IBM Software Group

WebSphere Plug-in Requests, Session Affinity, Load Balancing and Failover

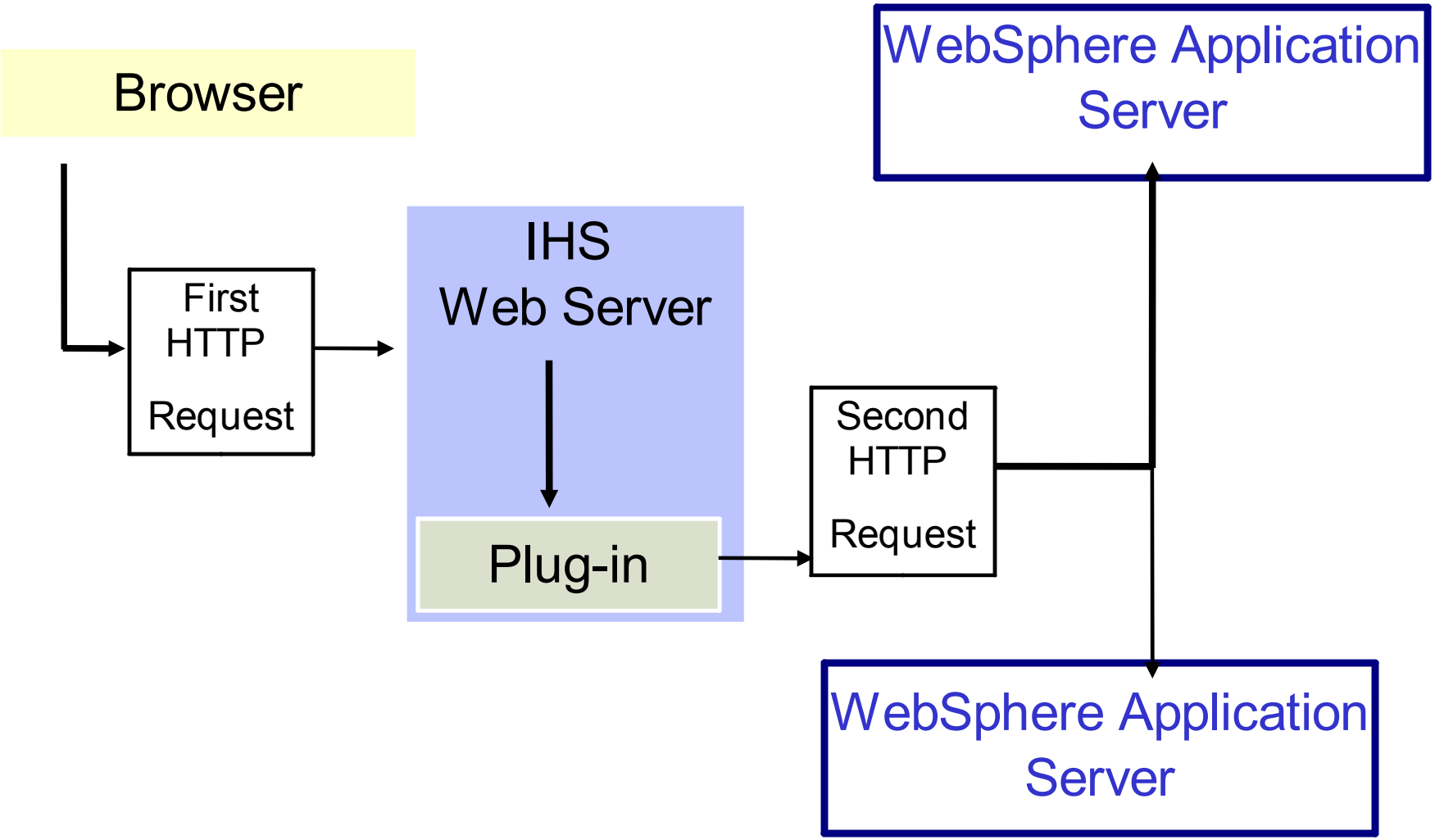
Bob Richter (brichter@us.ibm.com)
WebSphere L2 support
18 October 2010

