Configuration

TCO Advantage: IBM WebSphere Application Server

This test focused on the initial installation and configuration of the application servers. The test case captured and compared the initial product acquisition, download, installation and configuration experience for JBoss, WAS and WAS ND. It also covered initial deployment (non-clustered) of the test application. The following list outlines key factors and findings that affected the test case TCO outcome:

- Production Server Installation Time
    - The initial download and installation of a single standalone JBoss server suitable for development purposes is simpler and smaller than WAS.
    - Applying patches and hot fixes to JBoss is a more labor intensive process than WAS.
- Server Configuration
    - Security hardening of JBoss administrative components for production use is more time consuming and difficult and JBoss Operations Network (JON) will not be able to help with these tasks:
        - Configuring the server for LDAP integration requires manually editing various files on the file system.
        - Securing JMS communications requires configuring various files on the file system. The documentation is incorrect on how to effectively accomplish this, and there are many conflicting sources of information on JBoss discussion forums, wiki pages, and blogs.
    - Security hardening of WebSphere administrative components is much simpler.
        - During the installation process, the administrator is prompted to select a username and password for the admin console.
        - Once initial installation is complete, the administrator can use a wizard in the administration console to connect the server to a variety of LDAP servers.
        - InfoCenter provided accurate steps for configuring access to our Microsoft Active Directory.

- Prebuilt Configurations
  - JBoss provides a number of pre-configured server configurations based on usage type. However, the differences between the configurations are poorly documented. It is also unclear what is required to create your own combinations of servers. Switching from one configuration to another requires maintaining the application binaries and relevant configuration information or third-party libraries within multiple directories on the filesystem.
  - Based on the applications deployed to a server, WebSphere will determine which components, such as web container, EJB container, JMS, etc., should be loaded at runtime. This approach does not require administrators to take any action, or configure a server any differently to optimize the server's runtime footprint. For an organization with multiple servers, this greatly simplifies the configuration, deployment, and optimization process for servers.
  - WebSphere provides an option for optimizing a server for development or production use. We found that in the default production mode WAS will enable features that increase startup time but improve overall efficiency when the application server is running for extended periods of time. The development mode decreases the server startup time by deferring initialization of these components until they are first used. Switching between production and deployment mode is simply a matter of clicking a checkbox and restarting the server; applications do not need to be redeployed or reconfigured.
- Web Server Configuration
  - Our testing indicated that IBM HTTP Server is not supported by JBoss's mod_cluster plug-in for web server integration. Because of this, we had to manually compile and install Apache HTTPD server for our JBoss environments.
  - The mod_cluster JBoss component uses IP multicasting to bind web servers and app servers together. We found several issues with this approach:
    - JBoss and mod_cluster by default listen to a specific multicast address. Failure to change this default address can lead to ad-hoc clusters being formed. This approach is counter to the principles of least privilege and least surprise. In our testing, two separate administrators accidentally created a cluster of two JBoss instances that were meant to be separate.
    - Changing the default multicast address led to other unexpected behavior during application server shutdown, including errors from the JMS provider.
    - It is possible to disable multicast advertisements, but this also led to unexpected behavior. In particular, restarting JBoss required a restart of the web server due to an "invalid checksum" error encountered during the binding process.
    - The JBoss documentation is vague on how to effectively and securely configure the integration between the web server and the app server.
  - In our JBoss testing, we also found the use of the mod_cluster-manager component is essential for verifying, diagnosing, and monitoring the web server configuration. This component must be configured using Apache's <Location> directive, which means that administrators should also be sure to secure the URL properly (i.e. restrict access to the URL from outside of the DMZ).
  - WebSphere provides a plugin for a variety of web servers including Apache HTTP server and IBM HTTP Server (IHS). For our testing we used IHS, based on our experience, shops using WebSphere tend to also run IHS. We could also use the IBM installer to automatically install the web server and plugin simultaneously to our server.

- Configuration of the web server for use with WAS and WAS ND was relatively straightforward and the initial integration of the web server with the app server was accomplished through the use of a script automatically generated by the installer.
- The one issue with verifying that the application was deployed correctly was confusion surrounding WebSphere virtual hosts and ensuring that host names and web container ports are configured correctly in the admin console.

Customers who have at least one WAS ND license can take advantage of the Central Installation Manager (CIM) functionality. CIM provides administrators with the ability to remotely install new installations or apply fix packs to a server via the admin console.

Using CIM, an administrator can install the deployment manager and at the same time create an initial CIM repository. The administrator can then identify target servers on which to have WAS installed, either to be a federated standalone server or a clustered server. The administrator specifies the host name, OS type, user name, and password to use for the installation process. The administrator can also add additional WAS installation media for supported operating systems to the repository. For example, a CIM repository installed on Windows can also contain installation media for Linux and Solaris. Once the administrator has provided the connection information and the appropriate responses to be used during the installation process, he can kick off the installation process from the admin console, and periodically check the installation progress. Multiple nodes can be configured at the same time, using this functionality.

WebSphere Fix packs can also be deployed in a similar manner. An administrator first installs the fix packs on the deployment manager, adds the update installer and fix packs to the CIM repository (or optionally downloads them using admin console), and then selects which servers to install the fix packs on. CIM automatically makes sure the target server has the correct version of the Update Installer installed, and then proceeds to update the server without interaction or intervention required by the administrator. Our administrative scenarios for WAS include the use of CIM for managing WAS ND based environments.

JBoss does not have equivalent functionality for automatically installing servers in a remote or distributed manner.

Overall, we found that administrative tasks with WebSphere were simpler to execute and did not require highly skilled administrators to be successful. WebSphere's out-of-the-box settings are more consistent with the principles of least permission and least surprise. WebSphere's administrative console has been consistent between the last several releases with minor changes to the navigational structure of the console; experienced WebSphere administrators will be able to quickly adjust to these changes. Using publicly available support information, such as IBM InfoCenter, and the built-in help for the admin console, administrators should be able to quickly install and configure standalone or clustered environments.

JBoss administration requires highly skilled administrators with a deep understanding of JBoss internals. While developers have been requesting a more fully featured administrative console to replace the JMX console, our testing indicates that both the JMX console and direct file editing are still required. Furthermore, we would advise customers against using the new admin console until it has more time to mature and become more reliable. Documentation continues to be challenging and is primarily targeted towards developers instead of system administrators. In most of our test scenarios we had to supplement or replace the JBoss documentation with information found on JBoss related discussion forums, wikis, and blog postings.

While evaluating the new embedded admin console and JON we encountered several inconsistent results. For example, when deploying an application with an error in a deployment descriptor we received a JBoss error that the deployment failed. Later when we tried to install the application after fixing the deployment descriptor, we received an error that the application couldn't be installed, because it was already installed. Unfortunately, the application did not show up in the list of deployed applications until we restarted the application server. After restarting the server, the application appeared in the list, but when we attempted to remove the application it indicated that the application could not be removed because it was not currently deployed; another server restart and things appeared to be corrected.

In another case, a WAR file was successfully deployed into JBoss, but could not be undeployed through the admin console or JON. In the end, the WAR file had to be manually deleted. These errors did not happen once. We had repeated the experiments multiple times.

The JBoss documentation also indicates that dual editing of files, or editing configuration settings via files and the admin console, is not fully supported. The JBoss EAP documentation explicitly states that resources deployed and modified by the console should be modified only by the admin console. It also states that the changes made in the console are not persisted directly to the original configuration file.[5]

This issue does not exist when using WebSphere's admin console or administrative scripting client.

To summarize, WebSphere production environments are easier to install and configure than JBoss EAP environments while JBoss is easier and faster to install on a developer desktop.

Cluster Administration

TCO Advantage: IBM WebSphere Application Server

Our testing focused on analyzing the following conditions related to clustered operations of WebSphere and JBoss. Specific cases being analyzed were:

- Differences required in moving from a standalone configuration to a clustered configuration
- Ability to securely configure and administer clusters
- Evaluate the effort to add a new node to a cluster
- Configuration and handling of various failover scenarios related to HTTP session failover and JMS recovery operations

The following list outlines key factors and findings that affected the test case TCO outcome:

- Required Skills
    - Application server cluster configurations for JBoss EAP require advanced skills in planning and architecting the clustered environment. This includes understanding the security design and configuring communications among web servers and clustered application servers. It also requires a deeper understanding of JBoss internals including mod_cluster, jbossweb, and JGroups caching.
    - Configuring a clustered WebSphere configuration can be accomplished by reading the available information in the InfoCenter and executing the recipes. A basic understanding of WebSphere architecture is required and can be gained from available documentation.
- Application Deployment
    - With JBoss AS 5.1 and JBoss EAP 5, farm deployment is officially supported again. It had been removed from JBoss AS 5.0 and 5.0.1. It appears that JBoss EAP 5 has added a number of configuration options to improve the configurability of the farm service.[6] It is enabled by default in the "*production*" and "*all*" configurations. Given the potential for abuse or confusion, the default settings used by this service appear to violate the principles of least surprise and privilege and documentation for this service is very vague.
    - With WebSphere Application Server, application deployment in a clustered environment is very similar to that of a standalone application. The application is centrally deployed through

[5] Chapter 11 of the JBoss EAP 5.0 Administration Console Quick Start Guide, http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/5.0.0/html/Administration_Console_Quick_Start_Guide/Administration_Console_Quick_Start_Guide-Updating_Resource_Configurations.html.

[6] http://community.jboss.org/wiki/FarmDeploymentsinJBossAS5x

the administration console, and the cluster is selected as the target instead of a standalone server. The changes are then automatically synchronized to nodes.

