



white paper

JBoss Application Server Migration Strategies

Unisys Global Commercial Industries Strategy and Solutions Management

solutions

Are you considering adopting a J2EE-compliant Open Source application server such as JBoss? Are you uncertain how to migrate from your existing application server to JBoss with minimal business disruption? This white paper focuses on these two topics while assisting CIOs with their application server migration strategy.

Table of Contents

Executive Summary	4
Migrating to JBoss	4
Factors Affecting the Migration Decision	4
Why Migrate Your Application Servers?	5
The Rise of the Enterprise Application Server	6
The Growing Importance of the Application Server	6
Java Application Server Market Dynamics	7
JBoss Platform and Strategy	8
JBoss – The Open Source Pioneer	8
JBoss AS	9
The Architecture of JBoss AS	9
Java Enterprise Middleware System (JEMS)	10
Professional Open Source	11
Migration Customer Example: The Hospitality Industry	12
The Benchmark	12
Migrating from WebLogic to JBoss	13
Going Live	13
Migration Planning	14
Migration Planning and Project Management	14
Business Analysis	14
Requirements Analysis	14
Deployment Design	15
Implementation	16
Skills Migration	16
Services Available	17
Service Offerings for JBoss and JEMS	17
Consulting Services	17
Professional Support Services	18
Training Services	18
Conclusion	18
About the Author	19

Executive Summary

IT organizations face a difficult mandate: to develop business solutions that create competitive advantage but with a lower total cost of operation. Many IT professionals are responding to this challenge by migrating to Open Source application servers such as JBoss.

Distributed under the lesser GNU public license (LGPL), JBoss is free, which means IT executives have one less hoop to go through. Still, questions remain. Some executives wonder if their organization is ready for an Open Source solution, while others question how their business goals align with the unique capabilities of JBoss. Further along in the decision process, professionals speculate on the best way to migrate to JBoss and what services are available to assist their staff throughout the migration.

This paper addresses these topics while striving to show that application server migration does not have to be painful. Adding a defined strategy and process to migration often leads to success.

There are three strategies that migrations can follow depending on business goals and constraints:

- Rip-and-replace
- Incremental transition
- Cohabitation and coexistence

Regardless of strategy, the migration process consists of five clearly defined phases:

- Business analysis
- Requirements analysis
- Deployment design
- Implementation
- Skills migration

In addition to discussing the strategy and process behind application server migration, this paper also explores the architecture of JBoss and how the Professional Open Source model of JBoss Inc. is working to remove the barriers and risks associated with enterprises adopting Open Source.

This white paper provides a quick and thorough overview of the migration process to help you understand the basic steps of adopting JBoss. If you are new to the subject, the steps outlined in this paper will help you become familiar with the planning that lies ahead. For those who have already begun, this paper will let you know whether or not you are on track.

Migrating to JBoss

Factors Affecting the Migration Decision

The application server market is a battle between behemoths IBM, BEA, and Oracle — and an Open Source upstart called JBoss. Unlike the others, however, JBoss distributes its software for free. This lack of a price tag has motivated IT professionals with a compelling TCO argument for adopting and migrating to JBoss.

Before CIOs can adopt JBoss though, they must first evaluate their organization's readiness to deploy and support Open Source. Beyond the obvious TCO argument for JBoss, what other factors should companies consider before taking the migration plunge? Organizations should ask themselves the following questions to determine how Open Source fits with their software stack:

- Is there adequate in-house expertise to manage Open Source deployment, modification, and maintenance?
- What support services are available for implementation and maintenance, and how much do they cost?

- What is the degree of vendor lock-in and reliance on proprietary features? Should applications adhere more strictly to the J2EE standard?
- Compared with proprietary solutions for given requirements, are the capabilities of Open Source solutions better, the same, or adequate?
- Does a given Open Source solution align with enterprise system requirements?
- Within current plans and future strategy, does a given Open Source solution meet the IT and business needs for stability, performance, and scalability?
- What are the costs associated with accommodating, administering, and maintaining Open Source code?
- Does the license agreement for the use, modification, and distribution of Open Source code match with IT and business objectives and address any legal concerns?

For many CIOs, it's a confidence issue. If management is confident in the applicability of Open Source and the IT organization's ability to support it, then adopting Open Source becomes imperative to impacting the bottom line.

Why Migrate to JBoss Application Server?

Many organizations have had their application servers in place for some time. Chances are those servers have been performing satisfactorily. So why migrate?

One major reason is, of course, the cost-effectiveness of the JBoss system, but other factors do exist. For instance, support for the existing version of the application server is being phased out. If this is the case, then migration is inevitable. Why not choose the most cost-effective option? Other reasons include a shift in programming paradigms. For example, some organizations have recently adopted component-based applications and see a migration to a J2EE application server as a natural step to achieve this. Others have realized that departmental applications don't scale well to an enterprise level, so they switch to a J2EE server to achieve this scalability.

Alongside the reasons to undertake application server migrations are the common challenges. The following bullets highlight a few that frequently emerge:

- The applications are distributed across different tiers — client, business, and database — and they use a different mix of J2EE technologies, including JSPs (Java Server Pages), EJBs (Enterprise Java Beans), and JDBC (Java Database Connectivity). The application programming interfaces (APIs) used across the applications are not usually uniform since they may have been developed at different times.
- The integration requirements of each application may differ. Some applications may entail integration with a single sign-on, existing security framework, while others may involve integration with third-party packaged products.
- Besides the Java platform APIs, the applications may use other technology components developed for Java applications, including messaging frameworks and utilities for logging and exception handling. Applications that depend on such components add complexity to a migration.
- Applications vary in their complexity. It's hard to apply the same migration methodology to all of them.
- The expertise of application teams ranges. All team members may not understand the benefits of a structured migration.