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS™

Agenda

- Plug-in Introduction
- Plug-in Properties
- Plug-in-in SSL considerations
- Plug-in Request Processing
- Plug-in Session Affinity
- Plug-in LoadBalancing
- Plug-in Failover
- Plug-in ESi cache


Browser
First HTTP Request
IHS Web Server
Plug-in
Second HTTP Request
WebSphere Application Server
WebSphere Application Server

Plug-in Overview

- WebSphere Plug-in Main Components
  - Plug-in configuration file (plugin-cfg.xml)
    - Generated and Propagated through WebSphere Administration console
  - Plug-in LoadModules
    - Loaded by IBM® HTTP Server at Web Server startup (mod_was_ap33_http.si)
- WebSphere Plug-in Function
  - Map, Route and LoadBalance requests through Web server to the Application Server(s)

Plug-in Properties

- Plug-in properties used for mapping WebSphere servlet request:
  - Route
  - VirtualHost is contained in a VirtualHostGroup
  - URI is contained in a URIGroup
  - Transport is contained in a Server

Plug-in Properties

- ROUTE
  - Identifies the Server or Cluster target for Plug-in to send a request
  - <Route ServerCluster="Cluster1“VirtualHostGroup="default_host"UriGroup="Cluster_URIs"/>
  - Multiple Routes in the Plugin-cfg.xml
  - Plug-in will check each Route in the Plugin-cfg.xml to find appropriate route for the request

Plug-in Properties

- VirtualHostGroup
  - Contains multiple VirtualHosts which define Hostname and Ports pairs for the Web servers and Application Server.
  - VirtualHostGroup is generated from the Application and Application server VirtualHost -> Host Aliases.
  - Example of VirtualHosts in Plugin-cfg.xml
    - <VirtualHost Name="<hostName>:9080"/>
    - <VirtualHost Name="*:80"/>
  - Wildcards are commonly used for HostName

Plug-in Properties

- VirtualHost property is generated from the VirtualHost Host Alias
- Managed through WebSphere Administration Console
  - Environment-> VirtualHost -> Host Aliases
- If you change the Host Aliases associated with Application Server you must:
  - 1. Generate and Propagate the Plugin-cfg.xml to reflect the new VirtualHost (HostName and Port)
  - 2. Restart the Application Server.

Plug-in Properties

- URIGroup
  - Contains URI properties in plugin-cfg.xml which are mapped to request URI
- Server
  - Contains Transports with HTTP and HTTPS protocols which identify the Hostname and Port of the mapped Application Server.

Plug-in Properties

- Transport (HTTP)
  - <Transport Hostname=“myhost.com" Port="9082" Protocol="http"/>
- Transport (HTTPS)
  - <Transport Hostname=“myhost.com" Port="9445" Protocol="https">

Plug-in Properties

- Transport (HTTPS)
  - Includes “keyring” and “stashfile”
    - Used for SSL configuration between Plug-in and Application Server
  - <Property Name="keyring" Value="/opt/IBM/WebSphere/Plugins/config/<webserverName>/plugin-key.kdb"/>
  - <Property Name="stashfile" Value="/opt/IBM/WebSphere/Plugins1/config/<webserverName>/plugin-key.sth"/>

Plug-in SSL Considerations

- SSL between the Plug-in and Application Server may not be necessary if Plug-in is running on the same machine as the application Server
- SSL setup requires
  - Extract Application Server Personal Certificate from key.p12
  - Plug-in KDB file (plugin-key.kdb) must add the Personal Certificate from each Application Server node as a “Signer Certificate”

Plug-in SSL Considerations

- Technote to set up SSL between Plug-in and Application Server
  - http://www-01.ibm.com/support/docview.wss?rs=177&dc=DB560&dc=DB520&q1=plugin+ssl+414&uid=swg21264477&loc=en_US&cs=utf-8&lang=en

Plug-in SSL Considerations

- To disable HTTPS transport
  - <Transport Hostname="myhost.com" Port="9445" Protocol="https">
  - Servers-> Application Servers -> <ServerName> -> Web Container settings -> Web Container Transport Chains
  - Click "WCInboundDefaultSecure" for HTTPS transport
  - Uncheck Checkbox " "Enabled"
  - Save and Generate Plugin-cfg.xml