- Cluster Creation
    - As mentioned in the previous section, configuring a JBoss cluster is very easy to do, and can be done accidentally, by simply using the default settings. However, the steps required to correctly configure and isolate multiple clusters on the same subnet requires much more time and effort.
    - Creating a cluster using WAS ND is a more deliberate process. In order for nodes to join a cluster they must explicitly be added to the cluster by an administrator. Unlike JBoss clusters, there is no guess work or accidental creation of a cluster with an out-of-the-box installation.
- Secure Cluster Communications
    - Unsecured cluster node addition is much easier in JBoss AS than WAS ND. The impacts of this are discussed in further detail later in this document.
    - By default communications between nodes in a WAS ND cluster use SSL and LTPA for intra-cluster communication.
- Node Installation and Addition
    - We used WebSphere Central Installation Manager Functionality for installing and patching our WebSphere cluster. There is no equivalent functionality to CIM in JBoss. More details about our experiences with CIM can be found later in this document. We found that this greatly reduced the amount of human time involved in the setup and configuration of application servers when using WAS ND (either clustered servers or federated standalone servers). This greatly simplified the cluster configuration process, because we had the option of adding the node to the server as part of the server installation process.
    - Adding a new node to a cluster in JBoss is the same as installing a new server. All steps must be manually completed via the command line. Most configuration files can be copied from one server to the next, but may require modification to preserve each node's unique identifier.
- HTTP Session Replication
    - HTTP session replication is difficult to configure although the documentation for configuring JBoss in this area is much improved over past releases. However, while testing HTTP session replication we found that some of the settings had no measurable impact.
    - Implementing session replication with JBoss was not possible with JON and required file access to change both the *web.xml* and the *jbossweb.xml* deployment descriptors. We also found that we had to have a more thorough understanding of the test application in order to select the appropriate synchronization parameters.
    - WebSphere supports several different flavors of HTTP session replication. These options range from session persistence in a database, to memory-to-memory using buddy groups, to memory-to-memory using active and passive failover. This can be configured either through the administrative console or the command-line wsadmin tool and does not require any changes to deployment descriptors to enable session replication.
    - Results of our failure testing can be found later in this document including the situations encountered with JBoss that resulted in noticeable customer outages.
- JMS Failover
    - Both application servers require a senior level administrator to properly configure JMS for high availability operations.

- WebSphere is less complicated to configure through the systems integration bus (SIBus) pages in the Admin Console.
- JBoss required modifications to several XML files to properly configure clustered JMS. Including the configuration of a JDBC data source and various property files. The command-line parameters to the server also had to be modified to include unique peer ids so each JMS server could be uniquely identified within a cluster
- JBoss uses the same multicasting approach for locating JMS servers and requires the same care to properly separate nodes. In contrast, WebSphere uses explicit configuration to identify which servers should be part of an SI Bus.
- Results of our failure testing can be found later in this document.

In summary, despite JBoss's attempts to simplify cluster configuration and to correct the propensity to create ad hoc clusters under default conditions, we found the process of configuring JBoss to be manual and error prone, requiring significant labor to validate proper configuration. In an environment with a single cluster JBoss is very easy to configure for cluster operations, however, in an environment of any complexity (such as a QA cluster and a production cluster) it becomes more difficult to configure and manage.

Our experience with WebSphere, especially when configuring clusters using Central Installation Manager (CIM), is that node installation and cluster creation is a relatively straightforward process. The skills needed to effectively manage standalone servers carry over to WAS ND through the use of a consistent administrative user interface. Additionally, out-of-the-box settings provide sensible defaults and eliminate guess work when configuring environments of varying size and complexity

Security Administration

TCO Advantage: IBM WebSphere Application Server

This test case focused on the configuration of various security aspects of the administrative interfaces. The environment was configured to use Active Directory as an LDAP server. The test case required use of LDAP administrative groups and password management to control access to selected administrative functions. For JBoss EAP, the JMX console. The following list outlines key factors and findings:

- Administrative Security
    - Password protecting the JMX and embedded admin consoles is now a required step when deploying JBoss EAP 5. At a minimum they will need to be configured using username and password in two different configuration files (one file for the JMX console and one for the embedded admin console).
    - Securing administrative access in JBoss was more difficult and not as thorough compared to WAS and WAS ND. This is due to additional functionality provided by WAS and WAS ND which is not supported in JBoss.
    - An experienced JBoss administrator can secure the administrative console in approximately the same amount of time as an inexperienced WebSphere administrator can secure WAS and WAS ND. However, in our test scenarios, an inexperienced JBoss administrator took approximately six times as long to configure LDAP authentication for JBoss compared to WAS.
- No default cluster configuration security in JBoss EAP
    - Many security breaches originate from within a company. These breaches range from minor issues all the way through loss of data and complete network failure. Some forms of breaches can also result in the external exposure of private network resources. JBoss does not provide the capability of encrypted communications within the cluster and allows unauthorized instances of servers to be added to the cluster so long as an intruder knows the IP multicast address of the cluster.

    - The model employed by WAS ND can be configured to require secure communications between nodes in a cluster and the deployment manager and is enabled by default. New nodes cannot be added to the cluster except by the express consent of the administrator when using the federation tools provided with WAS ND.

- Default JBoss EAP configuration can lead to security vulnerabilities
    - **Automatic Cluster Formation** – As discussed elsewhere in this paper, JBoss will automatically form clusters when using the "*all*" or "*production*" configurations. This can lead to unanticipated results and potential security risks since extra steps must be taken to untangle clusters from each other.
    - **Farming Service** – By default each node in a JBoss cluster will participate in distributing applications deployed to the farm service to other nodes. This means that if someone were able to create a new node and add it to a cluster, they could both receive a copy of the deployed applications and their configurations, and deploy their own versions to the cluster.
    - **Multicast and Server Advertisements** – mod_cluster is the new JBoss service for connecting web servers and JBoss application servers. It is considered a replacement to mod_jk. By default it uses IP multicasting to bind web and application servers together. In its default configuration an application server will allow any mod_cluster configured web server it finds to process requests. Fortunately, there are ways to turn this off, but when doing so, we found that we had to bounce the web servers any time one of the bound application servers was restarted.
- No administrative roles for security in JBoss
    - WebSphere provides an administrative security model that allows role-based access control to administrative functions in both the web-based administrative interface and scripting language.
    - JBoss security is exclusively reliant upon operating system-managed, XML file-based security.
    - WAS and WAS ND uses several administrative security roles for performing different administrative tasks (Operator, Administrator, Monitor, Security Auditor, Deployer, Configurator, etc.) and also allow users to limit access to resources for different administrators (Peter can manage Application A, but not Application B). This feature enables system administrators to more finely control who can perform different functions within the realm of application server administration. JBoss does not have a similar concept.
    - JON does have support for roles and can map user roles to actions and resources. Because we were unable to get JON to work reliably, administrators will be forced to configure to manually edit files on the file system. This would negate the benefits of the roles enforced via JON.

Secure Audit of Administrative Actions

TCO Advantage: IBM WebSphere Application Server

In many organizations, an audit trail of all system configuration changes must be maintained. Because JBoss keeps all of its configuration information in a series of XML configuration files it is easy to make configuration changes outside the view of change tracking software. Changes must be manually recorded or reviewed, or a process would have to be developed to ensure that changes are being tracked.

Conversely, WAS can be configured to log all configuration changes made which can then be shipped to auditors for review. There is a special "Auditor" role in WAS that allows this capability. Users granted this role can view and modify the configuration settings for the security auditing subsystem. For example, a user with the auditor role can enable and disable the security auditing subsystem, set the audit policy, define which security events are to be audited. The auditor has full authority to read and modify properties of the security auditing subsystem. The administrator can view, but cannot modify auditing settings.

Deployment of Applications

TCO Advantage: IBM WebSphere Application Server

Secure and scripted control of application deployment is important in environments where operations and administrative staff maintain production environments separate from application developers and administrators. This test case focused on the tasks required to script the deployment of an application to a cluster, including resource definitions (JDBC and JMS). The following list outlines key factors and findings that affected the test case outcome:

- All JBoss configuration information is maintained in an array of XML files. Some files may be hot-deployed; others require a server restart. WAS and WAS ND also maintain server configuration as an array of XML files but expose access to them through both web-UI and command line-based administrative tools. Some changes require a restart of both application servers. With WAS ND, the web administrative user interface indicates when a server restart is required for each change.
- With JBoss, the administrator must manually distribute cluster-wide configuration changes or develop and test scripts to handle the distribution automatically. With WAS ND, administrative commands may be applied to an individual server, all servers on a node, or on a cluster-wide basis.
- WAS and WAS ND supports JACL and Jython-based versions of the interactive wsadmin tool. Changes can be very fine grained (e.g., just changing the password for a JDBC data source). With the Command Assistance feature of the IBM WebSphere Admin Console, the appropriate Jython commands can be accessed based on actions executed manually in the Administrative Console. This can be used to generate scripts which can be used for additional environments or deployments.
- Scripting for JBoss is typically done with Ant and shell scripts which assemble and move groups of files. The manner in which JBoss organizes information dictates granularity of how changes are secured and distributed. For example, when updating thread pool sizing for the JBoss web container we had to modify the jbossweb.sar/server.xml file. This file contains at least one setting which is specific to the name of the server when identifying cluster nodes. The level of scripting required to set the thread pool size while not affecting the name of the server is beyond that of a simple file copy, Ant or shell scripts and will require a more powerful scripting tool which is not provided by JBoss.

