CLUSTER ARCHITECTURE FOR JAVA WEB HOSTING AT CERN

Michal Kwiatek, CERN, Geneva, Switzerland

Abstract

Over the last years, we have experienced a growing demand for hosting Java web applications. At the same time, it has been difficult to find an off-the-shelf solution that would enable load balancing, easy administration and a high level of isolation between applications hosted within a J2EE server.

The architecture developed and used in production at CERN is based on a Linux cluster. A piece of software developed at CERN, JPSManager, enables easy management of the service by following the self-management paradigm. JPSManager also enables quick recovery in case of hardware failure. The isolation between different clients of the service is implemented using multiple instances of Apache Tomcat, but the architecture is open and a different J2EE server can be incorporated if necessary. This paper describes this architecture in detail and analyses its advantages and limitations. Examples of HEP related applications, which make use of this architecture, are also given.

INTRODUCTION

The number of Java web applications used at CERN has increased considerably over the last few years. The fact that users were running web applications on their PCs was considered as a potential security threat to the laboratory, as it was impossible to enforce a satisfactory level of patches to the application server software run by individuals.

Furthermore, maintaining a production service requires skills and resources different from those needed to create an application. Although individual teams of analysts and developers have the intimate knowledge of specific requirements that enable them to build successful applications for their users, smooth operation of the service is ensured through mundane tasks such as: provisioning of reliable hardware, providing redundant hardware, performing upgrades and application patches to the application server software, providing backups and monitoring. These tasks tend to be similar regardless of the application and performing them centrally can produce considerable synergies.

It has been therefore decided that a central service for Java web hosting, hereafter called J2EE Public Service[1], should be created at CERN. This paper describes the architecture and management software that has been built and is currently used within J2EE Public Service, analyses its advantages and limitations and gives examples of HEP related applications hosted within the service.

REQUIREMENTS

It has to be noted that from the very beginning large, mission critical applications were out of the scope of the service. Such applications often have very particular needs and the effort necessary to satisfy all potential needs in a general way would be too expensive. Also, mission critical applications very often require and can afford a dedicated service run on specific hardware.

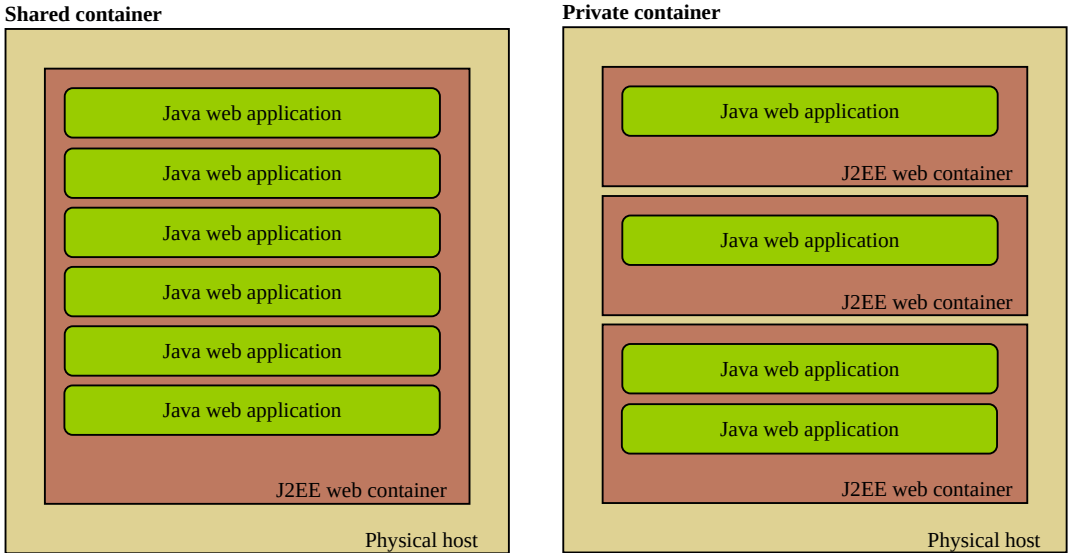

Figure 1: Important design choice: shared and private container scenarios.

Consequently, the project focused on medium-sized applications. However, the requirements were gathered throughout discussions with owners of both types of applications. The two groups actually agreed that most interest lay in the J2EE web container, as opposed to the full J2EE stack. Apache Tomcat has been proposed as a good alternative to Oracle OC4J.