As the preceding list of challenges makes clear, a migration plan is helpful. The first step is to decide the high-level strategy. The strategy sets the context and framework around which more detailed planning activities are created.

Table 1 lists the three common web-based application infrastructure migration strategies. Phased approaches such as incremental transition and cohabitation tend to be less risky and less costly than a rip-and-replace approach. Organizations determined to rip and replace can reduce risk through extensive prototyping and benchmark testing.

Table 1: Migration Strategies

Selecting the incremental and cohabitation (phased) approaches means management first has to decide which applications should be migrated, and which of these applications will deliver an ROI quickly. Business priorities determine these decisions, and the business analysis report documents them.

Business analysis is the first step of several in a successful migration. The follow-on steps are requirements analysis, deployment design, implementation, and skills migration. Each step will be discussed in detail in the coming sections. First, however, we consider the market dynamics of migration and how the technology and business models of JBoss fit into them.

The Rise of the Enterprise Application Server

The Growing Importance of the Application Server

In the beginning of the Internet era, static Web content delivery simply required a Web, or HTTP, server to deliver prebuilt HTML files to a browser. This simple arrangement did not last long. As the Internet matured and began to achieve its potential as a medium that could dynamically interact with customers, prospects, and partners around the globe, a comprehensive method of interacting with an enterprise’s applications and databases was required. Application servers assumed this role.

The rise of application servers reflects the evolution of web-based applications. The dynamic Web sites that today’s market demands require application servers in the software stack. Eventually, a three-tier client/server model was created, and the middle tier of the client/server environment became the realm of the application server. The application server provides the processing between a user’s Web browser and an organization’s backend, and allows for the dynamic interchange between the browser and the organization.

Migration Strategies	Description
Rip-and-replace	Big bang, all-or-nothing approach that is relatively high-risk if anything goes awry
Incremental transition	Low-risk, opportunistic approach based on migrating components that deliver rapid payback. This is an appropriate strategy for new projects or a new service using JBoss
Cohabitation and coexistence	A combination methodology that aims to manage risk in complex organizations. It is especially useful for migrating mission-critical applications that require substantial evaluation and testing

Table 1: Migration Strategies

As the use of the World Wide Web exploded in the mid-90s, it quickly became apparent that a standard was necessary for this tier of processing. Java, with its write-once-and-run-anywhere capabilities, began to fill this void. J2EE (Java 2 Platform, Enterprise Edition) would eventually become the standard for application servers.

J2EE is a Java platform designed for the mainframe-scale computing typical of large enterprises. J2EE simplifies application development and decreases the need for programming and programmer training by creating standardized, reusable modular components and by enabling the application server tier to handle many aspects of programming automatically.

The Enterprise Java Bean, or EJB, is the software component of the J2EE platform. EJBs provide a pure Java environment for developing and running distributed applications. They are written as software modules that contain the business logic of the application. EJBs allow enterprises to control change at the server rather than update individual computers with a client whenever a new program component is changed or added. Organizations can also reuse the EJB components in multiple applications. To use an EJB, it must be part of a specific application called a container.

Java Application Server Market Dynamics

The success of the Internet as a communications medium resulted, to some extent, from the standardization that J2EE provided. Today, with the ubiquity of the Internet and web-based applications, J2EE application servers are becoming commoditized.

In the past, J2EE vendors (such as BEA and IBM) sought to differentiate their products on traits such as scalability, reliability, and high availability. With the commoditization of the J2EE infrastructure, however, traditional licensing vendors are struggling to sell their J2EE products based solely on performance factors. Consequently, these products must distinguish themselves on features and functionality that go beyond the J2EE standard.

The Open Source movement adds yet another competitive pressure. Open Source J2EE application servers (such as JBoss) that are free of charge are eroding the market share of the large vendors (such as BEA and IBM) that rely exclusively on a licensing business model. This further motivates the vendors to create extended high-end features that will increase user lock-in.

The application platform suite (APS) is another example of how traditional licensing vendors have competitively responded to market pressures. The emergence of the APS

indicates a growing trend toward larger, more comprehensive suites of integrated solutions. The integration and portal components of an APS are less standardized than J2EE. Even when standards apply, they are not, for the most part, J2EE standards. The downside is that these additional elements are entirely proprietary to each specific vendor, and dramatically increase the degree of vendor lock-in for the applications that leverage them.