When using JBoss, the implementation of audit trail generation is up to the user's script development. JBoss EAP environment files are typically backed with a source control system such as Subversion; wsadmin scripts can be managed as part of promotion procedures between environments. With WAS and WAS ND, changes resulting from script execution or actions of admin console users are logged in the activity log file for the server.

While testing application deployment we came across some of the JBoss errors reported by other users in the public JBoss JIRA (see Appendix D). Those errors forced JVM, which in production environment would reduce system availability and increase cost.

Additional cost savings can be realized with the new WebSphere Cloudburst Appliance by automatically provisioning entire cluster configurations into the cloud of available resources managed by hypervisors in the matter of minutes. These additional savings are not included in this study. There are no comparable offerings from JBoss side.

Failover and Speed of Recovery

TCO Advantage: IBM WebSphere Application Server

Sooner or later things break. There are many types of failures: human errors, system overload, application defects, hardware failures, etc. While avoiding failures completely would be ideal, when they do happen, recovering from them quickly without significant loss of service is essential. This means that your application server must be able to transfer transactions logs, JMS queues and other critical information to backup machines in a cluster or perhaps to a completely different cluster. WAS provides very mature implementation of these capabilities via the High Availability Manager and provides a very fast data replication engine. For example, the typical failover time for all JMS destinations of a server is under 15 seconds when using the recommended HA configuration.

In our study, we tested the time for memory-to-memory replication to failover to another server between WebSphere and JBoss. Our tests for WebSphere indicated that failover would take less than two seconds to occur.

Our HTTP session replication failure tests used the following high-level steps for both WebSphere and JBoss:

1. Started multiple nodes within a cluster
2. Started a single web server
3. Launched a light number of virtual users using our load generator
4. Recycled each node in the cluster gracefully
5. Killed one of the application servers via "*kill -9*"
6. Watched log files on the surviving survivor and error rates in the test application
7. Manually restarted the dead server
8. Killed the other application server via "kill -9"
9. Manually restarted the dead server

Our findings with WebSphere are as follows:

- Using the admin console's Ripple Start feature, we expected each server to gracefully start and stop. Instead, the application server did an immediate stop followed by a start. This behavior was unexpected.
- Manually starting and stopping individual servers using the *stopServer.sh* and *startServer.sh* commands and via the admin console worked as expected. In flight requests finished and subsequent requests from the same session were routed to the other application server.
- Manually killing the application servers via the *kill -9* command resulted in a sub-second notification to the other server and subsequent requests were automatically switched to the surviving server. In one instance we saw a database transaction that was in flight on the killed server be rolled back on the surviving server due to a database deadlock because the dying server never committed the transaction. This behavior was the appropriate and expected behavior.
- The web server did not need to be restarted at any point during this test.

Our findings with JBoss were not as successful:

- When stopping the application server gracefully via the *shutdown.sh* script we found that the sessions transferred to the secondary server in a few seconds. The following is a summary of the events and steps taken to restore connectivity:
  - Unfortunately, mod_cluster continued attempting to send requests to the stopped application server.
  - When we started the application server again, we found that the web server was unable to bind with the application server due to an invalid signature being reported by the web server.
  - We then restarted the web server and operations resumed normally after the web server and application servers were able to rebind again.
  - Tweaking several of the session replication settings to improve session replication behavior did not have any noticeable results. For example, toggling the stickySessionForce option did

not have any effect on avoiding the need for a web server restart (the documentation implies that it might).

- When we forcefully stopped a server via the *kill -9* command results were somewhat unpredictable. In some instances sessions were not routed to the surviving server and session state was lost for those instances. This appears to be related to either mod_cluster and/or our session replication settings.

The biggest issue when testing session replication with JBoss was the need to restart the web server when bringing an application server back online. This would have resulted in noticeable customer outages even if we had a web farm consisting of multiple web servers since each web server would need to be restarted.

Our JMS failover tests used the following high-level steps for both WebSphere and JBoss:

1. Started multiple nodes within a cluster
2. Launched a test client to submit JMS messages to the queue
3. Recycled each node in the cluster gracefully
4. Killed one of the application servers via "*kill -9*"
5. Watched log files on the surviving survivor and error rates in the test application
6. Manually restarted the dead server
7. Killed the other application server via "kill -9"
8. Manually restarted the dead server

Our findings with WebSphere are as follows:

- Using the admin console's Ripple Start feature, we expected each server to gracefully start and stop. Instead, the application server did an immediate stop followed by a start. This behavior was unexpected.
- Manually starting and stopping individual servers using the *stopServer.sh* and *startServer.sh* commands and via the admin console worked as expected. Transactions failed over to the other bus members gracefully
- Manually killing the application servers via the *kill -9* commands resulted in a notification to the other bus members and subsequent requests were automatically switched to the surviving bus member.

Similar to our HTTP session replication findings, our tests with JBoss were not as successful:

- When stopping the application server gracefully via the *shutdown.sh* script we found that requests transferred to the secondary server within a few seconds.
- After stopping the second server and then starting the first server again we began to see errors in the client. After some research, it appears that we encountered issue JBMESSAGING-1743.[7]

In summary, despite JBoss's inclination to automatically form clusters, its clustered operations is not as reliable as WebSphere.

Server Monitoring

TCO Advantage: IBM WebSphere Application Server

As part of our standalone server testing we also evaluated the embedded admin console (aka embedded-JOPR) which is new to JBoss EAP 5. Our testing uncovered several issues described below:

- As part of our testing, we attempted to deploy an application (EAR file) with bad deployment information to test troubleshooting. The admin console reported that there was an error deploying the

[7] https://jira.jboss.org/browse/JBMESSAGING-1743 and http://community.jboss.org/thread/146514

application, however, it failed to rollback the installation and the previous good application was no longer available to serve user requests. Subsequent attempts to deploy the application with a correct deployment descriptor resulted in an error indicating that an application with that name already existed, even though the application did not show up in the list of deployed applications. After rebooting the application server, the application appeared in the list of deployed applications and it was in an error state.

- WebSphere does not exhibit the same behavior with failed application installations or updates. A failed installation is rolled back. If this is a new install, no traces of the failed installation were found. In the case of an application update, the failed installation was successfully rolled back, and the last successful deployment was still available and working properly.
- Many administration tasks within JBoss still require manual modifications to one or more configuration files on the app server's file system. For instance, increasing thread pools for the web container or configuring the web server integration still requires an update to an XML configuration file. Direct editing of XML files not only has the potential to introduce typos, leading to configuration errors, but also represents a security exposure because administrative access cannot be segregated between different roles and no auditing of changes is possible.

As part of the performance tuning exercise we evaluated the built-in capabilities of each tool. For JBoss we utilized the built in JMX console and for WebSphere we used the PMI capabilities and Tivoli Performance Viewer embedded in the admin console. Here is the summary of that evaluation:

- The JBoss JMX console contains quite a few options, and uses cryptic naming conventions for the JMX MBeans. Knowing which items to track requires a deeper understanding of the underlying JBoss architecture and the components involved. It is not currently possible to track multiple values from different resources. For example, tracking database connection pool size, and current number of web container threads requires multiple browser windows or manually navigating between the pages.
- For WebSphere, the Tivoli Performance Viewer (TPV) user interface is non-intuitive, but does have the ability to track multiple resources at one time. The TPV user interface would do well with an RIA (Rich Internet Application) front-end to eliminate the constant IFrame refreshes currently employed.

WebSphere provides an embedded Tivoli Performance and Diagnostic Advisor. It provides advice to help tune systems for optimal performance and provide recommendations on inefficient settings by using data collected via the Performance Monitoring Infrastructure (PMI). For example, the Advisor can suggest optimal settings for HTTPSession cache size, or JDBC connection pool settings, JVM heap size and many other settings based on the analysis and comparison of the current vs. past workload and the state of the system[8].In addition to testing the JMX console and the new embedded admin console we also tested JBoss Operations Network (JON). Here is a summary of that evaluation:

- JON is slow in responding to operations. For example, adding a script for JBoss took about five minutes for the server to respond. Looking through another instance indicated that the script was created and available, even though the executing JON server indicated a pending status.
- Status tracking in JON is an issue. 13 minutes after executing the *run.sh* script to start a server, the console showed a status of "*In Progress*". 15 minutes after the script was run the status changed to "*Failure*" with an error message of "*time out*". Manually checking the list of running processes on the physical server indicated that the server had been successfully started.
- Statuses of JBoss servers are not accurate in JON. All JBoss instances in our cluster were reported to be "*down*" when they were in fact up and running.

8

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/cprf_choosingperfadvisor.html

- JON failed to stop a running instance of JBoss EAP. The response was that the server was already shutdown. The server to be stopped continued to run.
- After the JON session timed out, attempting to log back in failed with an error message stating, "*the backend data source is unavailable*".

Backwards Compatibility for Applications

TCO Advantage: IBM WebSphere Application Server

Beyond the server upgrade and its additional installation and configuration costs, there may be development costs associated with an upgrade. JBoss has not provided the same level of backward compatibility as WAS and WAS ND for applications. IBM provides commitment to an N-2 release support for product features. JBoss does not make a public statement about backwards compatibility in its documentation and the JBoss 5 release notes indicate several areas where backwards compatibility has been broken. As part of the 2009 study we upgraded both the WAS and JBoss environments. The results were compelling in that the J2EE 1.4 version of the test application used in the previous analysis (WAS V6.x versus JBoss 4.x) was able to run with no modification or configuration changes in WAS 7 while the exact same application would no longer run in JBoss AS 5.0.

