Proxy server versus the HTTP plug-in: Choosing the best WebSphere Application Server workload management option

John Pape
WebSphere Application Server SWAT Team
IBM

Robert Westland (westland@us.ibm.com)
WebSphere Application Server WLM Architect
IBM

Since IBM® WebSphere® Application Server Version 6.0, the WebSphere proxy server has been available to provide intelligent routing of HTTP requests based on configured routing rules and performance metrics. Although not as smart as the on-demand router component of IBM WebSphere Virtual Enterprise, the proxy server can provide services above-and-beyond the traditional WebSphere HTTP plug-in implementation seen in practically all IHS-fronted WebSphere Application Server clusters. This article compares these solutions so you can make determine the best choice for your requirements.

Introduction
In clustered IBM® WebSphere® Application Server environments, HTTP and Session Initiation Protocol (SIP) requests are typically load-balanced through a combination of network layer devices and one or more HTTP server processes augmented with the WebSphere HTTP plug-in module. This setup is great for load balancing these requests to back end application servers, as well as dealing with fault tolerance in the case of server failure. But there is at least one other way to do this.

This article looks at the traditional approach to load balancing Web requests to a WebSphere Application Server cluster, and then examines an alternative method using the WebSphere proxy server. This article will provide a high level overview of each method and compare them so you can make informed decisions on which is better for your applications.

Using the HTTP plug-in
If you have a WebSphere Application Server cluster deployed into a production environment, then chances are good that you have a set of HTTP server instances placed upstream from your cluster that are outfitted with the WebSphere HTTP plug-in. This configuration provides load balancing
and simple failover support for the applications that are deployed to the cluster. The HTTP plug-in module is loaded into an HTTP server instance and examines incoming requests to determine if the request needs to be routed to a WebSphere Application Server. The plug-in examines the host and port combination along with the requested URL and determines whether or not to send the request on. If the request is to be serviced by WebSphere Application Server, the plug-in copies pertinent information (such as headers from the original request) and creates a new HTTP request that is sent to the application server. Once a response is given from the application server, the plug-in then matches up the response with the original request from the client and passes the data back. (It's actually much more complicated than that under the covers, but this level of detail is sufficient for this discussion.)

Failover support is also a crucial consideration for a WebSphere Application Server cluster. When a specific server fails, the HTTP plug-in will detect the failure, mark the server unavailable, and route requests to the other available cluster members.

Figure 1 shows a picture of a sample topology that uses the WebSphere HTTP plug-in.

**Figure 1. Sample topology with the HTTP plug-in in use**

If all you want to do is route requests using simple load balancing (such as round-robin), then the plug-in will work for you. But what if you want to setup more complex routing? What if you want to direct traffic to one cluster during the day and another during the night? What if you want your routing to be sensitive to the amount of load on a certain server? These things are possible when you replace the HTTP plug-in with a WebSphere Application Server proxy server instance.
Using a proxy server

The WebSphere proxy server was introduced in WebSphere Application Server Network Deployment V6.0.2. The purpose of this server instance is to act as a surrogate that can route requests to back end server clusters using routing rules and load balancing schemes. Both an HTTP server configured with the HTTP plug-in and the WebSphere Application Server proxy server can be used to load balance requests being serviced by WebSphere application servers, clusters, or Web servers. Both are also used to improve performance and throughput by providing services such as workload management and caching Web content to offload back end server work. Additionally, the proxy server can secure the transport channels by using Secure Socket Layer (SSL) as well as implementing various authentication and authorization schemes. The load balancing features provided by the proxy server are similar in nature to the HTTP plug-in.

The proxy server, however, has custom routing rules that the HTTP plug-in does not, plus significant advantages in terms of usability, performance, and systems management. In WebSphere Application Server V6.1, the proxy server became a first class server; it is created, configured, and managed from the deployment manager either using the console or the wsadmin commands. It uses the HA manager, on demand configuration (ODC), and a unified clustering framework (UCF) to automatically receive configuration and run time changes to the WebSphere Application Server topology.

With the release of WebSphere Application Server V7.0, a new type of WebSphere proxy server instance is available: the DMZ secure proxy. This server is similar in form factor to the original proxy server except it is better suited for deployment in the demilitarized zone (DMZ) areas of the network.

