Application Server Clustering

Chris Shuster
University of Colorado at Colorado Springs
Colorado Springs, CO
cshuster@uccs.edu

# Abstract

Application servers are becoming increasingly common and varied in their uses. As such applications servers are being used in demanding environments where a single server is insufficient to handle the workload and provide high availability. Clustering is used to allow multiple servers to handle the workload and to provide redundancy. The focus of this research report will be an analysis of clustering application servers, how clustering works, the various techniques of clustering and creating/deploying a cluster.

# Overview

This section will provide an overview of the software used and constraints of this research report.

## Application Servers

Application servers unlike web servers are geared specifically for hosting and processing applications. Application servers however are generally meant to be used in conjunction with a traditionally web server. This is due to the fact that while application servers can and do host static content in addition to web applications a traditional web server is better suited for hosting static content, such as images. This is because each server type is optimized for a different purpose.

Application servers provide the base architecture for web applications allowing developers to focus exclusively on the web application being developed. Application servers as part of the base architecture provide services such as LDAP interfaces for authentication and database connection pools for data save and retrieve. This provides a standardized way for web applications to access such resources without needing prior knowledge for the type and location these services.

Platform and operating environment flexibility is one of the key features of application servers. Application servers are not only available for multiple operating systems and architectures but also multiple programming languages, including but not limited to Java and C++. Some application servers even make it possible to run the same web application on multiple platforms without needing to modify the application. This is due to application servers abstracting the specifics of the platform away from the application and due to the availability to application servers in platform independent languages such as Java.

## Research Constraints

The subject of application server clustering is a very large and complex matter due in no small part to the large variety of application servers, platforms and clustering techniques. It is for this reason that this research report must be constrained as time and resources are limited. As such the following constraints, with justifications as necessary, have been placed on this research report.

Only Java based application servers, more specifically JBoss Application Server 5, will be analyzed. Java is a common and well known development environment. Java is commonly used in web application development as the language provides features and functionality, in the form of the Java Enterprise Edition application API, which is well suited for this task. Additionally most modern application servers are geared towards Java, such as JBoss as the use of Java makes the application server and web applications platform independent. JBoss, which is a Java based application server, was specifically chosen as it is a common, well known and widely used application server.

Java based application servers have two main architectures for applications. The architectures are client side interceptor architecture and server side load balancing architecture. These architectures will be discussed in section 3. This research report will focus primarily on server side load balancing architecture. Server side load balancing architecture has similarities to traditional web server load balancing and assumes standard client software instead of specialized software client software like with client side interceptor architecture.

There is literally hundreds of different ways of load balancing servers. As such this research report will only focus on using Apache web server with MOD\_JK as the load balancer. This is due to Apache being one of the most well known and commonly used web servers on the internet and the ease of integrating Apache with an application server through the use of MOD\_JK.

## JBoss

JBoss is a Java based application server that implements, in its current version, JBoss Application Server 5, the Java Enterprise Edition 5 API. JBoss like other Java application servers provides prebuilt libraries that handle common tasks, such as database connection pools, that remove the need for the developer to manually implement. JBoss also utilizes standard XML for configuration purposes which increases the readability of the configuration file for anyone familiar with web technologies. JBoss is widely used commercially and privately to host applications.

## Apache Web Server and MOD\_JK

Apache web server is one of the most common and well known web servers on the Internet, with just under 50% of all web servers on the internet using Apache in 2009 [1]. Apache in addition to being so well know is also highly flexible as it is available on all common platforms such as UNIX, Linux, Windows and Mac OS. Further adding to Apache web server’s flexibility is its ability to load modules. One such module is MOD\_JK which provides both load balancing functionality and forward to application servers capability similar to MOD\_PROXY.

# Clusters

Clusters come in many forms and application server clusters like other forms of clusters solve several challenges will presents new unique challenges. The two main goals of a application server cluster is to provide high availability and handle large amounts of traffic. Both of which are critical as more and more applications become web based. On the flip side clusters add the unique difficulties of needing to be able to replicate configurations, applications and sessions to the members of the cluster. This starkly contrasts with static content hosted on a traditional web server as it only needs to replicate the actual web content data.