When verifying the JBoss AS 5.0 application against JBoss AS 5.1 (a component of JBoss EAP 5) we found that compatibility was broken between these two versions. We traced the issue back to a bug in the EJB3 container which was subsequently fixed in the JBoss EAP 5 released in November of 2009.

When migrating from v7 of WebSphere to fix pack 11 (v7.0.0.11), no issues were encountered with the application as a result of the upgrade process.

Server Upgrade and Migration

TCO Advantage: IBM WebSphere Application Server

Our study is based on the assumption that there will be one major version upgrade of the entire environment over the five-year evaluation period. For instance, in year 0 of the study, version 7 of WAS will be installed and in year 3, the servers will be upgraded to version 8. For the cost analysis portion of the study, we used the results of our upgrade scenarios from v6.1 to 7 of WAS and for the migration of JBoss 4 to JBoss 5.

In our scenario, we upgraded from JBoss 4 to JBoss 5, using JBoss's migration approach of installing and configuring a new server and manually redeploying all applications to the new server. This included the manual migration of data sources and other JEE resources. All of these configuration files had to be tracked and configured manually. Also, combined with some of the backwards compatibility issues encountered, the upgrade path from JBoss 4 to JBoss 5 may be more difficult. For an organization with a small number of application servers this may be a minor issue, but any organization with more than a few servers will find this to be time consuming and potentially error prone. By comparison, the WAS installation media includes a migration tool that will export the settings from a previous version of WebSphere Application Server.

This test focused on upgrading and migrating an application from the previous release to the current server release. This involved migrating the legacy J2EE 1.4 version of the DayTrader application from WAS and WAS ND 6.1 to version 7, and migrating from JBoss 4 to JBoss 5.

Deployment of the J2EE 1.4 version of DayTrader on WAS ND used the built-in migration wizard. The application worked without modification or reconfiguration of JDBC or JMS resources.

Deployment of the J2EE 1.4 version of DayTrader failed to execute on JBoss 5. JBoss does not provide a migration wizard; this means configuration settings and deployed applications must be manually copied from several XML files in the previous installation to the new XML files. The configuration was validated against the JBoss 4 environment to ensure that the configuration settings and application were the same. In essence, this means deploy and debug cycles must be repeated for every application. For larger installations this could become quite time consuming.

This same concept applies to any new version of JBoss – sometimes within the same major revision number. Since the JBoss install process does not upgrade the previous configuration, for most major upgrades you would need to migrate applications manually as described above.

Hot fixes are handled differently between WebSphere and JBoss. Hot fixes, or interim fixes, for WebSphere are typically installed using the Update Installer utility. Whereas, JBoss hot fixes must be manually applied to the server. Backing out a hot fix within WebSphere is again done via the Update Installer utility; for JBoss this must be performed manually. During initial application verification, we attempted to install a number of hot fixes for JBoss with negative results. The application server itself, failed to start after applying hot fixes using the documentation provided by JBoss for each hot fix.

Problem Determination and Troubleshooting

TCO Advantage: IBM WebSphere Application Server

Using the same tooling, the process of configuring the test application to successfully deploy on JBoss took five times longer than it did on WAS. The extra time required to configure the application for use on JBoss was due to inaccurate or missing documentation, as well as the lack of problem support and diagnostic tools. Troubleshooting issues with JBoss required trial-and-error based on incomplete or incorrect information scattered throughout official documentation, forums, and blogs. Code changes were required due to bugs in JBoss's implementation of the EJB3 specification, and issues with the underlying Hibernate dialect for IBM DB2®. It is important to note, that the time to fix the DB2 dialect issue is not included in the time required to bring the application up on JBoss, since presumably a different database platform may not have had the same issue.

In contrast, when configuring the application for WAS and WAS ND, no code modifications were required, and issues encountered when configuring the application were readily identifiable using IBM's InfoCenter and problem determination tools.[9]

Through the course of verifying the test application for both JBoss and WAS we also found that significant improvements have been made to WAS and WAS ND from previous versions. No restarts of the application server were required during the application server configuration process, such as configuring JDBC providers and data sources as well as JMS resources. In previous versions of WAS, frequent restarts were required and were quite time consuming.

JBoss required frequent restarts during the configuration process. In some cases, these restarts were to allow configuration changes to take effect. During our testing we also tested the effects of introducing mistakes into the manually edited configuration files. In several of these tests, we had to stop the server, delete temporary directories and redeploy the application prior to starting the server again. The following table summarizes the list of situations where we found a server restart to be necessary:

Action	**Requires Restart?**	
	WebSphere	**JBoss**
Change JDBC provider classpath	Y	Y
Adjusting JDBC data source pool size (cell-level)	Y	Y
Adjusting JDBC data source pool size (node or server-level)	N	Y
Adjusting JDBC data source settings (pool size, connection string, username, password, etc.)	N	Y
Adding a new JMS bus member or node	Y	N

[9] IBM WebSphere Application Server InfoCenter, http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/welcome_nd.html.

Changing JMS configuration	N	Y
Re-deploying an application after a failed deployment	N	Y
Binding a web server to an application server	N	Y

Restarts of JBoss took between 60 and 90 seconds depending on whether the test application was deployed. Restarts of WAS 7 were consistently 30 seconds regardless of whether the test application was deployed or not. Deployment of the application was consistently 10 seconds on WAS while JBoss varied between 15 and 30 seconds per deployment.

We also tested the effects of common configuration mistakes, such as the misspelling of JNDI names in EJB references. This test on JBoss resulted in a NullPointerException being thrown during the deployment process. The exact cause of the error was not readily identifiable because of the terse nature of the error reported. Using online support forums for JBoss did not help with resolving this issue. If paid support were available, this could require frequent problem support requests and potentially lead to lost productivity while waiting for responses. Conversely, WAS provided an error code which could be used to determine the cause of the issue.

Troubleshooting WebSphere was much simpler than troubleshooting JBoss. Most issues encountered with WebSphere configurations include an error code which can be looked up online and pointers to the appropriate resolution can be found rather quickly. On the other hand, JBoss favors the use of Java's exception handling mechanism for reporting errors. Unfortunately, in many cases searching for NullPointerException and various pieces of the stack trace or surrounding messages would yield marginally useful results.

Serviceability and Documentation

TCO Advantage: IBM WebSphere Application Server

Logging, tracing and other diagnostics can at times be challenging, especially in complex environments. JBoss EAP does not provide a mechanism to access error information or log files outside of the file system. WAS and WAS ND have the ability to access log files without file system access. Administrative commands and the admin console provide trace analysis tools and error reports upon failure. It was our experience that the error reporting for WebSphere was more consistent, thorough and accessible than JBoss errors.

In JBoss 4, the help available in various online support forums and mailing lists was fairly mature. JBoss 5, however, was rewritten to use the new micro-container architecture. This architectural shift calls into question the efficacy of much of the JBoss 4 documentation being applied to JBoss EAP 5. Through our testing, we found many errors that would cause the server to fail to start. We also found that documentation became increasingly unreliable when being applied to complex configurations or production-level configuration and security hardening. In our own testing we noticed that performing root cause analysis for some issues was difficult due to the number of configuration changes required to solve issues due to the time and effort to painstakingly track and validate each change made.

We also found that many JBoss forum postings and wiki articles provided conflicting information. In some cases, this was due to the long release cycle for JBoss 5 and the number of defects that were opened and closed during the development process. Unfortunately, when verifying whether these proposed solutions would alleviate our issues, we generally either uncovered additional issues or found that the solution did not meet our needs.

A general trend we observed with the JBoss documentation was its slant towards developers versus administrators. For example, many of the quick start guides were geared at quickly setting up a single/local development environment as opposed to configuring servers for use in multi-server environments.

Management of Large Topologies

TCO Advantage: IBM WebSphere Application Server

WAS has a unique capability called Flexible Management that allows you to submit administrative jobs asynchronously for application servers registered to administrative agents and for deployment managers. Jobs can be submitted to one or more servers, including geographically dispersed servers. The administrative job manager can queue administrative jobs directed at the standalone application server nodes or clustered domains. The job manager can asynchronously administer job submissions and can complete the following tasks:

- Set the job submission to take effect or expire at a specified time.
- Specify that the job submission occur at a specified time interval.
- Notify the administrator through e-mail that the job has completed.

JBoss does not have comparable functionality. This functionality could reduce off-hours work required by administrators or could be used to avoid potentially expensive site visits.

