

Architectures for J2EE Application Clustering in BEA and IBM Application Servers

Sebastian Popa
Helsinki University of Technology
T-110.551 Internetworking Seminar
spopa@hut.fi

Abstract

Java 2 Platform Enterprise Edition (J2EE) was created as a standard with the purpose to develop applications for the enterprise. The enterprises require multi tier applications that access databases concurrently and provide high scalability and availability at the same time with reduced cost and rapid development. One solution is to build such applications by using the standardized services provided by J2EE such as: database access, transactions, security and messaging. The clustering techniques used by the J2EE application servers enable them to serve large amount of users and provide high availability services. This paper presents the general concepts and possibilities provided by the clustering of the J2EE applications and then describes the solutions provided by BEA and IBM. The paper is intending to give an overview and provide pointers for further detailed descriptions of the technologies.

KEYWORDS: J2EE clustering, WebSphere, WebLogic

1 Introduction

Historically, the enterprise applications were developed firstly on mainframe computers. The client server and 3-tier architectures came later, when the networks and computers became cheaper. The standardization of such a architecture brought the possibility of having commercial implementations of the standard that bring state of the art quality at reasonable cost. Enterprises can use already developed platform and write their business application only.

The J2EE standard was developed by Sun Microsystems to address this and it is currently at the 1.4 version [1]. The standard include the specifications of the Application Programming Interfaces (API) that the application can use and the specification of the containers that provide the run-time environment for the applications. There is a container for hosting java servlets and Java Server Pages (JSP) pages, called WebContainer, and a container for running Enterprise Java Beans (EJB) components [2]. There are even containers for the application client and for the applets but they are not the subject of this paper.

In this architecture, the clients are usually running a web browser and they are communicating with a HTTP server that contains static pages and from there with the WebContainers that include java servlets and JSP pages. There can

be stand alone java clients that directly access EJB components which run in the EJBContainer. More common, the JSP and Java servlets or some other EJBs access the EJB components. The protocol through which they are communicating is the RMI-IIOP protocol (Remote Method Invocation over Internet Inter-Orb Protocol).

The J2EE application servers are implemented in Java language. The Java Virtual Machine (JVM) in which they are running has limited resources on one physical machine. Furthermore, in case of a software failure or hardware failure, their service will not be available. This is not acceptable for some business applications. They usually have requirements for high availability.

Clustered solutions can be used in order to achieve higher performance than a single JVM can provide and to improve the availability. The rest of the paper is organized as follows: chapter 2 will present general techniques for clustering J2EE, chapter 3 and 4 will present some details from the implementation done by IBM and BEA and the paper is concluded by 2 chapters with comparisons of the solutions and the conclusion.

2 Techniques for clustering EJB Applications

We will introduce general concepts and possible solutions for clustering J2EE. The clustering can be done in a 3-tier approach or a 4-tier approach as presented by Ed Roman [3]. For both of them the first tier is the browser and the last tier is the database. The 3-tier approach is the one that has the WebServer components, such as JSPs and servlets, running in the same process like the EJB components. The 4-tier variant has the Web server components and the application server components in separate processes. Ed Roman identifies some possible places where the clustering can be realized: JNDI driver, Container, Home Stub and Remote Stub. Among the bean types, the stateless session beans are the most scalable since they are not maintaining state, so they can be easily fail-over in the client code.

The stateful beans can be fail-over if the state replication feature of some of the vendor is used. However the replication comes at the expense of the performance of the server. Load balancing for stateful beans can be achieved at the remote stubs *create()* method, that can be sent to different servers in the cluster. The same result cannot be achieved at

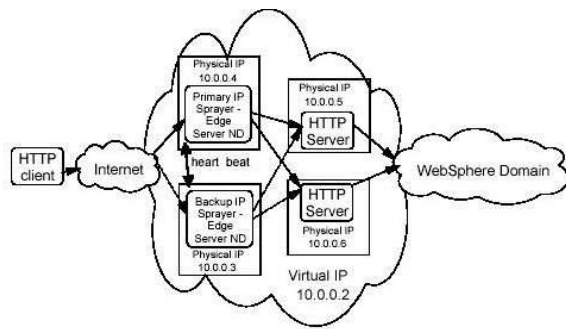


Figure 1: Highly Available WebServer configuration (reproduced from [4])

the remote stub.