Figure 2 shows a typical proxy server topology.
Figure 2. Sample topology with the proxy server installed

You will notice that Figure 2 is very similar to Figure 1, which shows how proxy server instances can easily replace the HTTP servers in the topology. A new WebSphere Application Server custom profile can be created on these hosts, and be federated back to the deployment manager for cell 1, creating proxy servers on each node. Also notice the new shapes added in the diagram for the core group bridge, on demand configuration (ODC), and unified clustering framework (UCF) elements, which shows the tight integration with WebSphere Application Server. These elements are components inside of the cell and, together with the HA manager, provide the proxy server with the run time and configuration information needed to support load balancing and failover.

A big strength of the proxy server is its ability to utilize routing rules that are configured by the WebSphere Application Server administrator. Routing rules are bits of configuration that can be applied to the proxy that enable routing inbound requests in any manner desired. Aside from routing rules, proxies provide other capabilities, including:

- Content caching (both dynamic and static).
- Customizable error pages.
- Advanced workload management
  - Performance advisors that can be used to determine application availability.
  - Workload feedback, which is used to route work away from busy servers.
- Customizable URL rewriting.
- Denial-of-service protection.
The HTTP plug-in also provides caching of both static and dynamic content but does not have the other advanced routing capabilities of the proxy server.

Comparing configurations

Looking at Figures 1 and 2 above, the HTTP server with plug-in and the proxy server are positioned right in front of the application server tier and fit into a typical multi-tiered topology in basically the same place. Both utilize WebSphere Application Server and application server clusters (in the application tier) to provide deployed applications with scalability, workload balancing, and high availability qualities of service.

The next sections compare the similarities and differences between the HTTP server and the WebSphere proxy server in the areas of their architecture, administration, caching, load balancing, failover, routing behaviors, and routing transports. Differences between the proxy server and the DMZ secure proxy server will also be noted. At the end, Table 1 summarizes the major comparison points.

Architecture

**Proxy server**

A WebSphere proxy server is a reverse caching proxy that is included in WebSphere Application Server Network Deployment (hereafter referred to as Network Deployment). The proxy server is basically a different type of WebSphere application server that manages the request workload received from clients and forwards them on to the application server that is running applications. Because the proxy server is based on WebSphere Application Server, it inherits these advantages:

- The proxy server can be dynamically informed of cluster configuration, run time changes, and application information updates by utilizing the built-in high availability infrastructure, unified clustering framework, and on demand configuration.
- The proxy server can also use the transport channel framework, which builds specific I/O management code per platform. Using this framework enables the proxy to handle thousands of connections and perform I/O operations very quickly.

The internal architecture of the proxy server was designed using a filter framework and was implemented in Java™, which enables it to be easily extended by WebSphere Application Server. Figure 3 shows the high-level architecture of the proxy server in a Network Deployment configuration.
Figure 3. Proxy server in a Network Deployment configuration

- HTTP plug-in
  The HTTP plug-in integrates with an HTTP server to provide workload management of client requests from the HTTP server to WebSphere Application Servers. The plug-in determines which requests are to be handled by the HTTP server and which are to be sent to WebSphere Application Server servers. The plug-in uses a plugin-cfg.xml file that contains application, server, and cluster configuration information used for server selection. This file is generated on Network Deployment using the administration console and copied to the appropriate directory of the HTTP plug-in. When any new application is deployed or any server or cluster configuration changes are made, the plugin-cfg.xml file must be regenerated and redistributed to all HTTP servers.
  Figure 4 shows the high-level architecture of the HTTP server with the plug-in routing requests to Network Deployment application servers.
• DMZ secure proxy server

New in WebSphere Application Server V7.0 is a proxy server that was designed to be installed be in a DMZ, called the DMZ secure proxy server. Its architecture is the same as a standard proxy server except that functions which are not needed or not available in the DMZ are removed.

There are three predefined default security levels for the server: low, medium, and high. When configured using low security, the proxy behaves and the cluster data is updated in the same manner as a non-secure proxy. When running with medium security, it again behaves the same as the standard proxy server, except that the cluster and configuration information is updated via the open HTTP ports. When the proxy is configured with the high security level, all routing information is obtained "statically" from a generated targetTree.xml file, which contains all the cluster and configuration information required for the proxy server to determine where to route the HTTP request.

Figure 5 shows the high-level architecture of the DMZ secure proxy server routing requests to Network Deployment application servers.
Figure 5. DMZ secure proxy server routing requests to application servers

Administration

- Proxy server
  The proxy server is available in Network Deployment and is easily created on any node in which WebSphere Application Server has been installed. Because a proxy server is just a different type of WebSphere Application Server, it is automatically integrated tightly with WebSphere Application Server system management, and leverages the WebSphere Application Server administration and management infrastructure. It is very simple to use the administration commands in the console to create a proxy server, and the proxy server is automatically configured as a reverse caching proxy (see Resources). Additional configuration settings are available to fine-tune the proxy server’s behavior to meet the needs of a particular environment. These settings include options such as the number of connections and requests to a server, caching, defining how error responses are handled, and the location of the proxy logs. Setting the proper configuration and enabling caching of static or dynamic content can improve the overall performance of client requests.