While this study focused on the Medium and Large scenarios described above, here are several Very Large scenarios and hypothetical company examples of situations where a job manager would be useful. Please note that we did not include this capability into our calculations of the cost, but for some companies this might be a significant factor:

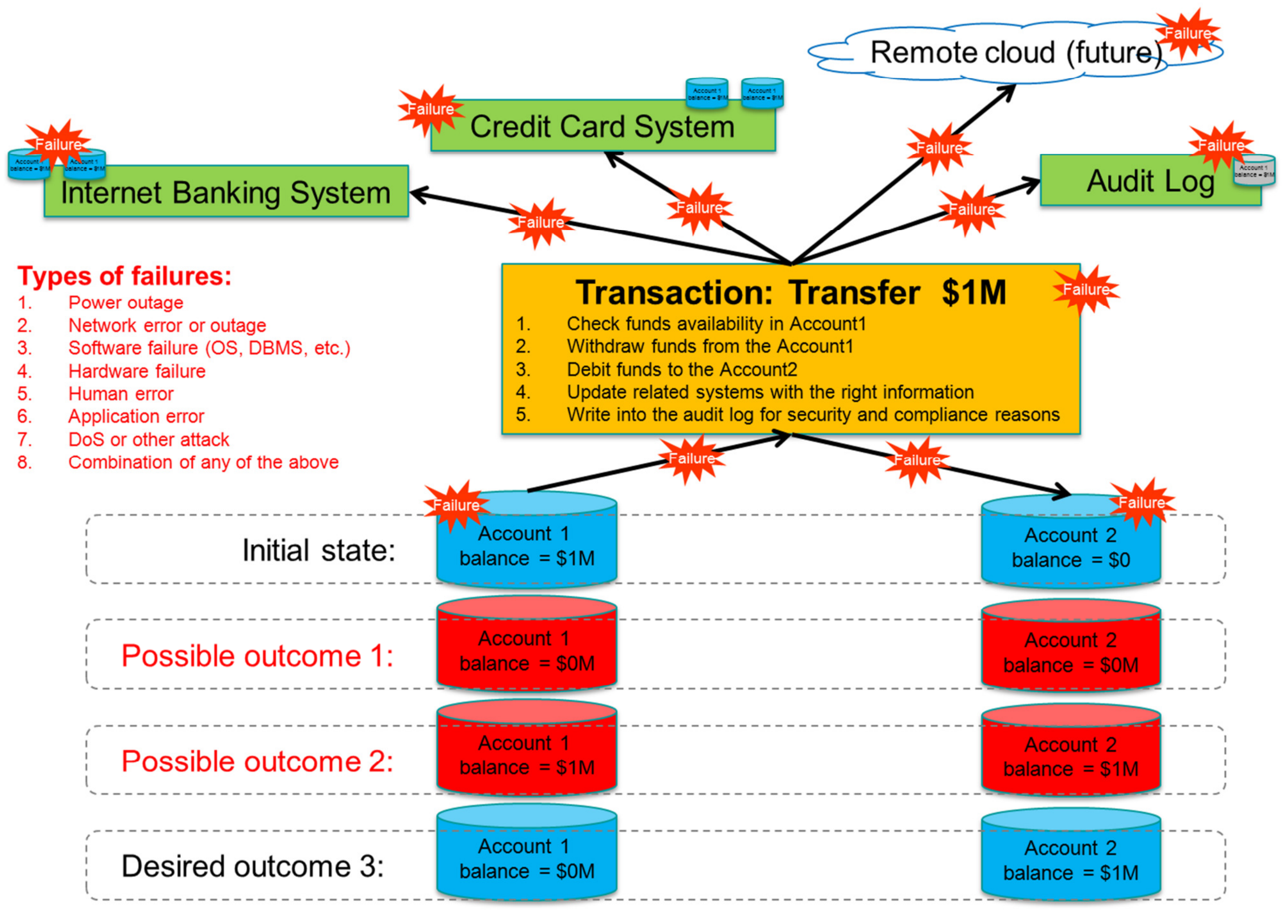
- Branch office environment
  - A business has a thousand stores geographically dispersed across the continent. Each store contains either a few application servers, or a small Network Deployment cell consisting of two or three machines. Each store is managed locally for daily operations. However, each store is also connected to the data center at the company headquarters, potentially thousands of miles away. Some connections to the headquarters are at modem speeds. The headquarters uses the job manager to periodically submit administrative jobs for the stores.
- Environment consisting of hundreds of application servers
  - An administrator sets up hundreds of low-cost machines running identical clones of an application server. Each application server node, which is registered with an administrative agent, is registered with the job manager. The administrator uses the job manager to aggregate administration commands across all the application servers, for example, to create a new server, or to install or update an application.
- Environment consisting of dozens of deployment manager cells
  - An administrator sets up hundreds of application servers, which are divided into thirty different groups. Each group is configured within a cell. The cells are geographically distributed over five regions, consisting of three to seven cells per region. Each cell is used to support one to fifteen member institutions, with a total of 230 institutions supported. Each cell contains approximately thirty applications, each running on a highly-available cluster of two for failover purposes, resulting in a total of 1800 application servers. The administrator uses the job manager to aggregate administration commands across all the cells, for example, to start and stop servers, or to install or update an application.

Based on the findings in our hands-on testing, managing JBoss in these kinds of environments may become very difficult without building a custom home grown management framework similar to what is discussed above.

Transactional Integrity

TCO Advantage: IBM WebSphere Application Server

Poor quality data will cost the banking industry 27 billion USD in operating costs[10]. Same study suggests that half of customers will give their banks only two times to make a mistake before considering a change to another bank. The diagram below illustrates a simplified version of the money transfer transaction. In most banks Core Banking System is a separate application from the Internet Banking, Credit Card System and several other systems not shown here. The "money transfer" operation must be able to reliably update all of the related systems either as a single unit of work (XA) or long running transaction.



Transactional integrity is important when multiple related steps need to be completed for a successful transaction. For example, what would happen if a customer were to initiate a funds transfer between two accounts only to have a server crash in the middle of the transaction? Ideally, the system would be able to automatically detect the failure and take the appropriate action. Another potential solution would be to rollback the transaction and allow the user to try again. If not handled appropriately though, valuable transaction data, which could include your customer's money could be lost or recorded incorrectly. A transaction failure in a critical system could cost a company millions of dollars in lost revenue and lost customer satisfaction. As a result of this risk, it is essential that applications handle these transactions correctly and ensure that enterprise data accessed during business transactions is maintained in a reliable and consistent state, regardless of any system or business failure that may occur.

WebSphere Application Server version 7 provides highly effective transaction engine, which was built on top of a solid foundation from the time of its introduction over ten years ago. It is designed to automatically detect and recover from transaction problems that occur during two-phase commits. WebSphere Application Server

[10] http://www.ibm.com/smarterplanet/us/en/banking_technology/ideas/?&re=spf

V7 includes powerful tools, such as transaction log viewers and easy steps for specifying transaction recovery settings. These capabilities help ensure transaction integrity for WebSphere Application Server users.

JBoss seems to have a design flaw that prevents its transaction engine from correctly recovering in-doubt transactions. The documented issue is in the JIRA JBoss document defect (JBAS-7965[11] and JBAS-6244[12]). This defect describes a JBoss AS transactional integrity recovery issue, but does not specifically address how to fix the problem. This raises the question as to whether JBoss can automatically recover from all transaction integrity failures. Moreover, there appears to be no documentation on how to configure or fix JBoss to automatically recover from the transactional integrity issues when in-doubt distributed transactions fail. Instead, there seem to be forum postings showing the specific transaction recovery problem with the JBoss Application Server, or discussions about the transaction recovery problem that don't address how to actually fix it.

IBM has done a study [13] based on this public information to attempt to determine whether the transactional integrity problem in JBoss can be recreated or if it has been fixed or resolved. [14] It appears that the problem still exists in JBoss EAP version 5.0 and the JIRA records are still in the state "Critical-Unresolved". This means that XA transactions with databases, messaging systems, adapters and all other resource managers cannot be reliably recovered by JBoss if the failure happened during the second phase of the 2PC protocol. This leads to a loss of data or significant labor costs to manually analyze and resolve every failed instance. It shall be noted that manual recovery is time consuming, very costly and can cause additional errors - it is manual after all.

Training Costs

TCO Advantage: IBM WebSphere Application Server

The available training from IBM for WAS 7 and Red Hat for JBoss EAP 5 is similarly divided in focus between application development and administration. The TCO analysis includes a base assumption that training for administrators and developers will be provided at the beginning of the five-year analysis period and again half-way through the five-year period to account for turnover, growth, or major upgrades that may occur.

The quantity and skill level of administrators is based on the results of the hands-on analysis. For the purpose of this study we used our study team size (three administrators). It is important to note that for larger JBoss environments, training costs could be higher due to the performance difference and the associated increase in the number of servers required.

In our study, the amount of training costs being factored into the TCO calculation over a five-year period is $46,875 for WAS ND training vs. $131,528 for JBoss EAP, a difference of 181%.

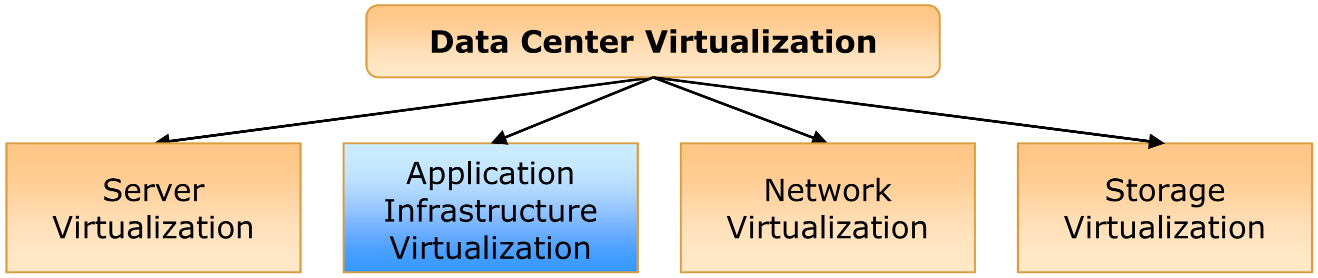
[11] https://jira.jboss.org/browse/JBAS-6244
[12] https://jira.jboss.org/browse/JBAS-7965
[13] http://www.youtube.com/watch?v=UzWVXRoe15Y
[14] ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/wsw14068usen/WSW14068USEN.PDF

Costs not Included in the Study

Our detailed testing described above attempts to cover the major quantifiable aspects of TCO that an organization may encounter in its evaluation of application servers. However, it does not cover all possible factors that affect TCO. Through our testing we uncovered a number of qualitative factors that may have some impact on TCO and the application server vendor selection process. How these factors are measured and affect the TCO for two different organizations may vary widely, therefore **we did not include these findings below in the formulas used to calculate the TCO results**. It is feasible that for certain organizations these additional considerations described below when taken together may well exceed the cost considerations discussed in the previous sections. Organizations that require high quality of services from their infrastructure should do a cost/benefit analysis of the capabilities discussed below.