Plug-in Request Processing

- All Requests received by Web server are FIRST processed by Plug-in before Web server handling
  - Modification of the Web server requests like alias will not be handled before Plug-in process
  - Must use RewriteRule with [PT] or [R] flags for web server to modify request URL before Plug-in processing

Plug-in Request Processing

- [PT] flag will passthrough directly to Plug-in
  - Example RewriteRule using [PT]
    - RewriteEngine on
    - RewriteRule /servlet/URI/Example.do [PT]
    - RewriteLogLevel 9
    - RewriteLog logs/<fileName>

Plug-in Request Processing

- [R] flag does a full redirect back to the client with location: header containing new URL.
  - Example RewriteRule using [R]
    - RewriteEngine on
    - RewriteRule ^/$ http://hosname/uri [R]
    - RewriteLogLevel 9
    - RewriteLog logs/<fileName>
- Redirect may also be used instead of RewriteRule [R]

Plug-in Request Processing

- Using WebSphere Plug-in with Mod_alias and mod _rewrite
  - http://publib.boulder.ibm.com/httpserv/ihsdiag/plugin_alter_uri.html

Plug-in Request Processing

- Request Flow –input URL http://hostname:port/uri

1. **Loop through all ROUTE plug-in properties**
2. Create Request and Response headers
3. **Check for ESI caching**
4. **Checking for Session Affinity**
5. **Check for Round Robin or Random**
6. Check for number of Primary Application Server
7. **Check for Plug-in failover**
8. Check Status and select server if available update Request Counts
9. Find Transport based on scheme (HTTP/HTTPS)
10. Open Socket

Plug-in Session Affinity

- **Session Affinity** allows returning requests to be routed back to the same server in a cluster that handled the initial request, if that same server is available.
- Plug-in Session Affinity is handled by the WebSphere Plug-in through a special cookie enabled and configured by the Application Server
- Default name for the Application Server session cookie is JSESSIONID
- Application Server Session **JSESSIONID** cookie is enabled and set through WebSphere Administration console
  - Application servers -> <Application ServerName> -> Session management -> Cookies

Plug-in Session Affinity

- JSESSIONID cookie contains
  - CacheID
  - SessionID
  - CloneID
- Only CloneID is used by WebSphere Plug-in for Session Affinity

Plug-in Session Affinity

- For Session Affinity to work a few things must be setup
  1. Cluster environment is created
  2. JSESSIONID Cookie is enabled by the Application Server
  3. CloneID is generate to the Plugin-cfg.xml , after Cookie has been setup and Enabled in the Application Server

Plug-in Session Affinity

- Plugn-cfg.xml CLONEID

```
<ServerCluster .>
  <Server CloneID="15d2hi0gn" ConnectTimeout="0"
  ExtendedHandshake="false" LoadBalanceWeight="2"
  MaxConnections="-1" Name="rjrNode07_server1"
  ServerIOTimeout="0" WaitForContinue="false">/
  Server>
  <Server CloneID="15d2hi3ic" ConnectTimeout="0"
  ExtendedHandshake="false" LoadBalanceWeight="2"
  MaxConnections="-1" Name="rjrNode07_server2"
  ServerIOTimeout="0" WaitForContinue="false">
```

Plug-in Session Affinity

- Plug-in log entry of an Application Server response with CLONEID
  - [Fri Sep 24 14:59:45 2010] 00002d30 00002298 - DETAIL: HTTP/1.1 200 OK
  - [Fri Sep 24 14:59:45 2010] 00002d30 00002298 - DETAIL: **Set-Cookie:** **JSESSIONID**=0000A0-ItRd37WYeiLGHKH_kcFp:**15d2hi3ic**; Path=/
- CloneID is set on the response from Application Server
- Once CloneID (**15d2hi3ic**) is set in JSESSIONID Cookie then affinity to this particular server will be observed by the plug-in routing.