## JBoss Clustering

JBoss servers in a cluster are referred to as nodes [4]. These nodes are then partitioned into groups. Each partition represents a cluster of one or more JBoss nodes. The ability to partition JBoss nodes allows for multiple unique and even redundant clusters to be present on the same network. JBoss clusters can be any size as show in Figure 1 below. JBoss cluster partitions can be created either manually or automatically. Regardless of the creation method each node must have a unique name assigned to it. Manually configuring JBoss cluster partitions requires specifying each node’s cluster membership in its configuration files [5]. With automatic cluster formation the JBoss application servers will detect other JBoss application servers running in automatic formation mode and cluster with them [4].



Figure JBoss Clusters

The members of a cluster partition automatically replicate sessions among themselves to provide high availability such that if a node fails another node has sufficient information to immediately take over the failed node’s session with the client. This means that from the client’s perspective the transition from communicating with one node to another node is seamless if the node that was originally be communicated with fails.

The members of a cluster partition also automatically replicate the web application amongst themselves. This is known as application farming [4]. Application farming allows for the web application to be deployed to only one server and then automatically replicated to the other servers of that cluster. The specifics of deploying the application to all members of the cluster are handles behind the scenes by JBoss and require no further intervention other then deploying the web application to the initial node.

# Clustering Architectures

There are two core clustering architectures which are client side interceptor architecture and server side load balancing architecture. Each architecture is geared towards distinctly different styles of client requests. One of the key differences with the two architectural styles is where load balancing takes place. The architectures are not mutually exclusive as it is very common to utilize both in large web applications. The following two sections will detail each architecture.

## Client Side Interceptor Architecture

Client side interceptor architecture is used in situations where the client is running a custom developed client application. This client application utilizes EJB, JNDI, RMI and similar methods to communicate with the server. With the client running a custom application the load balancing and failover takes place on the client as seen in Figure 2. This is pre-implemented for the application developer in the form of a proxy class by JBoss and as such to enable the client side load balancing and failover the client application only needs to utilize a pre-existing JBoss library [4]. The client application needs to be aware of two or more nodes in the cluster to allow it to communicate with the cluster. The client actively communicates with the clusters nodes to determine where to route its requests.



Figure Client Side Interceptor Architecture

## Server Side Load Balancing Architecture

Server side load balancing architecture is used in situations where the client is running standard software such as a web browser. Clients communicate with the server using HTTP and similar style requests. As the client has no custom software it is then necessary to hide the specifics of the cluster from the client and as such load balancing and failover occurs on the server side as seen in Figure 3. The load balancer, also known as a director, handles the routing of requests. The director utilizes a load balancing algorithm while also taking into account if a node or multiple nodes have failed and routes request accordingly [2][3][4]. Additionally the director is configured to maintain persistent connections so that multiple requests from the same client session go to the same node [4].



Figure Server Side Load Balancing Architecture

# Cluster Setup

The following section will briefly detail setting up a server side load balancing architecture based cluster with JBoss and Apache web server equipped with MOD\_JK.

## Prerequisites

At least three servers or virtual machines are needed with each having a unique IP and being able to communicate across a network with each other. These servers should use an operating system that Apache and a Java virtual machine is available on, such as Linux or Windows. It is assumed that the one server will be the director and the rest will be nodes. As such it is assumed that the director server already has a default installation of Apache web server and that the node servers already have default installations of JBoss. Additionally it is assumed that all needed software to setup this cluster has already been downloaded, such as MOD\_JK.

## Notes

This setup demonstration was performed using three virtual machines all of which were installed with Ubuntu Server 8.10. The director uses the default installation of the Apache and the nodes were installed with the default installation of JBoss Application Server 5. Additionally all nodes were installed with Java 6.

As such this setup guide is geared towards setting up the cluster on Ubuntu Server 8.10 based servers. However this setup guide can be easily adapted to any platform. Additionally this setup guide is based on guides available from JBoss and Apache and as such their guides will be more generalized [2][3][4][5][6][7].

Be aware that JBoss Application Server 5 running as part of a cluster requires a substantial amount of memory and it is for this reason each virtual machine hosting a JBoss node should be allocated with at least 1GB of memory. The director on the other hand only needs 512MB of memory.

## Configuring Apache Web Server and MOD\_JK

1. Assuming MOD\_JK has already been downloaded copy the “mod\_jk.so” to the Apache modules directory which is located at “/usr/lib/apache2/modules”.