For the Entity Beans, the load balancing can be achieved through home stubs similar as the session beans. Ed Roman argues that since the Entity Beans are usually wrap with a session façade pattern then this facility is little used.

3 IBM architecture for J2EE application clustering

In an e-business solution, normally the clients run a web browser and access the server side application through HTTP. The configuration of a cluster can be done in a variety of ways, here is presented one recommended configuration as is described in [4].

3.1 Clustering of the HTTP server

The configuration includes an IP sprayer that provides a gateway toward the members of the cluster. The requests are routed to the sprayer which forwards them to the Cluster members of the Web Servers. Figure 1 is representing the highly available webserver configuration. The name of the IP sprayer is Network Dispatcher [5]. The Network dispatcher runs monitor threads that supervise the status of the Web Servers and this way a failure will be isolated. Another concept that is used in the Network Dispatcher is the server affinity. For improving the performance this sends all the requests made by one client to the same server.

Another technique to cluster the HTTP requests is DNS spraying. This is based on DNS round robin technique. The disadvantage of this technique is that intermediate DNS servers may cache the entries and not update them quickly enough to provide updates for one node failure. Also, the DNS must implement failure detection for the node that is balanced.

3.2 Clustering the Web Container

The communication between the HTTP server and the web container that is running in the WebSphere Application Server is based on the routing information that is read from a "HTTP plug-in". The plug-in contains the web container servers to which the HTTP server will route the requests. The information in the plug-in is generated initially and

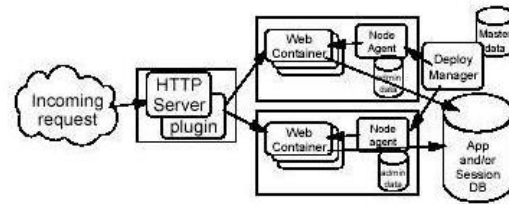


Figure 2: Highly Available Web Container configuration (reproduced from [4])

copied manually under the HTTP server files. At runtime, the information for the routing is updated by the Application Server, that has the most recent information.

3.2.1 Clustering using the HTTP plug-in configuration

Figure 2 is representing the highly available web container configuration.

The HTTP plugin can send the request to a cluster that is composed of several servers. The servers are clones so they can serve the same type of requests, and are identified by an Id. When a request comes in, the HTTP plugin does look-up in its configuration file for a cluster that can handle the request. Further, the plug-in choose a server based on the CloneID parsed from the end of the SessionID in the client request. Also, the transport communication, either HTTP or https, it is chosen based on the protocol request coming from the client.

The HTTP plug-in contains a weight for each server in the server cluster. This will be used in routing the request using a weighted round-robin routing. This process goes as follows. The clients request are sent in round-robin fashion to the servers and each time, the weight is decreased by 1. When one server reaches 0, it will not be used until all of the servers reach 0 weight, and then the counters are reset and the process is repeated.

3.2.2 Session affinity concepts

The session affinity is used to maintain the client information across multiple HTTP requests. The server maintain the session information and the client receives a cookie with the Id of the session.

3.2.3 Session persistence and failover

This is covering the mechanism of replication of the sessions to overcome a failure of one of the servers. The state information of a HTTP session can be replicated in memory or persisted in the database. The memory replication can be done to another instance of the server. This way if one server goes down the other has the state information still available and will serve the request. Since the HTTP plug-in is not aware where the session information is replicated, my interpretation is that the pair replication is preferable in WebSphere case. The replication is done periodically, in memory by using a message broker to publish the sessions information.

3.3 Clustering of the EJB Container

The clustering of the EJB containers is based on some feature implemented at the ORB level. As the communication between the client and the EJBs is done using RMI-IIOP protocol, the ORB part that is distributed to the client takes care of the workload balancing and failover. This will work only for the IBM java distributed clients. The scenario is described in the following paragraphs. The client's IBM ORB has an WorkLoadManagement part embedded. In order to get the initial routing information the request will be addressed to a Location Service Daemon (LSD). This will in turn deliver one IOR of the requested service to the client. The client can then connect to the service, but during this process the server will send back to the client information for all the servers that can serve his requests. The client updates his local information. From now on, the WLM code on the client side is able to load balance the following requests. This is called client based routing and this is how they can load balance or fail over.