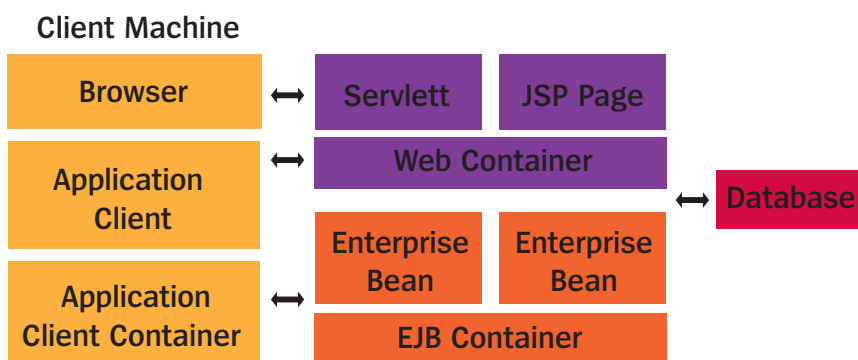


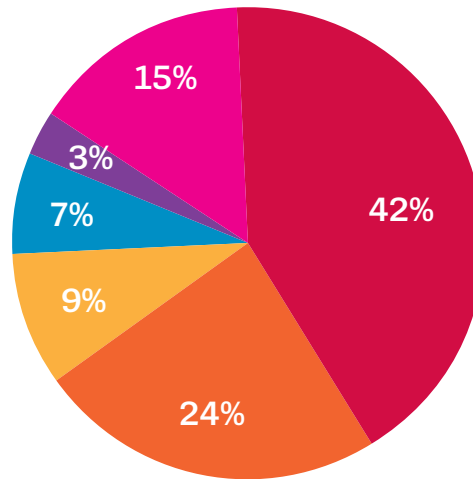
Figure 1: Application Server Overview

JBoss Share Position:

#1 Application server in development environment¹

#1 Integrated application server for OEMs and ISVs

#3 Position in production usage (27% JBoss, 34% BEA, 40% IBM). Fastest growth rate of all application servers²



¹ Source: BZ Research, November 2003

² Source: Gartner, February 2004

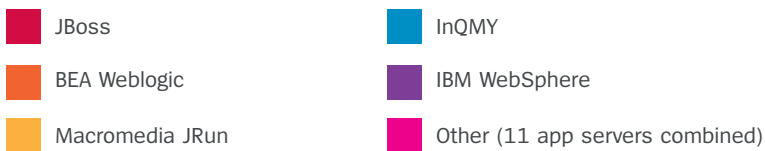


Figure 2: JBoss Growth in Market Share

This trend suggests that IT managers should take great care when selecting vendors because the lock-in will result in a long (greater than five-year) commitment to fully realize the investment. Organizations can protect themselves from lock-in by identifying open, J2EE standards-based infrastructure components to power their web-based applications. By adopting an Open Source strategy, IT organizations can have the widest degree of flexibility and control, while reducing the total cost of ownership (TCO) of the enterprise's infrastructure.

JBoss AS is an Open Source, J2EE 1.4 certified application server. The certification means that standard enterprise Java applications should run seamlessly on JBoss AS without requiring costly modifications. In the sections that follow, we will examine the major features of JBoss and its Professional Open Source model, as well as present some considerations for migration planning.

JBoss Platform and Strategy

JBoss – The Open Source Pioneer

JBoss is both a technology, as well as the company that stands behind that technology. The term JBoss is used in three contexts:

- JBoss: the leading Open Source application server (AS).
- JBoss: the umbrella set of Professional Open Source projects. Beyond the application server, the JBoss umbrella includes Tomcat, Hibernate, JBoss Cache/JGroups, and jBPM (Java Business Process Management). This collection is also referred to as JEMS (Java Enterprise Middleware System).
- JBoss Inc.: the corporation that employs leading JBoss developers and formalizes and promotes Professional Open Source, the second generation designation of the Open Source movement.

JBoss stands for Java Bean Open Source Software, and its cornerstone Open Source project is JBoss AS.

JBoss AS

From its beginning, JBoss AS has been compatible with J2EE, but official certification was begun in November 2003. In order to help meet the cost of certification, the JBoss J2EE Founders Program was created to pool resources among the companies involved in the project. The founders include Unisys, Intel, Borland, IONA, Sonic Software, and webMethods.

The Founders Program was instrumental in helping JBoss AS 4.0 become the world's first J2EE 1.4 certified Open Source application server in September 2004 (achieving its certification ahead of rivals IBM and BEA). The JBoss AS was designed with simplicity and flexibility in mind.

The Architecture of JBoss AS

The architecture of JBoss AS differs significantly from other J2EE application servers on two counts: the microkernel and Aspect Oriented Programming (AOP) layers.

The microkernel-based approach of JBoss AS is extremely small in footprint, so it takes up fewer machine resources by comparison. The microkernel layer uses several plug-ins (modules that implement particular services and functionality). The interaction between the microkernel and the plug-ins

occurs via a J2EE standard event-style protocol known as JMX (Java Management Extensions). The JMX protocol now forms part of the new J2EE 1.4 specification.

Highly modularized software and a formal interface for internal communications (the JMX) boost the application server's flexibility. In the case of Open Source products, where component services may be coming from different contributors at different times, this architecture works well, and helps form a solid foundation for JEMS (Java Enterprise Middleware System).

Through JMX, JBoss AS affords plain old Java objects (POJOs) access to transactional and clustering support typically reserved for EJBs. (POJO, a term coined by Martin Fowler and others, denotes a normal Java object that is not a Java Bean, and does not implement any special interfaces of any of the Java frameworks.) The interaction with POJOs is an important differentiator because POJOs are easier to program and the majority of Java applications do not use EJBs, and, therefore, do not benefit from some of the higher-end scalability features of the typical J2EE application server.

The Founders Program was instrumental in helping JBoss AS 4.0 become the world's first J2EE 1.4 certified Open Source application server in September 2004 (achieving its certification ahead of rivals IBM and BEA).

With the Aspect Oriented Programming (AOP) model used by JBoss, tags referring to tasks that need to be accomplished are placed in standard object-oriented (OO) code. When the code is compiled a weaver processor inserts relevant code, known as aspects, at the tag points. Aspects herald a new paradigm in code development, in the same way that object orientation did a decade or two ago. AOP makes it easier to encapsulate behavior that is usually messier, harder or impossible to do with regular OO techniques. AOP makes code development more streamlined and less repetitious.