A question asked many times during these discussions was the following: how do we ensure a proper isolation of applications hosted within the service? This proved to have important implications for security, stability and manageability of the service. How do we make sure that one application does not overwrite another application's files?

Two scenarios have been considered (see Fig. 1). The standard approach, in which many applications share the same web container, clearly does not provide sufficient isolation. Applications hosted in the same container would be run with privileges of the same operating system user, which would make file system access restrictions ineffective.

A private container scenario, in which each application owner is given a separate container, overcomes these limitations. Containers can be run as separate operating system users and custom configuration changes can be put in place, if needed by the applications. However, this scenario has a performance overhead due to running many Java virtual machines, so its feasibility had to be proven.

Performance tests have been carried out to compare the two approaches and to determine which application server software to use. Based on the experience we had, we considered Apache Tomcat[2], Oracle OC4J and Oracle iAS.

Test results[3] proved that private containers indeed had a performance overhead over a shared container, but that overall performance of private containers satisfied the design goals. Memory has been identified as the key issue limiting the number of applications that can be hosted on a single host and Apache Tomcat performed best in this scenario thanks to having the smallest memory footprint.

THE ARCHITECTURE

The architecture designed and built at CERN is based on a Linux cluster: it uses a set of machines configured in a similar way to provide scalability (see Fig. 2). User applications are hosted in private J2EE web containers (separate containers are configured for each application owner) and are run in separate operating system processes. Consequently, they accept network connections on different port numbers.

An additional component, hereafter called the proxy, is used to make the actual location of the application transparent to the end user of the application (web reader). Web readers always connect to the applications using the proxy and always use standard ports 80 and 443.

In case of hardware failure of one cluster node, service components hosted on this node (including the proxy) can be quickly restarted on a different node.

JPSManager software has been developed at CERN to manage this architecture.

Security

This architecture enables standard file system level

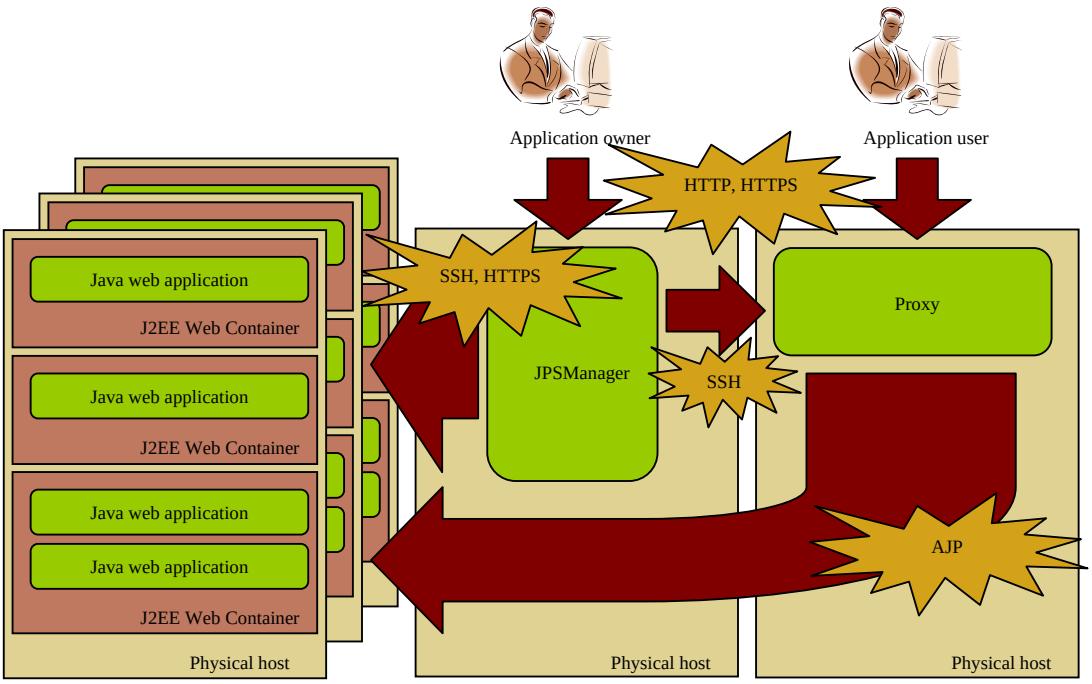

Figure 2: Cluster architecture used currently at CERN.

access control, which is the basic layer of security. On top of that, Java security manager is configured. It gives the service manager fine grained control over the actions that are allowed or disallowed (for example: opening a network connection to a remote database service is allowed but listening on a local socket is not allowed).