In case the client is non-WebSphere type of client, then the communication is done with the LSD only. Then the LSD will send back one direct IOR to the server. Subsequent requests from the client will go to that particular server. The load balancing will be done by the server in this case. If the load is increased then it can return an error code to the client. Then the client will need to call again the LSD and a different server will be indicated.

The routing algorithm is the weighted round-robin algorithm.

3.3.1 Failover scenarios

LSD is the initial reference for the clients so it needs to be able to fail-over to another LSD. The clients will have both LSDs in the initial requests, so in case one is not responding the next one in the list will be used.

4 BEA architecture for J2EE application clustering

BEA J2EE application server is called WebLogic and can be configured in many different deployment configurations that can be applied to different usage scenarios. BEA's reference documentation [6] is identifying 3 tiers in the Web applications. First is the web tier, that is a HTTP server and serves the static pages. The Presentation tier is responsible for the dynamic content and is done using a web container for JSPs and servlets. The Object tier is the last tier and this is providing the EJBs and Java objects that are running the business logic.

I will present only 3 cases from the architectures described in [6].

The simplest architecture is depicted in Figure 3 Basic Architecture configuration.

This is a combined tier architecture, where the three tiers of the web application are running in a single WebLogic server cluster. There are advantages of this approach: The cluster is easy to administer, using just the administration

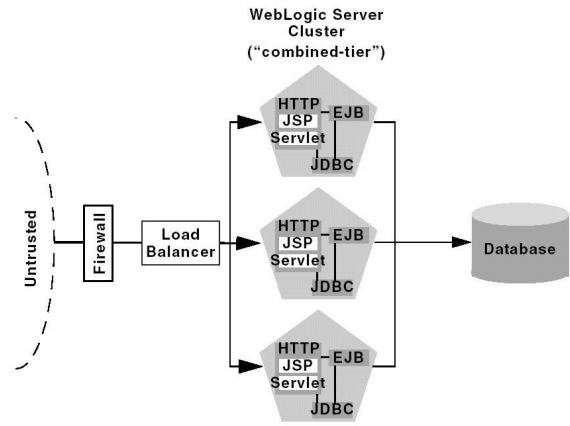


Figure 3: BEA WebLogic: Basic Architecture configuration (reproduced from [6])

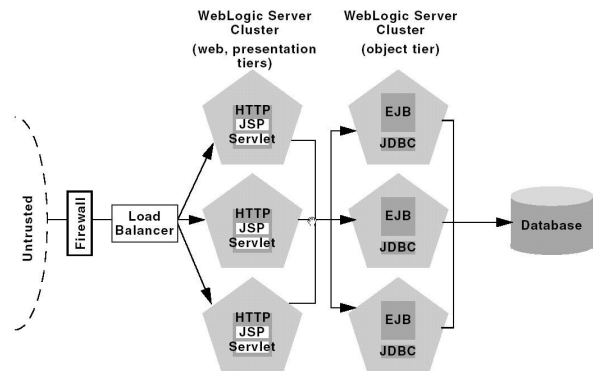


Figure 4: BEA WebLogic: Multi-tier architecture (reproduced from [6])

console of the WebLogic Server Console. The load balancing can be configured to take into account the server load. The security is robust, since the configuration of the firewall is simpler. The performance is optimal for the normal applications that use the presentation layer to access EJBs in the object layer, because the calls are made locally to the JVM. This can be more efficient than request to remote objects that incur extra network overhead.

If the object tier invoke methods on the other object tier then the solution for load balancing is to split the presentation and the object tiers of the Web application as described in Figure 4 Multi-tier architecture.

The Presentation layer consists of instances of WebLogic Server that are hosting static HTTP servers, servlets and JSPs. There is a separate object layer, that will allow the load balancing at the level of EJB methods.

The advantages of multi-tier architecture are:

1. Load balancing for method calls of the servlet to the EJBs.
2. Improved server load balancing between the HTTP, servlet and JSP load and the EJB object load.
3. Higher availability by using more servers and having less dependency between HTTP traffic and object traffic.