JBoss AS is built from AOP, and its use reduces the need for EJBs in many cases. The combination of Aspects and POJOs greatly simplifies enterprise application development. Working with Aspects means that application developers rather than system architects can be involved in coding since the Aspects separate out the system architecture related features. AOP makes development with JBoss a much simpler and less costly task than it is for comparable J2EE application servers.

Java Enterprise Middleware System

Java Enterprise Middleware System (JEMS) refers to a portfolio of best-of-breed software products designed to work together, alone, or with an existing middleware solution. JEMS gives data centers the flexibility to choose products that can lead to the service-oriented infrastructure the market demands. With the introduction of JEMS, JBoss has expanded beyond its J2EE roots, turning its federation of Open Source products into an advanced Java middle-ware/development platform.

The microkernel architecture allows for loose integration, which permits plug and play of components. The plug-and-play feature means organizations can flexibly deploy JBoss and other components (JEMS) throughout the enterprise. The result is both extensibility and interchangeability of functional components. The plug-and-play microkernel architecture protects a company's technological flexibility, and reduces vendor lock-in.

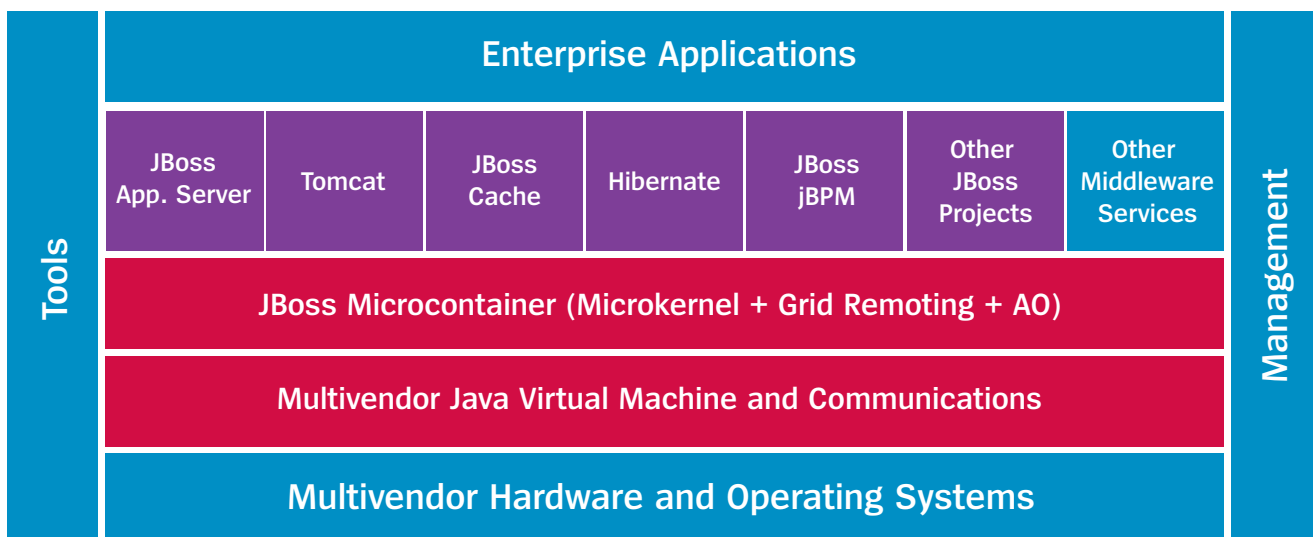


Figure 3: The Architecture of JBoss AS and JEMS

Professional Open Source

JBoss AS and the JEMS suite of products are licensed under the Open Source GNU Lesser Public License (LGPL) and are free to download and to use for both development and production deployments. In addition, the LGPL allows companies to embed JBoss into commercial products, which will broaden the impact of JBoss.

JBoss considers itself the leader in the second generation of Open Source, which its leaders have termed Professional Open Source. Professional Open Source brings professional structure to Open Source projects so that they are safe for commercial consumption and reduce the risk of deployment. This structure includes a legal foundation, a developer organization, code base responsibility, indemnification, ecosystem support, and service offerings. The Professional Open Source methodology has the following strategic elements:

- JBoss hires and pays experts and key developers in the Open Source community to write exceptional and innovative software full-time.
- JBoss offers services like 24/7 technical support, training, and deployment.
- JBoss has pushed Professional Open Source to provide the same level of legal protection as a commercial software license. JBoss offers an Open Source license, but through support and subscription services, users can get indemnity.
- JBoss employs certified partners. Its certified partners represent the company's primary sales channel, which both complements its own efforts and magnifies its reach.

As companies adopt standards across their IT infrastructure, they must ensure their investments are future-proofed against obsolescence. The JBoss Professional Open Source methodology directly addresses this mandate. Since all JEMS products are designed to be stand-alone, each product must drive innovation and achieve technical superiority on its own merits. JBoss invests heavily in continued innovation and R&D for all JEMS products by hiring the Open Source project leaders and experts as full-time employees. This important step of the JBoss Professional Open Source methodology results in innovative, best-of-breed products such as JBoss AS, Apache Jakarta Tomcat, and Hibernate — all of which are JBoss funded products.

JBoss considers itself the leader in the second generation of Open Source, which its leaders have termed Professional Open Source. Professional Open Source brings professional structure to Open Source projects so that they are safe for commercial consumption and reduce the risk of deployment.

Unlike Linux and most other Open Source projects, JBoss Inc. both develops and distributes its products. This arrangement combines the roles of the Open Source consortium and Open Source distributor. It gives JBoss much greater control over product evolution, and provides for high quality support services.