Plug-in Session Affinity

- CLONEID is parsed from JSESSIONID cookie and compared to the CLONEID in the Plugin-cfg.xml
- Plug-in "Trace" level log entries compare request CloneID to the Application Server CloneId in Plugin-cfg.xml
  - [Fri Sep 24 14:59:56 2010] 00002d30 00002298 - TRACE: ws_server_group: serverGroupFindClone: Comparing curCloneID **'15d2hi3ic'** to server clone id **'15d2hi0gn'**
  - [Fri Sep 24 14:59:56 2010] 00002d30 00002298 - TRACE: ws_server_group: serverGroupFindClone: Comparing curCloneID **'15d2hi3ic'** to server clone id **'15d2hi3ic'**

Plug-in Session Affinity

- Match CloneID in JSESSIONID cookie to plugin-cfg.xml CLONEID and Application Server is selected
  - Server is selected base on CloneID match
  - [Fri Sep 24 14:59:56 2010] 00002d30 00002298 - TRACE: ws_server_group: serverGroupFindClone: **Match for clone** 'rjrNode07_server2‘
  - Application Server rjrNode07_server2 is selected

Plug-in Session Affinity

- Plug-in request from client must contain the JSESSIONID cookie with the cloneID for Session Affinity to be observed.
- Plug-in "detail" log level entry, POST request and JSESSIONID cookie with CLONEID.
  - [Fri Sep 24 14:59:56 2010] 00002d30 00002298 - **DETAIL**: POST /PlantsByWebSphere/servlet/AccountServlet?action=login&updating=false HTTP/1.1
  - [Fri Sep 24 14:59:56 2010] 00002d30 00002298 - **DETAIL**: **Cookie: JSESSIONID** =0000A0-ItRd37WYeiLGHKH_kcFp:**15d2hi3ic**

Plug-in Session Affinity

- Monitor Session Affinity by enabling plug-in loglevel="Stats"
  - Plug-in "Stats" log level entries shows the affinityRequests and totalRequests for each Application Server Per Plug-in Process
    - [Tue Sep 28 09:16:58 2010] **00002d30** 000020c0 - STATS: ws_server: serverSetFailoverStatus: **Server rjrNode07_server1** : pendingRequests 0 failedRequests 5 **affinityRequests 79 totalRequests 111**
    - [Tue Sep 28 09:16:00 2010] **00002d30** 00002608 - STATS: ws_server: serverSetFailoverStatus: **Server rjrNode07_server2** : pendingRequests 0 failedRequests 2 **affinityRequests 89 totalRequests 114**

Plug-in Session Affinity

- Session Affinity can interfere with an even distribution of requests.
  - An even distribution of requests should not always be expected with Session Affinity
- If an Application Server is **not available** then All new requests will be assigned affinity to the available Application Servers
- Session Affinity will be broken when the Application Server on which the affinity has been set is not available.
- An Application Server may be marked down and a request can Failover to the next available Application Server.

Plug-in Fail-over

- Plug-in Failover, in cluster environment, occurs when the Plug-in can no longer connect or receive responses from a specific Application Server
  - The HTTP plug-in is unable to establish a connection to a cluster member's Application Server transport.
  - The HTTP plug-in detects via timeout that there has not been a response from the Application Server from previous submitted request.
  - The HTTP plug-in detects a newly connected socket that was prematurely closed by a cluster member during an active read or write.

Plug-in Fail-over

- Plug-in properties that directly affect Failover
  - ServerIOTimeout
  - ConnectTimout
  - RetryInterval
- Application Server Markdown
  - Means Plug-in will no longer attempt to route requests to an Application Server
  - Application Server is not included in routing process

Plug-in Fail-over

- ServerIOTimeout
  - Number in seconds which specifies how long the Plug-in will wait for a response from the application before timeout of the request
  - Recommended value 120 seconds
  - ZERO we are relying on the OS TCP timeout
  - +Number Plug-in will not mark down that server, but just fail over to the next one.
  - -Number Plug-in will mark down that server and Failover to the next one.

Plug-in Fail-over

- ConnectTimeout
  - Number in seconds which specifies how long the plug-in should wait for a response when trying to open a socket to the Application Server
  - Recommended Value is very small (5)
  - The default is (0) which means never time-out. In that case, the time-out is left up to the OS TCP layer, which is NOT ideal.

Plug-in Fail-over

- RetryInterval
  - value is a number in seconds which specifies how long the Plug-in should wait before attempting to route requests to an Application Server that was previously “Marked Down
  - Recommended value 60 seconds.
  - Most Mark Downs are recoverable scenarios
  - If the problem was terminal, then the time it takes to restart the Application Server could be a factor.