A similar configuration can be created using existing HTTP servers that provide the static content. This is called

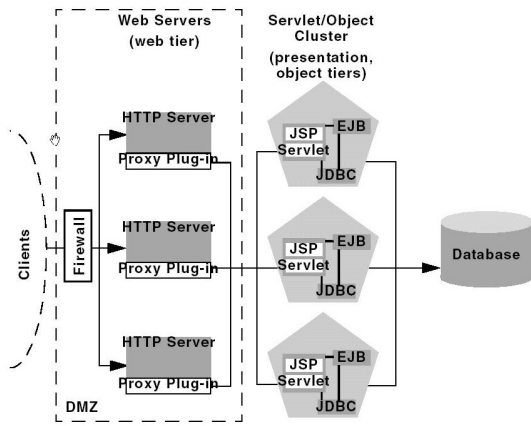


Figure 5: BEA WebLogic: Two Tier Proxy Architecture (reproduced from [6])

proxy architecture and is illustrated in Figure 5 Two Tier Proxy Architecture. One WebLogic proxy plug-in is configured to forward the JSP and servlet requests to the WebLogic cluster.

The advantages are the compatibility with the older static content and familiar firewall configuration. The access shall be granted only for the HTTP server machines. The drawbacks are the additional administration required by the HTTP server and the plug-in configuration. Also, the load balancing is limited to round-robin algorithm.

4.1 Clustering of the HTTP server

In the the basic architecture recommended by BEA, the HTTP server is included in the same JVM like the WebContainer and EJB container so the load is managed by the load balancer. It is possible to use a BEA WebLogic proxy as a load balancer in front of the BEA WebLogic Web Server [7], but in my opinion this introduce a single point of failure. The recommended alternative is a hardware load balancer. BEA supports most of the commercial solutions including F5 BIG IP and Nortel Alteon.

4.2 Clustering the Web Container

In case of the proxy configurations where the load balancing and the failover is done using a proxy plug-in for the Http server. This is similar to the already described solution used by IBM.

In case of the Basic architecture or Multitier architecture the clustering is achieved by spraying the traffic with a load balancing hardware. The load balancer must support a compatible passive or active cookie persistence mechanism. Passive cookie means that the load balancer can be configured to use the same cookie returned by WebLogic to associate a client request with a server. The active cookie persistence is a technique where the load balancer writes information on the cookie sent back to the client. Only the load balancers that don't modify the WebLogic cookie or just add information to the cookie, are supported. There is a mechanism to persist the state of the servlet into the memory or into the database.

If the active server fails, the load balancer will route the request as configured to a different server from the cluster. The new server can inspect the client cookie to find out the client id and where to get the already saved state information.

4.3 Clustering of the EJB Container

The load balancing and fail-over for the EJB objects is handled using "replica-aware stubs" which can locate instances of the EJB throughout the cluster. A replica-aware stub appears to the client as a normal RMI stub. When the client is looking-up to the JNDI tree, the implementation of a clustered object is replaced by a replica-aware stub, which is received by the client.

There are 2 levels at which the EJB can be load balanced.

Clustered EJBHome The client gets a EJBHome stub that has references to the homes on each server. When the client calls create() or find() method, it will get an object using the load balancing algorithm.

Clustering EJBObjects There are 3 different types of EJBObjects, stateless session bean, stateful session bean and entity beans.

In the case of Stateless Session Bean, the object has references to all the similar objects in the cluster. A fail method will not be redirected all the times. If the method was written in such a way that it does not affect the bean state, then has to be marked as idempotent in the deployment descriptor. Only this will enable the automatic failover in all cases.

In case of one Stateful Session Bean, the fail over requires the bean state to be replicated. In case of a failure the following calls will be redirected to a new server. The new server will recreate the Stateful Session Bean using the information from the replicated state.

The Entity Beans clustering is different for read-write entity beans and for read-only entity beans. For the read-write entity beans the load balancing and fail-over can occur only at the home level. Once the bean object was created, the reference to him is unique. Since the object is created every time there is a read, and saved in the data base every time it is committed, this bean will run only in one server.

The Read-only entity beans can be on the other hand can be load balanced on every call. Still, they are not automatically fail over in the event of a recoverable call failure.

5 Comparisons

There are many similarities between IBM WebSphere and BEA WebLogic products. Both are presenting a clustered solution that offers high availability and scalability.

Both solutions from IBM and BEA are highly configurable. BEA documentation had better example configurations for small types of deployments and IBM had many configurations on clustered hardware.