For the most part, the proxy server setup and configuration is the same for all WebSphere Application Server distributed platforms and for System z. However, there is one limitation: on System z, you cannot deploy an application that could be used to serve up a defined error page for various errors.

Creating a cluster of proxy servers helps in the administration of multiple proxy servers. It is easy to create a single proxy server, get it fully configured the way you want, and to create the cluster based on the configured proxy. Once the cluster has been created, it can be used to
easily add additional proxy servers, all configured exactly the same as the original member. Having a cluster of proxy servers enables an external IP sprayer or HTTP plug-in to spray requests to the proxy cluster to eliminate single points of failure and to support load balancing.

**DMZ secure proxy server**
WebSphere Application Server provides a separate installation package to enable a proxy server to be installed into a DMZ. A DMZ proxy server requires some additional configuration and setup because there is no administrative console on the server itself. Rather, the administration of the secure proxy server is handled with scripting or by using an administrative agent. There is also support that requires the use of a Network Deployment back end cell administrative console. A DMZ proxy server profile can be created and configured, and then exported to the secure proxy profile of the DMZ image. The profile created on the Network Deployment cell is for configuration only and should not be used for any other purpose. Only the secure proxy profile on the DMZ image is fully operational.

To harden the security of the DMZ secure proxy server, these capabilities are available:

- Startup user permissions.
- Routing consideration.
- Administration options.
- Error handling.
- Denial of service protection.

You select the security level you want from one of the three predefined default values (low, medium and high) during proxy server creation. You can customize various settings, but the resulting security level will become the lowest level associated with any of the settings.

**HTTP plug-in**
The HTTP plug-in is shipped with WebSphere Application Server as a separately installed product and runs inside various HTTP servers to replace the basic routing provided by the HTTP server.

You must install an HTTP server first, and then install the HTTP plug-in. When the installation is completed, the plugin-cfg.xml file needs to be created by the WebSphere Application Server deployment manager and saved to the appropriate plug-in directory on the system where the plug-in is installed.

As workload is sent to the HTTP server, the server uses information from its configuration file to determine if the request should be handled by itself or by the plug-in. If the plug-in is to handle the request, it uses the information contained in a plugin-cfg.xml file to determine which back end application server the request should be sent to. When configuration changes occur, the plugin-cfg.xml file must be regenerated and replaced in the plug-in directory. The HTTP plug-in automatically reloads the file at a configured time interval; the default is every 60 seconds.

Since WebSphere Application Server V6.02, the HTTP plug-in can be created to be one of two different types:

- A **topology-centric** file includes all applications within a cell and is not managed by the administrative console. It is generated using the GenPluginCfg commands and must be manually updated to change any plug-in configuration properties.
• An application-centric file has a granularity that enables each application to be mapped to its specific Web or application server, and can be managed and updated using the administration console.

The HTTP plug-in has numerous configuration settings contained within the plugin-cfg.xml file (see Resources). The HTTP servers can also be configured as reverse caching proxies, but additional configuration is required after installation to support this. This type of configuration is typically used when you want clients to access application servers behind a firewall.

Load balancing and failover
Both the HTTP plug-in and WebSphere proxy server support workload management for load balancing and failover of client requests to application servers. Each has some administration control of where and how the requests can be configured, with more functionality available in the proxy server. However, there are some important differences between the two.

• Cluster support
  The main difference centers around how each gets access to cluster data. The HTTP plug-in uses "static" configuration information obtained from the plugin-cfg.xml file. The proxy server obtains cluster data dynamically so that when run time changes are made, such as starting or stopping a member, changing the weight, or member’s availability, the information is updated in the proxy server at run time. Therefore, the proxy server is able to use an application "runtime view" of the cluster during selection, so only running members of the application are included, plus any run time configuration settings that have been made.

  The HTTP plug-in uses the cluster configuration information from the plugin-cfg.xml file. This information is static and is not updated dynamically during run time. It takes an administrative act to generate a new plugin-cfg.xml file and make it available to the running HTTP plug-in. Network Deployment does permit you to configure the Web server to provide some support for automatically generating the plug-in configuration file and propagating it to the Web server. The proxy server also supports defining a generic server cluster. This is a cluster that is configured to represent a group of resources whose management falls outside the domain of WebSphere Application Server. The cluster is configured and used by the proxy server to load balance requests to these resources. Keep in mind that because these are not WebSphere Application Server servers, the same qualities of service available to a WebSphere Application Server cluster is not available.