Plug-in Fail-over - Links

- How do the properties ServerIOTimeout and PostBufferSize affect plug-in behavior?
  - http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg21408884
- Recommend Values for Plug-in Properties
  - http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg21318463
- Understanding HTTP Plug-in Failover in a Clustered Environment
  - http://www-01.ibm.com/support/docview.wss?uid=swg21219808

Session Affinity Vs Session Persistence

- **What happens during Session Affinity Failover to another Server in the cluster?**
  - Session Affinity is replaced on Failover to a new Application Server
  - If you have no Session Persistence Mechanism then the request will be redirected to a new affinity Application Server but without **Session Data** the request will either require re-login or fail.

Session Affinity Vs Session Persistence

- CloneID will be replaced with a NEW Application Server CLONEID via Set-Cookie on Failover
  - [Fri Sep 24 14:59:45 2010] 00002d30 00002298 - DETAIL: **Set-Cookie:** **JSESSIONID**=0000A0-ItRd37WYeiLGHKH_kcFp:**15d2hi3ic**; Path=/
- Old CloneID 15d2h13ic for Server1 is replaced with CloneID **15d2hi0gn** for server2
- SessionID is not changed.

Plug-in Session Affinity

- **Session Persistence** is the means by which the Plug-in will maintain session integrity (NOT affinity) after the Plug-in has failed over to the next available Application Server.
  - As requests Failover to next Application Server if there **is persistent data shared** between Application Servers in the cluster the Failover request **will not** fail or require login

Plug-in Session Affinity

- Session Persistence - Application Server and Plug-in handle the Failover over a little differently
  - Request is routed to next available Application Server
  - New Application Server ADDs a new CloneID
  - The OLD CloneID is NOT replaced:
  - Set-Cookie: JSESSIONID=0002O9Ra5SyRIRoycNXAnKusKEb: **15d2hi0gn** :**15d2hi3ic**; path=/
- Since we have 2 CloneIDs the Plugin will always match on the initial CloneID and check Server status
- If the initial Server status is still “Markdown” then the 2nd CloneID is matched and the second Server will be selected

Plug-in Session Affinity

- Session Persistence Mechanism - 2 methods:
  - Memory to Memory
    - Session data is distributed between the configured Application Servers using Memory-to-Memory Replication
    - Must Configure memory to memory at time of creating Cluster
      - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tprs_config_p2p_default.html
  - DataBase Session Sharing
    - Session data is written to the database

Plug-in Load Balancing

- Random – Not much to discuss
- Round Robin
  - Evenly distribute work across cluster members.
  - Round Robin works best with Web servers that have a single process.
  - If the Web server is using multiple processes to run the WebSphere Plug-in, the Random option can sometimes yield a more even distribution of work across the cluster.

Plug-in Load Balancing

- Factors that affect load balance:
  - Application Server resource contention
  - Session Affinity
  - Application Server availability
- Main direct factor that affects Plug-in Round Robin Load Balancing
  - Configured LoadBalanceWeight for each Application Server
    - Managed independently in each Web server process running Plug-in

Plug-in Load Balancing

- **LoadBalanceWeight**
  - Static weights initially configured by Administrator in the plugin-cfg.xml
  - WebSphere Administration Console
    - Servers -> Clusters-> cluster Members
  - The weight associated with each server in cluster when the Plug-in distributes requests using Round Robin load balancing
    - LoadBalanceWeight="8"

Plug-in Load Balancing

- Terms and factor related to Round Robin LoadBalancing
  - Greatest Common Divisor (GCD)
  - Internal Routing Table Weights
  - Sticky and Non-Sticky requests
  - Algorithm

Plug-in Load Balancing

- **Greatest Common Divisor (GCD)**
  - Uses to determine the Plug-in Internal Routing Table Weights from the configured LoadBalanceWeights.
  - For Example LoadBalanceWeight (8,6,18)
    - GCD = 2
- **Internal Routing Table Weights**
  - Derived from LoadBalanceWeight and GCD
  - For example LoadBalanceWeights (8,6,18) GCD 2
    - Internal Routing Table Weights(4,3,9)
  - Internal router table weights are scaled down LoadBalanceWeights