2. In the Apache home directory which is located at “/etc/apache2” create a folder named “conf” and another folder named “logs”.

3. In the “/etc/apache2/mods-available” directory create two text files with the names "mod\_jk.conf" and "mod\_jk.load".

4. Insert the following text into the "mod\_jk.load" file.
LoadModule jk\_module /usr/lib/apache2/modules/mod\_jk.so
5. Insert the following text into the "mod\_jk.conf" file.
<IfModule mod\_jk.c>

# Where to find workers.properties

JkWorkersFile conf/workers.properties

# Where to put jk logs

JkLogFile logs/mod\_jk.log

# Set the jk log level [debug/error/info]

JkLogLevel info

# Select the log format

JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE

JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat

JkRequestLogFormat "%w %V %T"

JkMountCopy all

# Mount your applications

#JkMount /servlet/\* loadbalancer

# You can use external file for mount points.

# It will be checked for updates each 60 seconds.

# The format of the file is: /url=worker

# /examples/\*=loadbalancer

JkMountFile conf/uriworkermap.properties

# Add shared memory.

# This directive is present with 1.2.10 and

# later versions of mod\_jk, and is needed for

# for load balancing to work properly

JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data

<Location /jkstatus/>

 JkMount status

 Order deny,allow

 Deny from all

 Allow from 127.0.0.1

</Location>

</IfModule>

1. In the “/etc/apache2/conf” directory create two text files with the names “uriworkermap.properties” and “workers.properties”.

2. Insert the following text into the "uriworkermap.properties" file.
# Mount the Servlet context to the ajp13 worker

/jmx-console=loadbalancer

/jmx-console/\*=loadbalancer

/web-console=loadbalancer

/web-console/\*=loadbalancer

1. Insert the following text into the "workers.properties" file.
# Worker List

worker.list=loadbalancer,status

# JBoss1

worker.jboss1.port=8009

worker.jboss1.host=192.168.84.128

worker.jboss1.type=ajp13

worker.jboss1.lbfactor=1

worker.jboss1.socket\_timeout=10

worker.jboss1.ping\_mode=A

# JBoss2

worker.jboss2.port=8009

worker.jboss2.host=192.168.84.129

worker.jboss2.type=ajp13

worker.jboss2.lbfactor=1

worker.jboss2.socket\_timeout=10

worker.jboss2.ping\_mode=A

# Load Balancer

worker.loadbalancer.type=lb

worker.loadbalancer.balance\_workers=jboss1,jboss2

#worker.loadbalancer.sticky\_session=1

# Status worker for managing load balancer

worker.status.type=status

## Configuring JBoss Nodes

1. Open and edit the “server.xml” located in “JBOSS\_HOME/server/all/deploy/jbossweb.sar/server.xml” directory, replacing “JBOSS\_HOME” with the directory containing the JBoss install.

2. Locate the line “<Engine name="jboss.web" defaultHost="localhost">” and insert ‘jvmRoute=”jboss1”’ replacing “jboss1” with a unique name for this node.

3. Locate the line “<Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}" redirectPort="8443" />” and uncomment the line.
4. Save the edits to the “server.xml” file and exit the text editor.
Note: Older documentation indicates that the line “<attribute name="UseJK">true</attribute>” needs to be added to the “server.xml” file, this is no longer necessary with JBoss 5 and later.
5. Repeat these steps for each node in the JBoss cluster.

## Starting the Cluster

### Director

1. Verify the Apache configuration by entering the command “sudo apache2ctl configtest”.

2. Start Apache by with the command “sudo apache2ctl start”.

3. Or restart Apache with the command “sudo apache2ctl restart” if it is already running.


### JBoss Nodes

1. Navigate to the “JBOSS\_HOME/bin” directory.
2. Execute the “run.sh” script with the following parameters “--configuration=all --host=0.0.0.0” so that the full command is “./run.sh --configuration=all --host=0.0.0.0”.

3. JBoss running.

4. Repeat these steps for each node in the JBoss cluster.

## The Cluster in Action

1. Open two different web browsers (such as Internet Explorer and Firefox) and navigate both to the address of the director, such as “http://192.168.84.130/jmx-console/”.
2. The page that loads in each browser has the name of the JBoss server that its connection is being forwarded to. They should be different.
3. The screen shot below shows the cluster functioning properly.