JPSMANAGER

JPSManager software has been developed at CERN to enable automatic management of the architecture described above. It is, in fact, an important element of this architecture as it configures a separate web container for a new user and later enables the user to deploy his/her application to the cluster.

Another goal of JPSManager is to provide a layer of abstraction over the functionality of the J2EE web container used within the service. This is meant to enable easy evolution of the service in the future: the underlying J2EE container, the proxy and even the policy to assign containers to users can be changed with only minor changes to the JPSManager software.

This flexibility has been achieved by building the software around three main interfaces (see Fig. 3):

- JPSContainerManager, which formally describes methods needed to be implemented when a new type of container is to be used within the architecture,
- JPSProxyManager, which describes methods for a specific type of proxy,
- JPSContainerAssigner, which formalises policies to assign container to application owners (web authors).

JPSManager also provides a rich administrative interface for service managers (both www and command-line) and SOAP web-services to enable integration with external systems. The software has been fully instrumented to provide logs and detailed performance records of the entire cluster.

J2EE PUBLIC SERVICE

The J2EE Public Service uses the architecture described above to provide a production Java web hosting service at CERN. The service uses JPSManager software, Apache Tomcat J2EE web container (application server) and Apache Web Server[6] as the service proxy.

The service is integrated with other standard services provided by IT Department at CERN:

- Central Web Services[5], which provide a central registry for all types of web applications,
- The Central Database Service
- NICE authentication: the standard authentication method at CERN, recommended to web authors to make authentication of their users (web readers) easier and more secure.

The Service Level Agreement is aimed for medium-sized, non-critical applications with full support within CERN working hours.

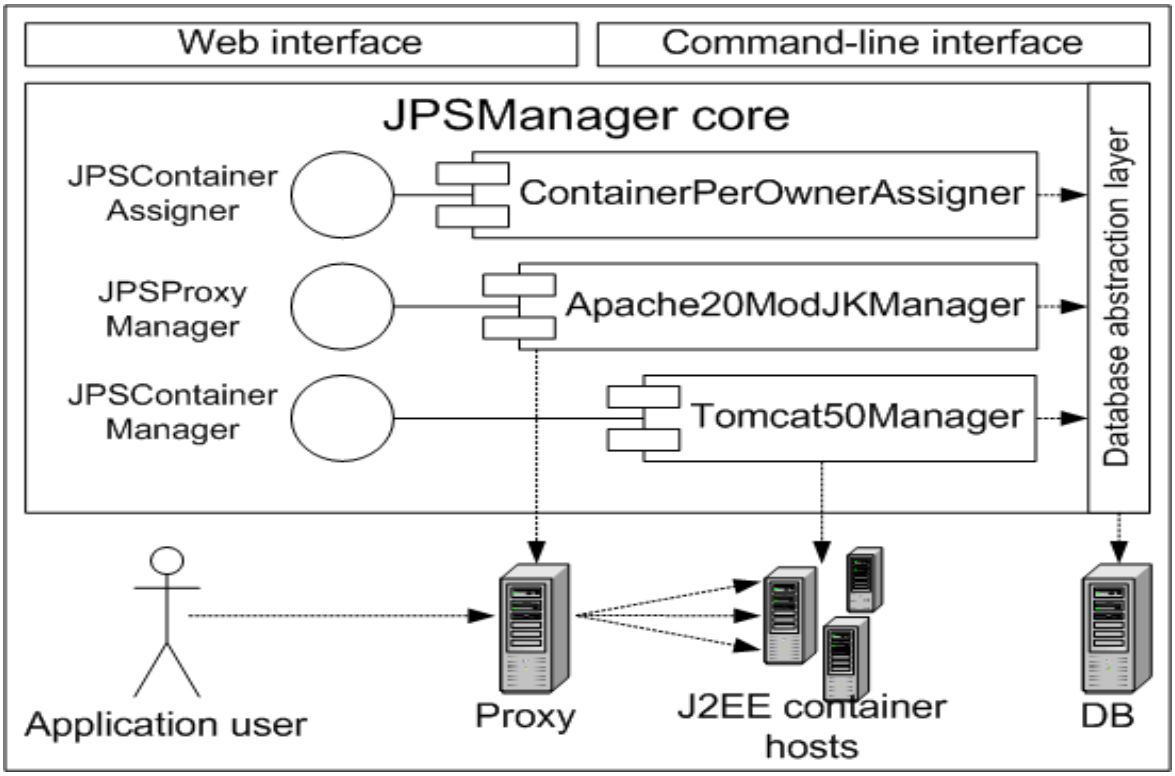

Figure 3: JPSManager - flexibility achieved by building the software around three main Interfaces.