Before diving into the details of migration planning, it is instructive to take a high-level tour of a successful migration.

Migration Customer Example: The Hospitality Industry

A leading company in the hospitality industry faced the challenge of incorporating the Internet as an important, growing distribution channel for customers. The company's IT group was charged with improving and adding new services to its

online reservation portal. The difficult part of that assignment was maintaining and enhancing the customer experience while reducing cost.

After encountering JBoss in early 2002, the IT professionals at the hospitality company decided to investigate whether JBoss and Tomcat could handle the online loads and provide the response that is critical to the hospitality industry. Their motivation for migration was to avoid paying product license and increased support costs as deployments grew. They also wanted a more compatible best-of-breed product that had the latest J2EE specifications embedded within.

The Benchmark

The company evaluated JBoss AS under a series of internal benchmark tests and compared it to their current vendor BEA WebLogic. The internal tests consisted of running each

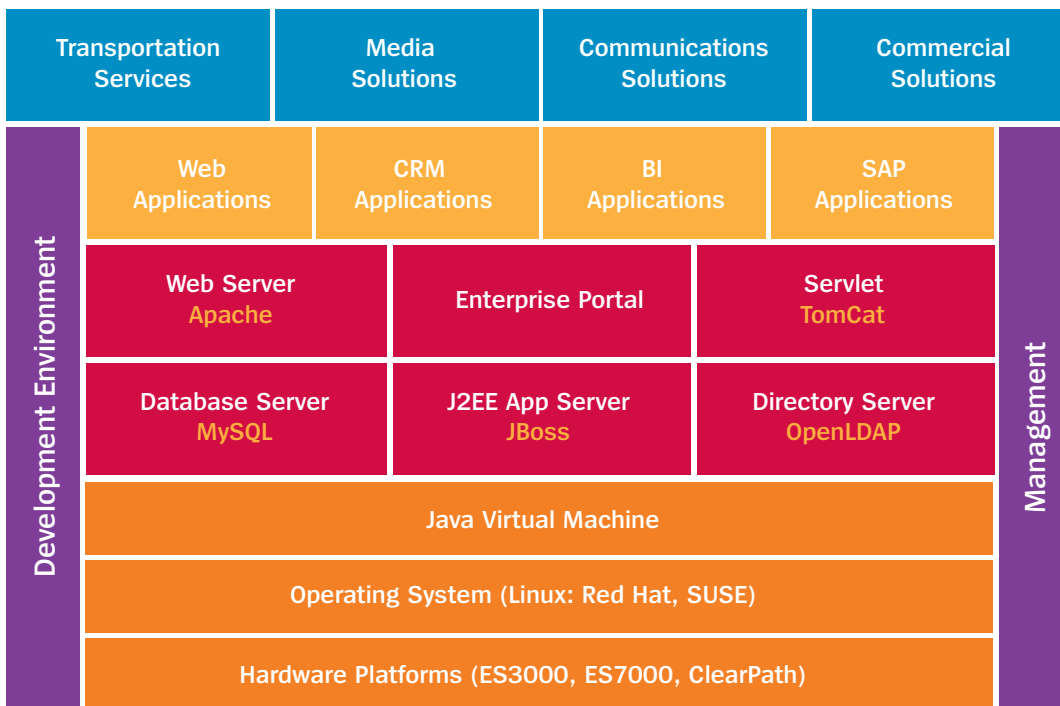


Figure 4: How JBoss Fits in the Open Source Stack

of its Java applications on both JBoss AS and BEA WebLogic under heavy load conditions and then measuring the resulting throughput, system utilization, and transaction response times. Internal testing revealed that JBoss handled 14% more hits per second, made 5% fewer transaction errors per second, and reduced transaction cycle time by 50%.

Migrating from WebLogic to JBoss

Although the benchmark tests were encouraging, the hospitality company was concerned about the workload necessary to port its existing Java applications running on WebLogic over to JBoss. The concern hinged on the use of custom extensions for WebLogic clustering. For the most part, the porting process exceeded expectations; however, since JBoss JSP (Java Server Pages) is a syntactically tighter implementation than WebLogic, some rewriting was required.

A senior Internet architect and the primary developer on the JBoss project saw firsthand the benefits of the Professional Open Source model. He stated, “The documentation was helpful and the few times that I needed to contact technical support, I was assisted by a very knowledgeable engineer.” The Professional Open Source model ensured that the company received assistance immediately and didn’t waste time chasing down support.

The project to switch to JBoss AS was completed in less than 6 weeks mainly by a single developer. It wasn’t until the final 2 weeks of the project that a second developer was brought on board to help with the configuration and tuning of JBoss for production.

Going Live

Although the real test of migration often is going live, the company was comfortable putting JBoss into production because of the internal load test and the support available from JBoss developers. It purchased JBoss Inc.’s 24X7 production support services with an average 2-hour response time. The support services covered JBoss AS, as well as all Open Source Java products in the JEMS suite — Hibernate, JBoss Cache, JGroups, Nukes, and even Tomcat.