# Advantages and Disadvantages

Application server clustering and more specifically server side load balancing architecture has advantages and disadvantages. The following subsection will detail the advantages and disadvantages as well as some methods of mitigating the disadvantages.

## Advantages

There are three key advantages to utilizing application server clustering, although there are many other advantages these are the key ones.

The application server provides automatic failure detection and handle. Unlike traditional web servers the developer does not need to take into account how detect and deal with failures as this function is already built into the application server. Additionally since functionality this already built into the application server the behavior and way failures are detect and handled will be the same regardless of the web application.

The application servers automatically replicate sessions among themselves in order to provide seamless failover. Automatic session replication means that each member of the cluster is aware of the other node’s sessions so that if a node fails its session can be immediately redirected to another node without the client having to reconnect. Furthermore JBoss supports replication of specific fields instead of entire sessions in order to reduce the traffic on the local network between the nodes. This allows the developer to specify that only key field need to be replicate in order to provide seamless failover. This functionality is not provided by default web servers.

The application servers provide the facilities to automatically replicate the web application to all members of the cluster. The web application merely has to be deployed to a single node which will then distribute the application to all other nodes. This means that ensuring all nodes are updated with newer version of the web application and have an identical copy of the web application is completely automated.

## Disadvantages

While there are disadvantages to clusters in general they will not be discusses instead focusing specifically on disadvantages of server side load balancing architecture. There are two key disadvantages to this architecture, although there are more disadvantages these are the key points.

Utilizing a load balancer (director) creates a single point of failure. All requests from clients are sent to the director and all replies from the servers are sent back through the director. Even if the entire cluster is functioning properly if the director fails it will render the cluster unreachable. As such the director would be a target denial of server attacks and would require the director to be run on high end hardware in order to sustain the loads and reliability.

Additionally this architecture requires the use of additional software and/or hardware beyond just JBoss on the server side in order to provide load balancing and failover functionality. This increases costs of the system as additional equipment, cooling and software licenses may be needed. This also increases the complexity of the deployment as the director has to be configured in addition to the JBoss clusters.

## Mitigation

There are multiple solutions to mitigate the disadvantages but for the purposes of this section only the key mitigation technique will be analyzed.

In order to mitigate the key disadvantage it is necessary to utilize multiple directors. This would require a cluster of directors in addition to the cluster of application servers. This cluster could be created using conventional techniques of clustering such as LVS. This however does present a challenge that must be overcome in order for the director cluster to function properly.

While all directors running Apache with MOD\_JK are aware of the availability of each application server MOD\_JK is not currently capable of replicating sessions to other Apache web servers running MOD\_JK. This means that if a director fails the client would have to reconnect when their connection is failed over to another director. As MOD\_JK and Apache are open source software they could be modified in order to add this support or a different piece of software and/or hardware could be used as the director instead of Apache. Possible alternatives are however outside the scope of this research report and topics for future research as detailed in section 6.

# Future Research

There are several challenges that the clustering techniques discussed in this research report present, that are not resolved in this report. In addition the client side interceptor architecture was not analyzed in depth. As such the following are topics for future research.

* Client Side Interceptor Architecture
* Director Clustering Methodologies
* Alternative Techniques of Providing Load Balancing
	+ Apache Web Server with MOD\_RPOXY
	+ Hardware Load Balancers

# References

1. Netcraft Ltd. 2009.
April 2009 Web Server Survey
http://news.netcraft.com/
2. Apache Software Foundation. 2009.
Web Server How To
http://tomcat.apache.org/connectors-doc/webserver\_howto/apache.html
3. Apache Software Foundation. 2009.
Reference Guide
http://tomcat.apache.org/connectors-doc/reference/apache.html
4. Brian Stansberry and Galder Zamarreno. 2008.
JBoss Application Server 5 - Clustering Guide
5. JBoss Community. 2009.
JBoss Application Server 5 - Installation And Getting Started Guide
6. JBoss Community. 2008.
JBoss Run Parameters
http://www.jboss.org/community/docs/DOC-10749
7. JBoss Community. 2009.
Using Mod\_jk 1.2 With JBoss
http://www.jboss.org/community/docs/DOC-12525