Examples of applications[*]

The following applications are, among others, currently deployed within J2EE Public Service:

- Automatic Holding Point is a central point of contact between AT Department engineers, analysts and external companies to enable follow-up of LHC main dipole and quadrupole production. Dipole data is first uploaded to the system by external companies and then reviewed by CERN analysts. About 5 decisions a day are made based on that data; these decisions are communicated to the companies and engineers at CERN, who later access the data.
- Dipole Geometry Viewer provides online graphs and is used by the CERN AT Department engineers for magnet measurements similarity checks.
- Application enabling optimization of Short Straight Sections alignment for LHC machine, by implementing possible transformation: shift or roll. This application displays mechanical and magnetic horizontal and vertical profile as well as the racetrack of SSS. Geometrical positions of service lines (helium, electricity, etc.) and the beam lines are also displayed to help users determine convenient transformation to SSS.
- Information retrieval system for searching documents stored in EDMS, CDS and on JACOW website. It combines traditional techniques based on the vector space model with ontologies and more advanced natural language processing to improve precision of the search system.
- ATLASMonitor – a Web Information System managing documents related to construction of the ATLAS detector. It profits from the flexibility of web technologies, such as XML and XSL, to represent respectively the data and its format. Through a user-friendly interface one can easily describe how to insert, edit, show and search for specific pieces of information that comprise a document. Storage of new types of documents does not require code adaptation. ATLASMonitor is currently maintained by the ATLAS Technical Coordination and uses J2EE to handle the Quality Control Sheets of the ATLAS Tile Calorimeter commissioning phase.

Limitations

J2EE Public Service does not currently offer load-balancing or high availability solutions other than horizontal scalability and hardware redundancy that are inherently built into the architecture and ensure fast recovery in case of hardware failure. Enterprise Java Beans (EJBs) are currently not supported.

Evolution

Certificate authentication is going to be implemented in line with functionality offered by Central Web Services at CERN.

Flexibility of JPSManager software enables easy evolution of the service. The following solutions are going to be evaluated and implemented depending on user needs and available resources:

- Use of other J2EE containers: JBoss, Oracle OC4J.
- Clustering of individual J2EE containers.
- Use of hardware load balancer as service proxy.

SUMMARY

The J2EE Public Service provides a robust server infrastructure and deployment support to owners of Java web applications at CERN. Synergies have been created and risks of security threats have been reduced since backups, monitoring, patching and server software upgrades are performed centrally.

The architecture of the service is based on a Linux cluster; a proxy is used to enable transparent access to applications regardless of their actual location in the cluster. Hardware redundancy is inherently built into the architecture. Each application owner is assigned his/her private container, which provides a high level of isolation between applications and increases manageability. A piece of software developed at CERN, JPSManager, enables easy management of the service by following the self-management paradigm.

The J2EE Public Service does not currently offer EJBs support or high availability mechanisms. However, solutions such as clustering of individual J2EE containers and use of DNS and hardware load balancing are going to be evaluated.

AKNOWLEDGMENTS

I would like to thank Artur Wiecek and Eric Grancher for their very important advice and support throughout the work on J2EE Public Service.

Derek Mathieson and James Purvis provided very important input when requirements were collected; Natalia Emelianenko, Gregory Bevillard and Jerome Beauquis were the first users to migrate their applications to the prototype. Their feedback was very important and I would like to thank them for that effort.

I would also like to thank Alexandre Lossent and Alberto Pace for their help in integrating J2EE Public Service with CERN's central Web Services.

REFERENCES

[1] J2EE Public Service: http://www.cern.ch/j2ee-public-service.

[2] Apache Tomcat: http://tomcat.apache.org.

[3] Michal Kwiatek, J2EE Public Service, prototype phase, CERN Desktop Forum, February 2005: http://indico.cern.ch/conferenceDisplay.py?confId=a05985

[4] Apache Web Server: http://httpd.apache.org/

[5] CERN Web Services: http://www.cern.ch/web

[*] This section is based on the original descriptions kindly provided by application owners: Natalia Emelianenko, Jerome Beauquis, Antonio Jimeno Yepes and Felipe Fink Grael.