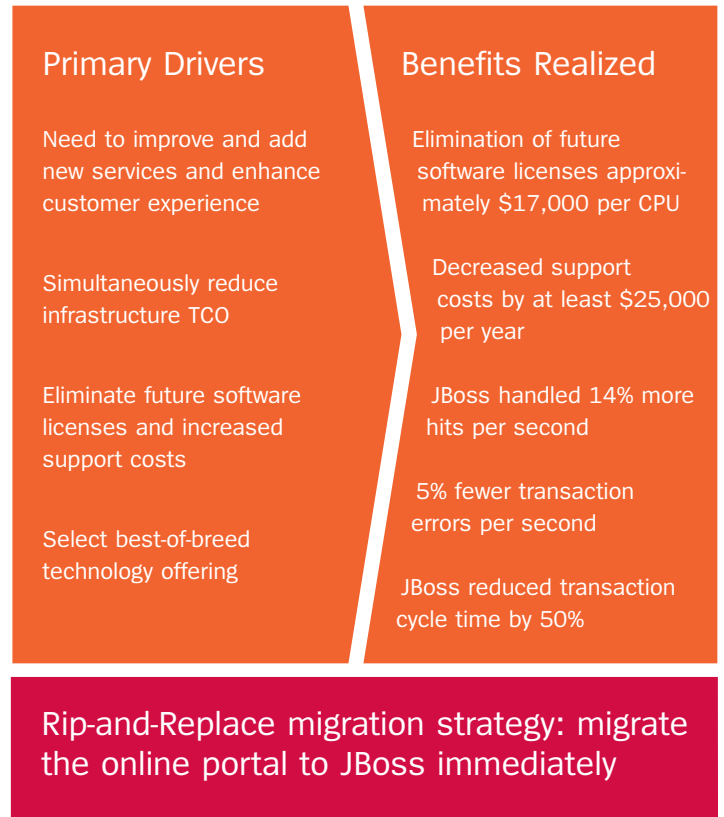


Figure 5: Rip-and-Replace Approach for Migrating to JBoss

Migration Planning

The portal deployment running on JBoss AS went live on a Saturday morning in March 2004, two days before Monday, which is typically the busiest day for reservations. JBoss AS successfully met the challenge of its first Monday, which, as it turned out, was the busiest day ever for the portal. That Monday, the portal experienced 20,000 visits by 18,000 unique visitors, 191,000 page views, and more than 3,300 reservations. And the organization didn't experience a single problem with its infrastructure.

Migration Planning and Project Management

Why was the hospitality company's migration to JBoss successful? Because management had a plan and followed it. Good migration strategies dictate what and how to migrate to JBoss by first determining the organization's business challenges and goals. Migration planning involves the following iterative phases:

- Business analysis
- Requirements analysis
- Deployment design
- Implementation
- Skills migration

Business Analysis

What goals should migrating to a new solution achieve? State them. The product of this phase is a business vision document that will be used as input for the requirements stage. It is critical that all stakeholders, including technical ones, are involved in this process. A successful business analysis will gain high-level support and begin the alignment of the organization. This analysis has two major components: business requirements and constraints.

A business vision statement resembles the executive summary of a project. It outlines the project's goals. They should be concise, such as "increase worker productivity" or "generate more online sales revenues."

The business problem statement encompasses two other areas: project scope and critical-to-quality (CTQ) features. The scope section briefly outlines what is in and out of bounds for the project. The CTQs are specific measurable characteristics that will define the project's success. An example CTQ could be to increase online availability by 1.9%, from 98.0% to 99.9%. Specific CTQs constitute an important aspect of migration planning because they become part of the next section — business constraints.

The constraints section formulates the business case for the project. It has several components: schedule, budget considerations, resources, cost of ownership, and ROI. By defining these areas, a foundation is laid for the project and the next step of migration planning can begin — requirements analysis.

Requirements Analysis

Requirements analysis means translating information about the business needs into use cases and other technical requirements. With this information, companies can identify possible solutions. They can decide which solutions are best for them by reviewing each one together with the business analysis.

A major consideration at this point is the degree of J2EE compliance of the displaced system(s). Many nonstandard services use proprietary features and functionality. It is critical to map these features against the J2EE standard to determine the impact of the migration on the current process. Similar considerations should be made for the analysis of all data sources that will interact with the JBoss system. At this juncture, an experienced migration service partner, such as Unisys, can be extremely helpful.

Requirements analysis should examine system qualities, particularly those qualities with an associated service level agreement (SLA). Some of the qualities to consider are performance, reliability, availability, scalability, security, and flexibility. Areas of improvement for specific SLAs must be documented and can be an extension of the constraints documented in the business analysis vision document.

The deliverable of this phase is a detailed assessment of the business and technical requirements drawn from domain knowledge and an understanding of the business objectives and the underlying system technology. At the end of this phase, the organization should have identified how JBoss can fulfill the business and technical requirements. The next phase, deployment design, sketches out the physical architecture.

Deployment Design

Deployment design begins by leveraging the detailed requirements developed previously and moves into sizing the system in terms of physical resources necessary to support the requirements.

The result of deployment design is a logical and physical architecture: a mapping of the requirements to a representative network topology. This topology is a network infrastructure that includes computing nodes, hardware requirements for each node, firewall design, and other devices on the network.

In addition to the design of the network components, organizations have to define how data will migrate to the new environment. This step may include developing provisioning techniques that will stage and transform the data to different formats. These data sources can include both transactional data as well as user profile information. The selection of tools and techniques to accomplish this migration aspect is a key component of deployment design.

Architecting the system is an iterative process composed of the following steps: design, deploy, and test. After an initial architecting session, examine the requirements and use cases to make sure that the proposed system fulfills the requirements. Build a test deployment and develop a series of tests based on representative use cases that check the deployment's ability to meet the business requirements. When the test deployment passes the test criteria, the architecture is ready for deployment in a real-world environment.

In addition to the design of the network components, organizations have to define how data will migrate to the new environment.