Plug-in Load Balancing

- **Sticky and Non-Sticky requests**
  - Sticky request
    - Session Affinity is established and request needs to be route to affinity Server
  - Non-Sticky Request
    - Session Affinity is NOT established
    - New request
- **Algorithm** for balancing based on historic distribution of requests
  - [(w + m * s) > 0]

Plug-in Load Balancing - Process

- Plug-in LoadBalance Processing
  - Initial Internal Router Tables Weights are set using GCD and LoadBalanceWeights
  - As each request is routed to a cluster member (application server)
    - Internal router table weight of the application server gets decremented by 1.
  - Non-sticky requests are not routed to any cluster member whose present Internal Router Table Weight is ≤ 0.
  - Sticky request <u>are routed</u> to a cluster member whose Internal Router Table Weight is ≤ 0
    - Potentially decreases the cluster member weight to a negative value (-).

Plug-in Load Balancing – Process (Cont'd)

- When the Internal Router Table Weights of **all** the cluster members are ≤ 0
  - Plug-in component **resets** ALL cluster members Internal Router Table Weights (when all weights are ≤ 0)
  - Plug-in resetting of Internal Router Table Weights may **not** take the Internal Table Router Weights to their original starting values!

Plug-in Load Balancing – Process (Cont'd)

- Algorithm is used to recalculate the Internal Router Table Weights
  - Goal - adjust the Internal Router Table Weights to overcome any current uneven distributions
- Find minimum number "m" such that equation is true for all Application Servers
  - $(w + m * s) > 0$
    - w = current Internal Router Table Weight
    - s = starting Internal Router Table Weight

Plug-in Load Balancing - Example

- Round Robin LoadBalanceWeight example using 3 servers in cluster (server1, server2 , server3)
  - Configured Weights
    - Server1 (8)
    - Server2 (6)
    - Serve 3 (18)
  - Router Table Weights (4,3,9)
    - Server1 (4)
    - Server2 (3)
    - Server3 (9)
  - GCD=2
- Requests are routed over a period of time but based on affinity, resources and other factors the current Internal Router Table Weights are:
  - Server1 (-20) - Negative because we continued to decrement for each request after “0”
  - Server2 (-40) - Negative because we continued to decrement for each request after “0”
  - Server3 (0)
- Note: Based on the current Internal Route Table Weights, we can see that we routed more requests to server1 and server2 than server3

Plug-in Load Balancing - Example

- Algorithm (w + m *s) > 0
  - Find minimal number “m”
  - w is the current Internal Router Table Weight (-20,-40,0)
  - S is the starting Internal Router Table Weight (4,3,9)
  - M=14
- Internal Router Table Weights are recalculated using the algorithm
- Results are new weights
  - Server 1 (-20 + 14 * 4) = 36
  - Server2 (-40 + 14 * 3) = 2
  - Server 3 (0 + 14 * 9) = 126

Plug-in Load Balancing - Summary

- Example Summary
  - Initial desired Internal Router Table Weight was (4, 3,9)
  - Actual distributions shows (-20 , -40 , 0)
    - This means the most number of request went to
      - Server1 -20 (+ 4 initial Router Weight) = 24 requests)
      - Server2 -40 (+3 Initial Router Weight) = 43 requests)
      - Server3 0 (+9 Initial Router Weight) = 9 requests)
- To adjust for the unbalanced actual requests we will now
  - Favor Server 3 - new Weight 126
  - Then Server 1 - new Weight 36
  - Last Server2 - New Weight 2

Plug-in Load Balancing (Round Robin)

- General Reference for setting up clusters and Workload Management
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tprf_tunewebserv.html

Plug-in Load Balancing

- Plug-in specific links on Round Robin Load Balance
  - http://www-01.ibm.com/support/docview.wss?rs=0&uid=swg21219567
  - http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg21318463
- Redbook old but good
  - http://www.redbooks.ibm.com/abstracts/TIPS0235.html