The IBM solution comes with one IP Sprayer, that can be used instead of a hardware load balancing device, making the cluster easier to deploy. The BEA solution is supporting a variety of techniques at the edge of the cluster, for the best

performance a load balancing hardware is required. In my point of view the extra components that IBM is packaging in their solution provides an advantage to the customer. You can build a more powerful HTTP cluster with their solution. The BEA solution is good also, but it requires some more investment in hardware.

Both vendors support the proxy plug-in configuration for the HTTP server. However, the IBM solution is targeted to this architecture whereas the WebLogic is concentrated on use of hardware load balancing. That is why IBM support weight based round-robin, and WebLogic supports just round-robin algorithm. Again I believe this is a slight advantage that IBM has.

To address the disadvantage of complexity of configuration the IBM simplifies the task of administering the HTTP server by providing an administrative application with a web-based interface. From the administration point of view, both application servers have an administration console server process that can be used to configure the cluster. Scripts are also a solution for automatic configuration. The administration is still very complex for the clustered solution.

In case of HTTP session persistence, the BEA model is more advanced and will allow more servers to persist the information. In WebSphere case, the preferred method is pair memory back-up and this might explain why the reference WebSphere cluster is build using pair of servers.

Both application servers are using the workload management of the EJBs at the client side. The differences in this point are small. The IBM solution is implementing this at the ORB level and BEA at the stub level. So, IBM solution is more efficient when used with the ORB provided by IBM, whereas the BEA solution is transparent. So depending on the application you implement, you might have a small advantage with BEA. In my opinion, because for a WebApplication the requests come from the presentation layer, JSPs or the servlets on the local network, this small difference is not important for a WebApplication.

WebLogic is providing load balancing at the level of EJB method call, that can improve the scalability of the cluster. Also, their solution allows you to control the routing based on the EJB method parameters using a custom CallRouter class. This is more customizable than the IBM solution.

6 Conclusion

J2EE is a collection of the best practices in the IT industry for applications for the enterprise [8]. It is an industry standard. I can observe that the techniques used for implementation are similar in both cases. The clusters are designed so that the member application servers share the load and persist in database or replicate in memory the state information. The initial request coming over HTTP are distributed to layers of servers that handle different kind of traffic. There were automatic procedures to recover from failures. The cluster configurations are complex and they grow exponentially when a big number of servers are added. The competition between vendors is to achieve capacity and speed improvements so the clusters will be able to handle more load. The future improvements are in the direction of simplification of

the administration, possible also in the tools for developing applications.

The paper that I presented is trying to simplify and to extract the most relevant scenarios for presenting a general picture of the domain. There are many things that are not covered or that are just superficially presented, but the details of the configurations are very intricate and the implementation is not documented so they are not the scope of the paper. Please consult the bibliography for some useful links.

As an overview of the application servers, BEA WebLogic seems even more configurable and scalable but IBM WebSphere ND package include an IP sprayer that can be used to achieve better high availability. My opinion is that BEA is more flexible and can be used by more diverse applications and IBM solution is more large and it can be used in large configurations. What is more important is that the J2EE technology is mature and that is demonstrated by the fact that is used with success to power today's large e-commerce sites.

References

- [1] J2EE v1.4 Documentation, 2003
<http://java.sun.com/j2ee/1.4/docs/index.html>
- [2] The J2EE Tutorial-containers, April 2002
http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Overview3.html
- [3] Ed Roman. *Mastering Enterprise Java Beans*. Second Edition, Wiley Inc, 2002.
- [4] H. Wang, M. Bransford Server clusters for High Availability in WebSphere Application Server Network Deployment, April 2003 <http://www-1.ibm.com/support/docview.wss?uid=swg27002473>
- [5] WebSphere Edge Server: Working with WebTraffic Express and Network Dispatcher SG246172, August 2001 available at <http://www.redbooks.ibm.com>
- [6] BEA WebLogic Server: Using WebLogic Server Clusters Version 8.1, October 2003 available at <http://www.bea.com>
- [7] BEA White Paper: Achieving Scalability and High Availability for E-Business Clustering in BEA WebLogic Server, March 2003 available at <http://www.bea.com>
- [8] Clustering in J2EE - Interview with Dean Jacobs, Architect BEA Systems March 2003 available at <http://www.theserverside.com>