  The HTTP plug-in does not support the generic server clusters, however you can manually edit the information in the plugin-cfg.xml file. This can provide some benefits for generic servers but it most useful for merging the plugin-cfg.xml files from two different cells so that a single HTTP server can route to multiple WebSphere Application Server cells. You can also group standalone servers or multiple cluster members into a manually-configured cluster that is only known to the plug-in. You must take extreme care when making any manual changes to the plugin-cfg.xml file. The proxy server does not permit this type of editing of cluster and routing information.

  In a DMZ secure proxy server, the security level determines whether the proxy uses dynamic or static cluster data. Using a low or medium security level, the proxy server uses dynamic...
cluster data and basically behaves as a non-DMZ proxy server. However, when running in a high security level, the routing information is obtained from the targetTree.xml file and the data is the static cluster configuration information. The targetTree.xml file is generated on the cell(s) the proxy will be sending the HTTP requests to. Any time the back end server and cluster configurations change the targetTree.xml file must be regenerated and updated in the secure proxy server.

- **Routing and selection**
  Since WebSphere Application Server V7.0, the proxy server and HTTP plug-in support two different routing algorithms:
  - The **random algorithm** ignores the weights on the cluster members and just selects a member at random.
  - The **weighted round-robin algorithm** uses each cluster member’s associated weight value and distributes members based on the set of member weights for the cluster. For example, if all members in the cluster have the same weight, the expected distribution for the cluster would be that all members receive the same number of requests. If the weights are not equal, the distribution mechanism will send more requests to a member with a higher-weighted value than to one with a lower weighted value. This provides a policy that ensures a desired distribution, based on the weights assigned to the cluster members. Valid weight values range from 0 to 20, with a default value of 2.

The proxy server selection also includes a client side outstanding request feedback mechanism called **blended weight feedback**. This uses the member weight information along with the member’s current observed outstanding request information. This feedback provides a mechanism to route work away from members that have more outstanding requests in relationship to the other members in the cluster.

- **Failover**
  Failover is provided in the event client requests can no longer be sent to a particular cluster member. This can be caused by a variety of conditions, such as being unable to establish a connection, or having the connection prematurely closed by the cluster member. When these failures occur, the proxy server and HTTP plug-in will mark the member as unavailable and route the request to another member. Both support configuration parameters that are used to fine-tune the detection and retry behavior. These parameters include things like setting the length of time for requests, connections, and server time-outs.
  Aside from connection failures and time-outs, the HTTP plug-in uses the maximum number of outstanding requests to a server to indicate when a server is hanging with some sort of application problem. Keeping track of outstanding requests and recognizing when the number of outstanding requests become higher than a configured value can be acceptable means of failure detection in many situations because application servers are not expected to become blocked or hung.
  There is one behavior difference that should be noted here: if a cluster member is stopped while client requests are being sent, the plug-in will continue to send requests to the stopped server until a request fails and the server is marked unavailable. However, the proxy server might be told that the member had been stopped before a client request is sent and remove the member from the selection algorithm. This can eliminate sending requests to
an unavailable server. Again, this is accomplished because the proxy receives its cluster information dynamically.

- **HTTP session management**
  The routing behavior of both the HTTP plug-in and proxy server are affected by how the HTTP session management and the distributed sessions are configured for the application and servers. This configuration involves session tracking, session recovery, and session clustering. When an HTTP client interacts with a servlet supporting session management, state information is associated with the HTTP session, is identified by a session ID, and is then available for a series of the client requests. The proxy server and HTTP plug-in both use session affinity to help achieve higher cache hits in WebSphere Application Server and reduce database access. When supporting session affinity, the proxy server and plug-in will try and direct all requests from a client -- for a specific session -- back to the same server on which the session was created. The default mechanism is to read the JSESSIONID cookie information passed along in the request, which contains the sessionId and serverId. This information will inform the selecting code to try and select the same member of the cluster for each request. Two other mechanisms that can be used to support session affinity are URL rewriting and SSL ID value.