Application Virtualization and Large Data Center Installations

Application infrastructure virtualization complements server, storage and network virtualization. It is a fourth category of virtualization in the data center (see figure below) that can enable your business to push the boundaries of its IT infrastructure further for greater agility, cost savings, operational efficiency, economy and manageability.


Data Center Virtualization
Server Virtualization
Application Infrastructure Virtualization
Network Virtualization
Storage Virtualization

WebSphere Virtual Enterprise's (WVE) potential impact on TCO was not considered in this study. WVE provides application infrastructure virtualization capabilities that lower operational and energy costs to create, run, and manage your enterprise applications and service oriented architecture (SOA) environment within a private cloud. Furthermore, administrators can deploy and utilize resources quickly and seamlessly during peak periods and in response to the peaks and valleys of unforeseen demand in processing for various mission-critical applications. Administrators can also enforce policies to help make sure application response times and service levels meet service level agreements. WVE can help extend the benefits of server virtualization using an approach that can address potential issues and limitations inherent in some applications. WebSphere Virtual Enterprise can be thought of as a hypervisor for application servers. In addition, it enables server virtualization and application infrastructure virtualization to be combined so that you can take full advantage of the strengths of both approaches.

JBoss does not have comparable functionality for supporting very large environments. The hardware, license costs and administrative labor savings result in even bigger WebSphere TCO advantage over JBoss.

Cloud Support

According to an IBM study, WebSphere Cloudburst Appliance can reduce software labor hours by up to 80% compared to manual deployment (see references). JBoss does not provide similar capability.

While virtualization and standardization can go a long way in reducing overall labor costs, the task of deploying a software stack as a VM image onto a virtualized server has historically been a highly labor-intensive task. For instance, one has to first deploy and configure the OS along with all requisite patches. After that, the administrator has to install and configure the application server and all its constituent components (e.g. HTTP server, etc.) as well as patches and other fixes. For applications requiring a database, that becomes yet another piece of middleware that needs to be installed and configured. Then there is the application itself. Collectively, deploying and testing a complete application manually can require days or weeks to accomplish depending upon its overall complexity. In a private cloud environment, this kind of turnaround is untenable.

WebSphere Cloudburst Appliance (WCA) is specifically designed to address this problem. Available as a hardware appliance, it takes 10+ years of best practices in WebSphere Application Server (WAS) deployments and encapsulates it into pre-defined, customizable images that can be dispensed to a variety of hypervisors used in virtualized servers. Its use of scripting and automation techniques greatly reduces the labor required to perform deployment tasks. WebSphere Cloudburst Appliance works very well with WebSphere Virtual Enterprise and both can provide significant value to WebSphere customers.

Software Development Costs

A commonly held belief is that open source solutions more quickly integrate new standards and functionality. Over the past three years of conducting this study, we have found that this is no longer the case. In fact, with regards to WebSphere and JBoss, this appears to have reversed. With WebSphere 6.1, IBM introduced the concept of a pluggable architecture via the use of free feature packs. These feature packs allow organizations to take advantage of new standards quicker without having to wait for the next full release of WebSphere.

The version 5 release of JBoss marks their entry into a pluggable microcontainer architecture that will allow them to compose different server configurations and potentially adapt to standards changes in a faster manner. Furthermore, while JBoss was the first application server to support portions of the EJB3 specification, it was behind both IBM and Oracle in passing compatibility tests for JEE5.

Quality and Availability of Vendor Support

Quality of vendor support can have a direct impact on the cost of operations.

IBM provides local language support in many countries in local hours. IBM offers several levels of support ranging from 9x5 (local time) through 24x7 support. In addition to paid support, IBM offers the InfoCenter. Over the past few years the quality and accuracy of the documentation available in InfoCenter has improved. Finally, IBM has settled on a quarterly release schedule for WebSphere Application Server fix packs and interim and hot fixes are still made available on an as needed basis.

Red Hat provides support options of either 12x5 or 24x7, but for many locations across the world there is no support in native language. Documentation is available online, however, as has been noted throughout this paper, it is greatly lacking. With regards to fix packs, it appears that Red Hat has not settled upon a fixed period for maintenance releases and hot fixes are released as needed.

Conclusions

Comparisons of WebSphere and JBoss frequently focus on an application development point of view alone – only looking at feature sets and initial acquisition costs. Unfortunately, the long-term administration costs associated with JBoss are not factored into the equation. Too often, the "free" initial costs lead developers to favor JBoss. Our experience and the analysis from this study reveal that there are many more factors for organizations to consider in their selection process beyond initial acquisition costs.

Our projections indicate that as the size and complexity of a deployment and the supporting organization grows, the bulk of costs shift from product acquisition to administration and operational activities. In addition to the higher skill levels required for JBoss EAP administration there proved to be significant functional shortcomings when JBoss is used in a production enterprise application environment – due both to its lack of important product capabilities as well as the difficulties associated with its administration.

JBoss EAP 5 lacks critical functionality in many areas such as security, administration, backwards compatibility, and cluster reliability. When we examined other factors including system administration, software maintenance, ease of integration and development costs we found that the products take very different approaches that significantly impact TCO. JBoss 5 also has significant shortcomings in the areas of available support and reliable documentation that hinder development and system administration. On the other hand, WAS 7 appears to be much more stable and has continued to build upon the lessons learned from previous versions of WAS.

After a lengthy development and public beta testing cycle, JBoss EAP 5 was released in 2009. Many of the same issues encountered in previous versions of this study still exist in this version, and very few new features were added to the product. Meanwhile, IBM has continued to add new features and functionality to WAS increasing its value to customers.

IBM has committed resources to ensuring a migration path from previous versions to the latest versions of WAS and providing migration tools to assist in the process. JBoss has no official statement regarding support for backwards compatibility and in some cases, applications that worked in JBoss 4 will no longer work in JBoss 5.

While WebSphere Application Server may have higher initial acquisition costs, over time it makes up the difference by having lower administration costs, better performance, better reliability, and less overall risk.

The bottom line is that even though an open source application server may appear to be free on the surface, it is in fact not free. Organizations should carefully examine the full costs and risks associated with an application server and choose a product based on TCO in the context of their staff capabilities, applications, environments and overall risk posture.

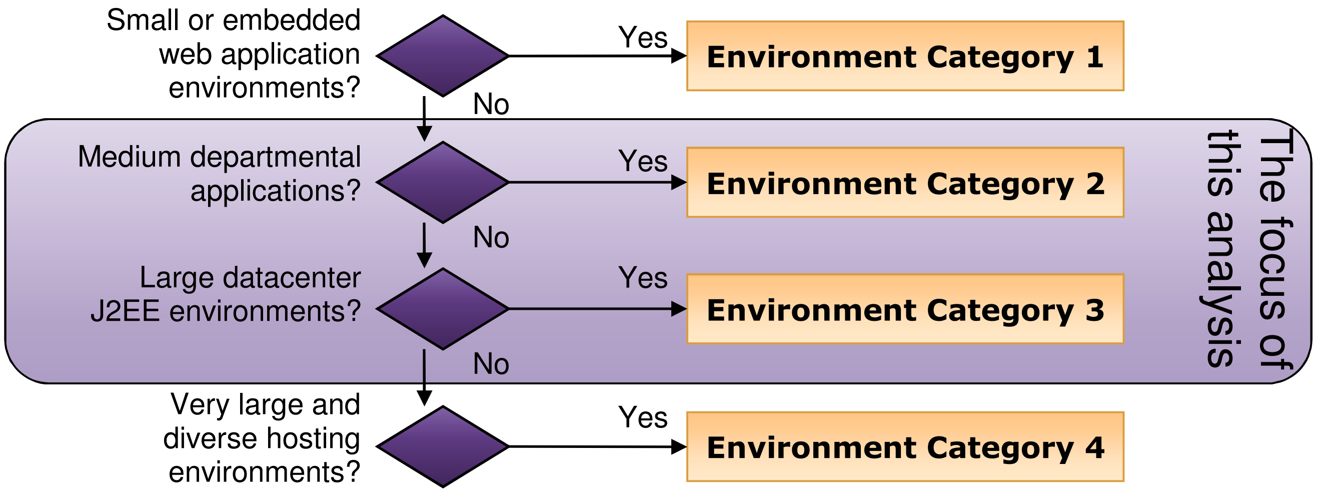
About this Study

The study was commissioned by IBM and performed by Summa Technologies. All findings of this study are fully backed by the solid reputation of Summa. Since 1996, Summa has been architecting and implementing commercial-grade applications for organizations of all sizes, including Fortune 100 companies. Summa's IT consulting services help companies evaluate and implement modernization strategies for n-tier distributed systems. Summa helps their customers make the right technology investments by first assessing the current environment, and then recommending and implementing the technology solutions that improve customer satisfaction, employee productivity and partner relationships. Authors of this report collectively have over 35 years of experience building and supporting large-scale systems.

For more information about Summa, please visit us at http://www.summa-tech.com.