Plug-in Load Balancing

- **Taking Application Server in Cluster Gracefully Offline**
  - When LoadBalanceWeight is set to zero, this is a signal to the Plug-in to stop sending **new** requests to that Application Server
  - This has the effect of taking an Application Server in cluster gracefully offline.
  - Sticky requests on that server will continue
    - As those sessions get terminated (because the user explicitly logs out or because the session idle timeout is triggered), the server, in time, will no longer have any active sessions.
- See this Technote for details:
  - http://www-01.ibm.com/support/docview.wss?uid=swg21247728

ESI – Edge Side Include Caching

- The ESI Cache is a cache within the Web server Plug-in that is used to cache static content being served from an application in WebSphere Application Server.
  - Static Content ONLY
    - gif, css, js
  - Dynamic Content (servlets, JSPs) are not cached in the ESI cache

ESI – Edge Side Include Caching

- To enable ESI Cache
  - <Property Name="esiEnable" Value="true"/>
    - ESI Cache is enabled by default
  - <Property Name="esiMaxCacheSize" Value="1024"/>
    - Maximum size of the cache in 1K byte units
    - default is 1M or (1024k)

Plug-in ESI – Edge Side Include Caching

- Each Web server process has a separate plug-in instance with a separate ESI cache
  - Can be variables hits and misses
  - Extra memory
- Monitor static caching in the Plug-in ESI cache using WebSphere Application CacheMonitor
- To Clear Plug-in ESI cache contents
  - Web server is restarted
  - Default Timeout of 300 seconds
  - Through the CacheMonitor WebSphere Application

Plug-in ESI – Edge Side Include Caching

- To install and configure CacheMonitor
  - Both the CacheMonitor.ear (Dynamic Cache Monitor) and DyncCacheEsi.ear (DyncCacheEsi) applications must be installed.
  - Set esiInvalidationMonitor =true in Plugin-cfg.xml
    - Servers-> <WebServerName> -> Plug-in Properties
    - Specifies if the ESI processor should receive invalidation from the application server
    - Generate and Propagate the Web server Plugin-cfg.xml
- NOTE: **cachespec.xml** is not necessary for enabling of the monitoring and managing of static content in the Plug-in ESI cache

Plug-in ESI – Edge Side Include Caching

- To display the static Plug-in ESI cache:
- http://<hostName>:port/cachemonitor/ where
  - Hostname is hostname of the Application Server
  - Port is the port of the Application Sever
- For Statistics
  - Click “Edge Statistics”
  - Click button “Refresh Statistics
- For Content
  - Click Edge Statistics
  - Click button “Contents” (at the bottom of panel)
  - Click button “Refresh Contents”

Plug-in ESI – Edge Side Include Caching

- To clear cache content and reset statistics using CacheMonitor
  - Click Edge Statistics
  - Click button “Clear cache”
  - Click button “Reset Statistics”
- NOTE: Always Click
  - “Refresh Statistics”
    - For Current Stats
  - “Refresh Content
    - For Current Contents

Plug-in ESI – Edge Side Include Caching

- To customize the timeout interval to clear ESI cache, use custom properties:
  - **com.ibm.ws.cache.CacheConfig.alwaysSetSurrogateControlHdr=true**
  - **com.ibm.servlet.file.esi.timeOut=120**
- To add JVM custom property go to WebSphere Administration Console task:
  - Application servers > <Application Server name > Process Definition > Java™ Virtual Machine > Custom Properties
- NOTE: Some documentation indicates this should be added to JVM arguments. This does not appear to be correct.

ESI – Edge Side Include Caching

- Check this technote for potential Limitations and performance considerations:
  - http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&dc=DA400&uid=swg27015501&loc=en_US&cs=UTF-8&lang=en&rss=ct180websphere

We Want to Hear From You!

Tell us about what you want to learn

Suggestions for future topics
Improvements and comments about our webcasts
We want to hear everything you have to say!