The last part of deployment design is creating a detailed implementation plan that includes milestones and skills needed for success. By this stage, there is sufficient detail to complete the business logic development for new projects and business logic migration for existing projects. There may be a temptation to begin work on the business logic very early in the process. Resist that temptation. Beginning these efforts too early in the cycle may result in significant recoding efforts, which only slow the project and create unnecessary “bumps” on the migration path. The details of business logic coding become clearer and can be created more smoothly when the bigger steps of business and requirements analysis are thoroughly documented and completed.

Implementation

After defining the deployment architecture, there is typically some time before actual implementation begins that allows for activities such as approval, contracts, and acquisition of resources and software.

Once implementation starts, organizations execute the steps outlined during deployment design. The implementation phase often includes creating prototypes in a test environment, running unit and system tests on the prototypes, and measuring the performance and other qualities of the deployment against the system requirements and business goals. After successful testing of the prototype, careful roll-out into production follows. System monitoring should continue to ensure that business goals are being fulfilled.

For complex or highly decentralized environments, there will most likely be iterative migration cycles that occur during the implementation phase. These cycles are necessary to balance the amount of migration activity with specific service functions within the business. These cycles are defined in the deployment design phase.

Skills Migration

Skills migration is an often overlooked aspect of the migration effort. Several different skill sets need to be considered:

- J2EE architecture and coding
- JBoss specific components architecture and coding
- JBoss specific components installation and configuration
- JBoss components operations and maintenance

If the displaced components are largely J2EE compliant, then the application development staff will likely have the Java skills required to accomplish the transition. If the development staff needs to increase their J2EE expertise, this ramp-up time must be added to the project plan.

The infrastructure and operations staff comprises the people who install, configure and keep the solution environment running efficiently. It is assumed that existing infrastructure staff is familiar with fundamental administrative issues and the operating systems on which the systems run, and has practical experience managing the daily operations of a site. The infrastructure team needs to learn how to install, configure and maintain the various components in a production environment. How much training the operations staff requires depends on the organizational goals. If outside help or consulting services are used, formal training may be reduced significantly because the operations staff can shadow the consultants.

Training deficiencies tend to show themselves very late in the deployment cycle. Neglected training can have long-term implications for the support and maintenance of the infrastructure. This causes project delays, unavailability, or malfunction. In short, it can prevent an otherwise well-planned project from delivering the intended business value. That is why it is critical to address these “soft areas” very early.

Services Available

Service Offerings for JBoss and JEMS

For many Open Source initiatives, customers frequently are uncertain of the effort required for migration. Experience from several migration projects indicates that customers have to consider the entire lifecycle of migration — performing the risky parts of the migration in a sandbox development environment, then performing it in a staging QA environment, and then actually moving to a production environment. JBoss is fully compliant with the J2EE standard; however, to varying degrees, displaced components of the infrastructure are not. This discrepancy can lead to complications that can only be solved through detailed product knowledge and migration experience.

Smooth and efficient migrations to JBoss have a common pattern: consultation with Unisys early in the process. A critical foundation for successful migration is the require-

ments analysis. Solid performance at this step can accurately define the level of effort and create the detailed statements of work that will allow for a winning migration.

Unisys offers the following services related to implementing and installing JBoss AS or JEMS components:

- Consulting services
- Professional support services
- Training services

Consulting Services

Unisys is an integral part of the JBoss Partner Program and a premier service provider. As such, Unisys can provide expert assistance at different stages of the application lifecycle.

JBoss Authorized Service Partners (JASPs), such as Unisys, are system integrators that can provide consulting and integration services and that have been thoroughly trained and certified by JBoss Inc.