Appendix A – Analysis Methodology

This section provides an overview of the methodology used to perform the analysis for this report. The TCO analysis and model provided both quantitative and qualitative comparison of JBoss AS and WAS ND based on a combination of test experiences, product research and the execution of a series of planned validation tests. Application server middleware is used to support a broad range of environments and application types. To perform our analysis, we first looked at characteristics of a set of environment classifications and organized the analysis into four categories.


Small or embedded web application environments?
Yes
Environment Category 1
No
Medium departmental applications?
Yes
Environment Category 2
No
Large datacenter J2EE environments?
Yes
Environment Category 3
No
Very large and diverse hosting environments?
Yes
Environment Category 4
The focus of this analysis

Category one - Small environments may host small non-critical applications, or packaged software products that require an embedded web application server. These applications may require less security, lower availability and a frequency of administration that is small compared to other categories. Features such as clustering and session replication for load balancing or fail-over are not required. These applications are typically used in very small organizations.

Category two - Medium is focused on departmental applications in medium and large sized companies. The applications are smaller in size compared to categories three and four, but may require some combination of security, clustering, or more frequent administration.

Category three - Large is focused on the large middle ground of web-application environments. These environments have a range of six to twenty servers; many require clustering for reliability, scalability, availability and performance. These environments also demand more comprehensive treatment of security than category one because of application criticality and exposure. Common attributes of applications/deployments in this environment include:

- Hosting of multiple independent and integrated applications;
- A need for moderate to high levels of security for both infrastructure and applications;
- High-availability requirements driving the need for clustering of services;
- Division of responsibilities between developers and administrators for either regulatory compliance (e.g., Sarbanes-Oxley) reasons, efficiency or other organizational reasons;
- Solid change management policies to manage software and server configuration promotion from development to testing and production environments;
- Sample applications in this category include moderately sized web banking and other financial services systems, insurance claims processing systems, and manufacturing process management systems;

Category four – Very Large environments are classified as very large, clustered, and highly available deployments. Example applications in this category may include very high-volume consumer-facing web applications or very large integrated online financial applications. Due to the need to support complex change management monitoring requirements for diverse technologies, these environments benefit from virtualization technologies.

The bulk of the analysis in this paper is focused on factors that impact category two and three environments. This focus was chosen for several reasons. Environment category one is a reasonable target for OSS, yet depending on the level of support purchased from the vendor WAS Base might be more cost efficient than JBoss. Category four is clearly a target for highly scalable commercial software, and therefore the detailed analysis of TCO is not as valuable. The gap between TCO impact and the typical understanding of the full range and magnitude of its factors is most significant in environments in the second and third categories.

The outline below highlights key elements of the analysis approach so that readers may understand the choices made in the selection of TCO criteria and results.

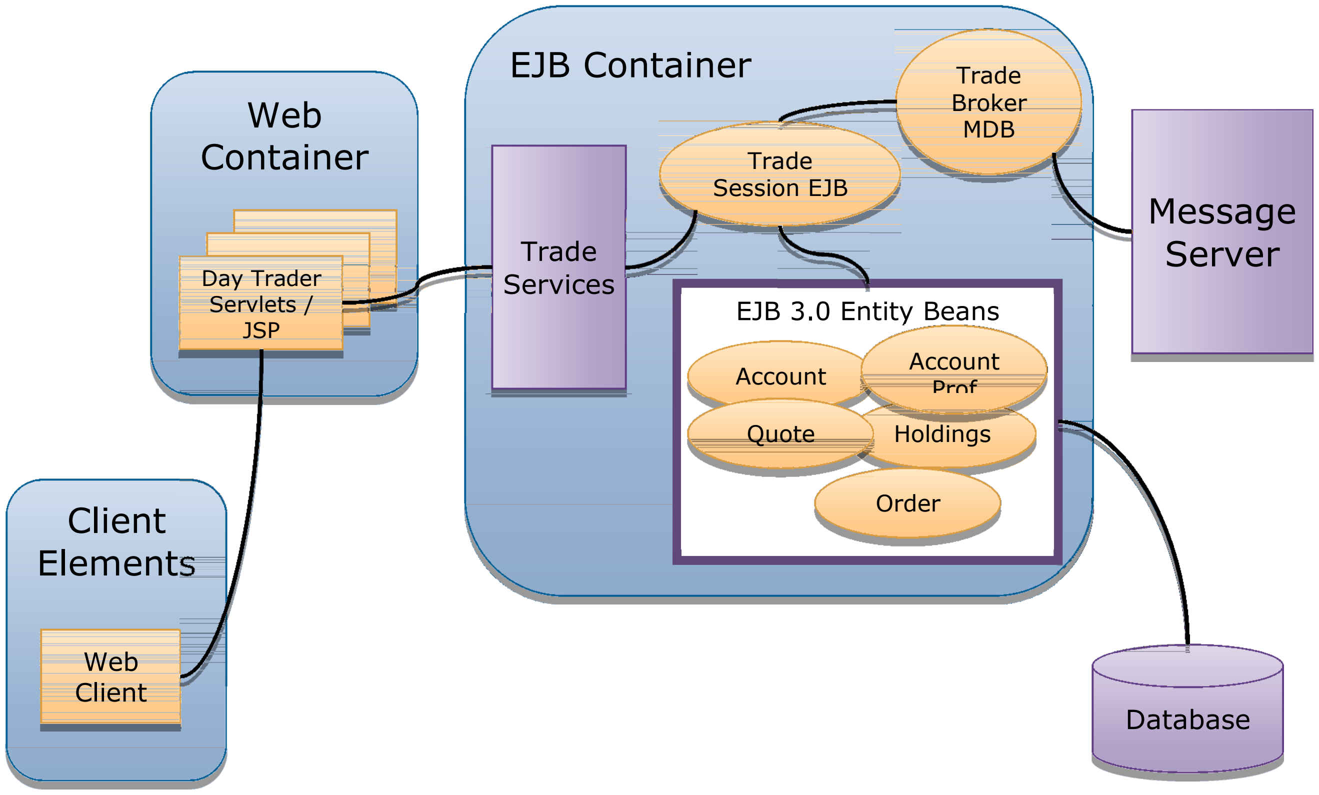
- A list of TCO factors for application server use was established against common architectural, deployment and operational requirements.
- From this input, the four high-level usage categories scenarios were established for comparison purposes. The categories segment analysis across environments that have very different weighting of TCO factors.
- Following the scenario and environment definitions, a set of test cases were developed and reviewed. The test cases defined configuration requirements, goals and scripts for the set of actions to be performed and measured against real-world issues and system configuration requirements for each product. The following list outlines the test case coverage:
    - Installation and Configuration
    - Administrative Security Configuration
    - LDAP Integration
    - Cluster Configuration
    - Clustering Security
    - Extending a Cluster
    - Transaction Logging and XA Configuration
    - Configuring JMS Persistence
    - Automated Deployment of an Application to a Cluster
    - Troubleshooting Clustered Application Issues
    - Application Server Memory Utilization
    - Long Running Stress Test
    - Managing Administration Audit Trail
- The tests were performed by timing operations on both application server environments. Activities included planning, implementation and validation of the execution steps for each test case. Details on the specific operations performed, issues encountered, issue resolution approach and failure cases were timed and documented. We separately measured and recorded time for:
    - Initial time to research and define options and approach
    - Time to perform the initial implementation
    - Time to perform subsequent operations
- In many cases, operations within the same test require different skill levels. For the evaluation, we tracked required skill levels against the Usenix SAGE special interest group defined system administration levels [SAGE 01]**.** Costs for administrators were associated with skill levels using supporting salary survey per skill level.
- The model derives TCO calculations based on the number of times operations are performed over the TCO analysis time. The values are based on environment size factors such as
    - Number of application servers, HTTP servers, LDAP servers, WLM servers, cache servers
    - Number of clusters
    - Number of applications and their complexity and frequency of updates
    - Number of administrators
    - Number of software developers

Appendix B – Test Application

In addition to establishing a representative server environment, a realistic test application was required. To evaluate the TCO, the freely available Apache Geronimo project DayTrader application was used. Reasons that the DayTrader application was selected include:

- Use of a significant set of JEE5 services, including JDBC, JMS, EJBs, Servlets/JSPs, JTS/JTA.
- Enterprise-grade services requirements including security, performance, clustering and transactions.
- Support of multiple databases and JMS environments.
- Performance in a clustered application server environment.
- Migration path from J2EE 1.4 application to JEE5.

The following diagram provides an overview of the DayTrader JEE application architecture.



DayTrader does not include web services capabilities. To test web services performance we have developed our own application, which uses moderately complex web service implemented as a JAX-WS 2.0 annotated POJO hosted by the application server. The development was done bottom up - first main POJO and all dependent POJO classes were developed and then main POJO class was annotated with @WebService. The business method used some of the input data to generate semi-random output. No backend services were called during the execution of this web service. The data classes used in the web service are representative of a real world application commonly found in financial sector. Total there are 30 inter-related POJOs that together comprise the payload of the input and output logic for the web service. Combined these 30 classes have over 150 primitive data fields - almost every primitive data type used in Java. Certain data types are arranged as arrays and appear in the input XML payload multiple times. The input workload for the web service performance test included SOAP/HTTP requests varied in sizes and included requests and responses of 1K, 2K, 10K, 110K.

Appendix C – Test Environment

To perform the analysis, a diverse and realistic environment was established for testing. The components of the test environment were selected based on common combinations encountered in real-world environments.