Please send your suggestions and comments to:
wsehelp@us.ibm.com

Questions and Answers

Additional WebSphere Product Resources

- **Plug-in propagation fails with PLGC0063E and PLGC0049E when copying to remote Web server**
  http://www-01.ibm.com/support/docview.wss?uid=swg21231515
- **TroubleShooting: Plug-in generation and propagation for V6.0 and V6.1**
  http://www-01.ibm.com/support/docview.wss?uid=swg21207587
- **MustGather: IBM HTTP Server V6.0 and V6.1 administrative problems**
  http://www-01.ibm.com/support/docview.wss?rs=177&context=SSEQTJ&dc=DB520&dc=DB560&uid=swg21285057&loc=en_US&cs=UTF-8&lang=en&rss=ct177websphere
- **MustGather: Plug-in propagation problems in WebSphere Application Server V6.0 and 6.1**
  http://www-01.ibm.com/support/docview.wss?uid=swg21254319
- **TroubleShooting: IBM HTTP Server Administrative Server for V6.0**
  http://www-01.ibm.com/support/docview.wss?uid=swg21229375
- **TroubleShooting: WebSphere HTTP Server administrative console problems for V6.1 and V6.0**
  http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg21261119

ESI Links

- **Potential limitations and considerations when downloading static content from the WebSphere Application Server**
  - http://www.ibm.com/support/docview.wss?uid=swg27015501
- **Configuring Edge Side Include caching**
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.soafep.multiplatform.doc/info/ae/ae/tdyn_esiedgecaching.html
- **How does mod_cache interact with the WebSphere Plug-in?**
  - http://publib.boulder.ibm.com/httpserv/ihsdiag/questions.html#cacplugint
- **Dynamic Caching**
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/welc6tech_dyn.html
- **Troubleshooting the dynamic cache service**
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/tdyn_probd.html
- **Toubleshooting tips for the dynamic cache service**
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/rdyn_trb.html

Performance/Tuning links references

- Recommended values for Web server plug-in config
  - http://www.ibm.com/support/docview.wss?rs=180&uid=swg21318463
- Tuning IBM HTTP Server to maximize the number of client connections to WebSphere Application Server
  - http://www.ibm.com/support/docview.wss?rs=0&uid=swg21167658

Appendix:
Plug-in Administration

Plug-in Administration

- Plug-in GENERATE – Plugin-cfg.xml
  - All plug-in properties are GENERATED from the Application Server or the Deployment Manager repository files.
  - Maps Web servers and Application Servers to Applications
  - Defines Application Server and Web server virtual Hosts
  - Defines Application URIs

Plug-in Administration

- Plug-in is Generated to WebSphere repository usually associated with Web server
  - Servers -> Web Servers -> click Generate button
    - Config/cells/<cellName>/nodes/<nodeName>/servers/<WebServerName>
- There is a Global Plugin-cfg.xml but the administration of this file is limited and not strategic.
  - “Update Global Web server Plug-in configuration”
    - Directory-> config/cells
  - Update to some Plug-in properties is not supported
  - No Propagation is supported for Global Plugin-cfg.xml

Plug-in Administration

- To Update Global Plugin-cfg.xml properties
  - In order to update plugin-in property values in the existing global plugin-cfg.xml:
    - First DELETE the Global Plugin-cfg.xml located in config/cells directory.
    - Issue GenPluginCfg.sh(bat).

Plug-in Administration

- Plug-in Propagation COPIES the plugin-cfg.xml for associated Web server to target Web server directory
  - Httpd.conf Directive **WebSpherePluginConfig**
  - Default location of plugin-cfg.xml
    - Plug-in InstallRoot/config/<WebServerName>
- Plug-in property "**RefreshInterval**"
  - Plugin-cfg.xml is reloaded value set for this property.
  - Default is 60 seconds.
  - Checks for File TimeStamp to change every 60 seconds, if changed , then reloads

Plug-in Administration

- AUTO GENERATE (default) setting in the PluginProperties panel for plugin-cfg.xml:
  - **No external notification is made** when Generate fails as result of AUTO GENERATE.
  - Auto GENERATE of the plugin-cfg.xml will only log errors in the deployment Manager log
- AUTO PROPAGATE (default) setting in PluginProperties panel for plugin-cfg.xml:
  - **No external notification is made** when Propagate fails as result of AUTO PROPAGATE.
  - Auto Propagate of the plugin-cfg.xml will only log errors in the deployment Manager log

Merge Multiple Plug-in configuration files

- Merging two or more Plug-ins
  - http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=plug+merge&uid=swg21139573&loc=en_US&cs=utf-8&lang=en