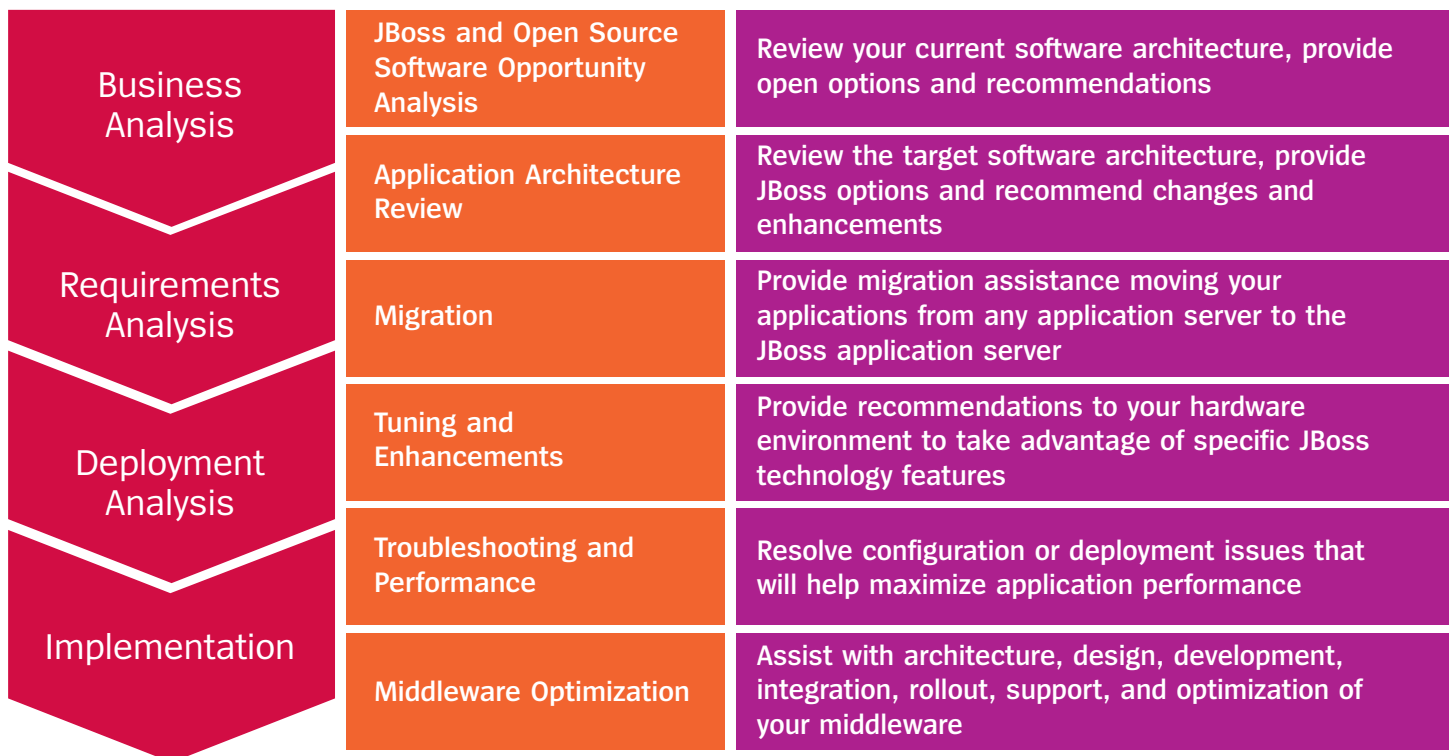


Figure 6: JBoss Consulting Services Across the Migration Lifecycle

Consider using the services of Unisys when a project requires the following:

- Acceleration of the development and deployment for critical projects
- Deep product knowledge about JBoss AS or a JEMS component that is essential for success
- Migration assistance from a proprietary application server to JBoss AS
- Performance tuning, optimization, development or architecture reviews

Professional Support Services

Unisys provides JBoss professional support services that help end users overcome all product-related issues involved in the deployment of JBoss AS or a JEMS component. These services encompass bug fixes, problem support, and developer assistance including configuration and performance optimization.

Professional support services have three levels for the entire portfolio of JBoss products. The levels range from 48-hour e-mail support to full 24x7 phone or e-mail support with 2-hour response times.

Training Services

Unisys also offers training services for JBoss AS and the JEMS components. Training services are essential for staff to become quickly productive. As mentioned earlier, companies often overlook staff training during the planning process. This is particularly true when migrating from an existing application server to JBoss AS. The training plan needs to consider not just the skills of the developers on staff, but also those of the maintenance and administration staff.

Conclusion

Reducing the cost of web-based applications while maintaining or increasing service levels marches on as one of every IT organization's biggest challenges. JBoss, an Open Source J2EE-compliant application server, can address both needs: greater TCO effectiveness and enterprise-wide scalability.

Taking advantage of these benefits requires a carefully planned migration strategy. This white paper provided an overview of what it takes to accomplish a migration from current state to JBoss and discussed the necessity of deliberate migration planning.

How can organizations begin to adopt JBoss?

- Document the business drivers and strategic direction of your Web services and application infrastructure.
- Examine the integration issues and licensing complexity of the current Web services infrastructure.
- Identify areas of improvement to meet the strategic direction.
- Discuss the plans and improvements with Unisys to determine if JBoss can help you achieve your goals.

Migrating your existing infrastructure to JBoss may seem like ascending Mount Everest. From a distance, it looks like a difficult and grueling task, but armed with the right information, you can begin moving mountains within your organization. Best wishes in your journey!

About the Author

Dr. Ravi Kalakota is Managing Partner and Vice President of Strategy and Solutions Management for the Global Commercial Industries practice of Unisys Corporation. He leads the development and delivery of integrated business solutions based on Open Source Software, mobile and radio frequency identification (RFID), and business analytics.

A pioneer and thought leader on the subject of IT-enabled business, Kalakota previously served as CEO of E-Business Strategies (EBS), a technology research and consulting practice. At EBS, he worked with both large and small companies to create and execute business process outsourcing and digitization (e-commerce, e-business, and e-service) strategies. In addition to consulting, Dr. Kalakota advised and spoke to business and technology audiences worldwide.

In addition, Kalakota has co-authored several best-selling books on e-commerce, e-business, m-business, services, and global outsourcing. His books include: e-Business 2.0: Roadmap for Success (Addison-Wesley, 2001), and M-Business: The Race to Mobility (McGraw-Hill, 2002), Offshore Outsourcing: Business Models, ROI and Best Practices (Mivar Press, 2004), Mobilizing SAP: Business Processes, ROI and Best Practices (Mivar Press, 2005), and Global Outsourcing: Executing an Onshore, Nearshore or Offshore Strategy (Mivar Press, 2005).

Dr. Kalakota received his Ph.D. in business administration from the University of Texas at Austin, his M.S. in computer science from the University of Hawaii, and his Bachelor of Technology in computer science from Osmania University, India.

For more information about Unisys and JBoss solutions and support, contact Unisys at: 1-770-368-6307.

Or, you can learn more on the Web at: <http://www.opensourcetrends.com>

© 2005 Unisys Corporation

All rights reserved.

Unisys is a registered trademark of Unisys Corporation. Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries. JBoss is a registered trademark and servicemark of JBoss, Inc. Linux is a registered trademark of Linus Torvalds. All other brand names or products referenced herein are acknowledged to be trademarks or registered trademarks of their respective owners.