The technical environments for testing used the same hardware and operating system configuration for both environments to ensure that environmental differences were not a factor in TCO measurements. WAS ND, and JBoss EAP, and JON versions were initially selected based on the most recent production and patch levels generally available as of April 2010 and later updated in July 2010. The components of the test environment were selected based on common combinations in real-world IT operating environments.

We configured a set of three separate virtual machine sets hosted under VMware® Server running on Red Hat® Enterprise Linux® environments. Each set of virtual machines allowed multiple administrators to execute test scenarios independently of each other. One set of virtual machines was designated for performance testing, and all other environments were shut down during performance testing to prevent interference from other images. Each virtual machine set consisted of the following servers:

- Red Hat Enterprise Linux host servers running RHEL 5.4, running:
  - IBM WebSphere Application Server Network Deployment v7, fix pack 11
  - JBoss EAP 5.0.1
  - JBoss Operations Network (JON) v2.4.0.GA
  - Sun Java SE 6 (for JBoss)
  - IBM Java SE 6 (for WebSphere)
- For JBoss environments, one Red Hat Enterprise Linux host server running RHEL 5.4, running:
  - Apache HTTP Server v2.2.15
- For WebSphere environments, one Red Hat Enterprise Linux host server running RHEL 5.4, running:
  - IBM HTTP Server v7, fix pack 11
- One Microsoft Windows Server® 2003 server running the following:
  - Microsoft Active Directory®
  - IBM DB2 9.5
- Several VMware images acting as a load generator running under a OpenSUSE Linux host operating system running:
  - Microsoft Windows Server 2003
  - OpenSTA, Apache JMeter, JavaBench (load testing tools)

Additionally, each administrator had a local test and development environment running:

- Sun Java SE 6
- IBM Java SE 6
- Apache Maven
- Eclipse 3.4 (including the JBoss Eclipse plugin)
- IBM Rational Application Developer (RAD) 7.5.5

Appendix D – JBoss JIRA unresolved issues

During the hands-on testing of JBoss we have run into several issues with JBoss EAP and had to come up with workarounds. Some of the issues we discovered can also be found on the public JBoss JIRA bug tracking system. Below is the brief description of those issues that are still in the "unresolved" and "open" status at the time of this writing. Please note that these may appear to be closed at the time of your reading this paper, however often times JIRA records are closed only to be reopen under a different ID or rolled into another record or version of the software.

- JBossWS - Hot Deployment, caching previous failed deployment (https://jira.jboss.org/browse/JBPAPP-3427)
  This error will force server restart and downtime when someone attempts to deploy a certain kind of a broken EAR or WAR file into running JBoss instance.
- JVM Crash during redeployment of an incomplete deployment (https://jira.jboss.org/browse/JBPAPP-3133)
  This error will crash the JVM and cause unexpected downtime if someone attempts to do incomplete deployment of an application EAR or WAS file.
- JBoss crushes with OutOfMemory error (https://jira.jboss.org/browse/JBPAPP-4726)
  JBoss crushes with OutOfMemory error when running under heavy load with XA transactions (JTS implementation) that span two JBoss AS instances.
- Message Store does not participate in JTA transaction (https://jira.jboss.org/browse/SOA-280)
  This issue prevents JBoss JMS from being part of the global XA transaction. Also see related issue https://issues.apache.org/activemq/browse/AMQ-2514
- JBoss XA configuration recovery does not work (https://jira.jboss.org/browse/JBAS-6244, https://jira.jboss.org/browse/JBAS-7965)
  This is one of several issues on the JBoss XA recovery (or lack of thereof). Search for "transaction" on JBoss JIRA with the status "open" for a complete list of issues.

For a complete list of open JBoss issues, please search https://jira.jboss.org.

Appendix E – High Level Features Comparison

The table below shows a high level summary of features and capabilities, some of which we evaluated and measured in this study. There are many more capabilities provided by WAS that are not found in JBoss. While each feature is "interesting", our study has attempted to put a dollar value and measure business impact of having each of the capabilities listed below. After all, there is no reason to have a "feature" if it does not provide business value to its user.

Capability ($ - indicates capabilities measured in the study)	*IBM WAS 7*	*JBoss 5*
JEE5 support $	●●●●❺	●●●●❺
Production support is available in local language and local hours $	●●●●❺	●●❸
Quality up-to-date documentation $	●●●●❺	●●❸
Completeness of the Administrative GUI $	●●●●❺	●●❸
Performance $	●●●●❺	●●❸
Scripted administration $	●●●●❺	●❷
Administrative security roles separation $	●●●●❺	●❷
Upgrade tools $	●●●●❺	❶
Problem determination tools (error messages, log analyzer, etc.) $	●●●❹	❶
LDAP support and compatibility $	●●●●❺	●●❸
Mature JMS provider with fast failover capability and high performance $	●●●●❺	●❷
DBMS support and certifications $	●●●●❺	●●●❹
Secure intra-cluster communications $	●●●●❺	●❷
Command assist feature (audit script) $	●●●●❺	❶
Hot deploy capabilities $	●●●❹	●●●❹
Install footprint on HDD $	●●❸	●●●●❺
Install time on developer machine $	●●❸	●●●●❺
Install and configuration time on production machine $	●●●❹	●●❸
Server startup time $	●●●❹	●●❸
HTTPSession failover	●●●●❺	●●❸
High availability and maximum up-time (avoid server restarts)	●●●●❺	●❷
Scalability	●●●●❺	●●❸
Integrated Development Environment	●●●●❺	●●❸
Support for latest WS* standards	●●●●❺	●●●❹
XA transaction support and recovery	●●●●❺	●❷
OS support	●●●●❺	●●●❹
Automatic WAN backup cluster support	●●●●❺	❶
Dynamic page fragment cache (Servlets, JSP, etc.)	●●●●❺	❶
Performance tuning advisor	●●●❹	❶
Flexible management capability (scheduled management of remote servers)	●●●●❺	❶
Eclipse based admin script development tool	●●●●❺	❶
Secure audit of administrative actions	●●●●❺	❶
DMZ hardened proxy	●●●●❺	❶
Ability to manage mixed version environment from one console	●●●❹	❶
Ability to fully configure the server without access to its file system	●●●●❺	❶
Portlet JSR 286, WSRP 2.0 support	●●●●❺	❶
SIP protocol support	●●●●❺	❶
SCA support	●●●●❺	❶

❶ - not supported, ❷ - weak, ❸ - limited, ❹ - good, ❺ - excellent

Appendix F – Studies on Software Costs

According to several independent industry studies, the cost of software maintenance and it evolution has risen to over 90% of the total project cost (source: http://users.jyu.fi/~koskinen/smcosts.htm).

The definition of the software maintenance costs in the table below is slightly different depending on the study at hand, but generally speaking it includes system administration and management, software upgrades, software enhancements and refactoring, bug fixes, troubleshooting and other support related cost factors.

Year	Proportion of software maintenance costs	Definition	Reference
2000	>90%	Software cost devoted to system maintenance & evolution / total software costs	Erlikh (2000)
1993	75%	Software maintenance / information system budget (in Fortune 1000 companies)	Eastwood (1993)
1990	>90%	Software cost devoted to system maintenance & evolution / total software costs	Moad (1990)
1990	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Huff (1990)
1988	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Port (1988)
1984	65-75%	Effort spent on software maintenance / total available software engineering effort.	McKee (1984)
1981	>50%	Staff time spent on maintenance / total time (in 487 organizations)	Lientz & Swanson (1981)
1979	67%	Maintenance costs / total software costs	Zelkowitz et al. (1979)

For a complete list of bibliography of over 20 titles, please visit http://users.jyu.fi/~koskinen/smcosts.htm.

Appendix G – References

- IBM WebSphere Application Server Library, http://www-01.ibm.com/software/webservers/appserv/was/library
- IBM WebSphere Application Server for Developers FAQ, http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/faq-wasdevelopers.html
- WebSphere Training and Certification, http://www.ibm.com/developerworks/websphere/education
- IBM WebSphere Application Server Pricing, http://www-01.ibm.com/software/webservers/appserv/was/appserv-was-pricing.html.
- JBoss Documentation, http://www.redhat.com/docs
- JBoss AS Wiki, http://wiki.jboss.org/
- Best Practices for Performance Tuning JBoss Enterprise Application Platform 5, http://www.jboss.com/pdf/JB_JEAP5_PerformanceTuning_wp_web.pdf
- JBoss Training, http://www.jboss.com/services/training
- JBoss issue tracking system, http://jira.jboss.org
- JBoss Pricing, http://www.redhat.com/f/html/jboss_channel_skus.html
- Geronimo DayTrader Example Application, http://cwiki.apache.org/GMOxDOC20/daytrader.html
- Usenix System Administrators Special Interest Group System Administrator Job Descriptions, http://www.sage.org/field/jobs-descriptions.html
- A Study on Reducing Labor Costs Through the Use of WebSphere Cloudburst Appliance: http://www-01.ibm.com/software/webservers/cloudburst
- Software Maintenance Costs, Jussi Koskinen, http://users.jyu.fi/~koskinen/smcosts.htm

Credits

Authors would like to thank members of the JBoss community and Red Hat developers who responded to forum posts with solutions and troubleshooting tips. We would also like to thank Red Hat JBoss Performance Team for tuning recommendations for DayTrader on JBoss provided during the JBoss World 2010 in Boston.

We would like to thank Roman Kharkovski (IBM SWG) and Roger Leuckie (IBM SWG CPO), and IBM Performance Lab employees for assistance in performance tuning and testing of WebSphere and JBoss.

Legal and Trademarks