- **HTTP session failover**
  To support session failover, it is important to know that session management must be enabled and that distributed sessions are configured. You can configure either database or memory-to-memory to save session data in a distributed environment. Depending on which setting is used, the HTTP plug-in and proxy server behave differently.
  Both the proxy server and HTTP plug-in keep track of which servers own which sessions by using the sessionId returned by the server. If session data is to be maintained in a database, and the server the session currently exists on fails, then one of the other cluster members will be selected, the session information will be obtained from the database, and the session will now be associated with this server.
  However, if the session data is configured as memory-to-memory, then WebSphere Application Server will select a primary and one or more backup servers to keep backup data for each session. This enables a failed session request to be sent to a server already holding backup data for the session. The proxy server automatically supports this behavior.
  The HTTP plug-in can be configured to support "hot session" failover, and when this is done, a table of session information is obtained from the WebSphere Application Server containing the appropriate mapping of server to sessionId. If a request to a specific session fails, the plug-in will make a special request to one of the other cluster members to obtain new session data, which will contain the new server to session mapping information that will be used for selection.

Table 1 summarizes the above comparisons.

**Table 1. Comparison summary**

<table>
<thead>
<tr>
<th>Subject</th>
<th>HTTP Plug-in</th>
<th>Proxy server</th>
<th>DMZ secure proxy server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>• Integrated into HTTP Server</td>
<td>• A reverse caching proxy</td>
<td>• A reverse caching proxy</td>
</tr>
<tr>
<td></td>
<td>• Uses a static configuration file, plugin-cfg.xml</td>
<td>• Specialized type of WebSphere application server</td>
<td>• Specialized type of WebSphere application server</td>
</tr>
</tbody>
</table>

Proxy server versus the HTTP plug-in: Choosing the best WebSphere Application Server workload management option
### Administration
- Application or configuration changes require a regen of plugin-cfg.xml file
- Extendable proxy filter framework
- Java implementation
- No XML files needed; dynamic run time cluster data (ODC, UCF, HAMgr)

### Caching
- Static content, such as images or HTML files
- Supports using FRCA to cache certain dynamically generated servlet and JSP files

### Load balancing
- Static routing uses plugin-cfg.xml file for routing data
- Must regenerate plugin-cfg.xml file when new cluster member is created and started
- Does not support generic server clusters with affinity
- Dynamic routing uses dynamically updated routing data
- Supports generic server clusters with passive or active affinity
- Automatically informed when new cluster member is created and started

### Failover
- Must detect server is unavailable through a failed request
- Configured setting retryInterval. Defines time to wait before retrying a failed server
- Is automatically notified when member becomes unavailable
- Automatically informed when failed server becomes available again

### Routing behaviors
- Supports round-robin selection
- Can be configured as a reverse caching proxy
- Can manually edit plugin-cfg.xml file to manipulate routing behavior
- Proxy server actions, rule expressions, routing actions, and routing rules enable an administrator to control routing behavior for various business reasons
- Custom advisor support
- Is a reverse caching proxy
- Supports round-robin and random selection

### Routing transports
- Uses standard HTTP transport protocol
- Uses the highly scalable WebSphere Application Server transport channel framework
Conclusion

While both the HTTP plug-in and the WebSphere proxy server deliver some overlapping modes of operation and capabilities, the WebSphere proxy server can be the more intelligent solution in many cases. However, "more intelligent" in this case can often times also mean more complex. Many deployments will probably continue to use the HTTP plug-in until they reach a point where improving caching or advanced routing becomes a critical requirement.
Resources

- Know your proxy server basics
- Redbook: WebSphere Application Server Network Deployment V6: High Availability Solutions
- Getting "Out Front" of WebSphere: The HTTP Server Plugin
- Information Center
  - Setting up the proxy server
  - Installing a DMZ Secure Proxy Server for IBM WebSphere Application Server
  - Configuring a DMZ Secure Proxy Server using the administrative console
  - Installing IBM HTTP Server
  - Configuring IBM HTTP Server
  - Installing Web server plug-ins
  - Plug-ins configuration
- IBM developerWorks WebSphere
About the authors

John Pape

John Pape currently works with the WebSphere SWAT Team and focuses on crit-sit support for clients who utilize WebSphere Application Server, WebSphere Portal Server, and WebSphere Extended Deployment. This role requires attention to detail as well and maintaining a “think-out-of-the-box” innovative mindset, all the while assuring IBM customers get the best support possible! John also has a development background in Domino/Lotus Notes, Microsoft C#, Java, C++, Perl, and python. John is a social media champion within IBM and an avid Linux user.

Robert Westland

Robert Westland is the architect of the WebSphere Application Server Work Load Management(WLM) component. He currently works on the WLM and HA Team at IBM Rochester Minnesota. His focus is on the architecture of the WLM support for the WebSphere Application Server. He has been working in the WebSphere Application Server organization and on the WLM team for 8 years.