One Pager: GlassFish 3.1/Load Balancer Plugin

Table of Contents

1. Introduction

1.1. Project/Component Working Name

GlassFish 3.1/Load Balancer Plugin

1.2. Name(s) and e-mail address of Document Author(s)/Supplier

Kshitiz Saxena : kshitiz.saxena@sun.com

1.3. Date of This Document

2010-05-20

Date	Revision	Comments
2010-05-20	1.0	Initial draft
2010-06-09	1.1	Incorporated review comments of Joe and Mahesh. Also added an opt consistent hash algorithm. All changes are marked in blue.
2010-06-22	1.2	Incorporated suggestions on Nazrul for upgrade and default values of changes are marked in shade of blue.

2. Project Summary

2.1. Project Description

In GlassFish 3.1 the support for clustering of application server instances is being introd need to have a load-balancer to front-end the cluster of instances. GlassFish 2.1.1 alrea same will be leveraged for GlassFish 3.1. The load-balancer for GlassFish is a nativ installed on a web-server. After installing load-balancer plugin on web-server and configu distributing requests across cluster of GlassFish instances and handle fail-over among ma of web-server are supported by load-balancer plugin. The list includes Sun Java System server and Internet Information Service(IIS). Being a external component, which is installe no functional changes required to make it work with GlassFish 3.1.

The load-balancer plugin gathers information on back-end cluster using a xml file usually xml. The load-balancer xml need to be generated by GlassFish admin framework. GlassFish 2.1.1 will be supported in GlassFish 3.1 to provide same user experience.

The load-balancer plugin need to be installed on web-server and then needs certain amo
server. The process is tedious and error-prone if done manually. An installer which a
required for load-balancer plugin installation and configuration will be provided to the use

The session replication framework in GlassFish is based on replica partner. So in case a
goes down, it will preferred to fail-over request to instance which acts as replica. Thi
throughput even in case of instance failure with session replication enabled.

2.2. Risks and Assumptions

Load-balancer plugin pushes certain information as proxy headers alone with request. Gla
be able to interpret these headers and populate request object appropriately. Also GlassFi
to stamp sticky information provided by load-balancer plugin in proxy header either as
correct functioning of load-balancer plugin.

The new feature of preferred fail-over instance will require sharing information of rep
session.

3. Problem Summary

3.1. Problem Area

With introduction of clustering feature in GlassFish 3.1, there is a basic requirement of hav
end the cluster. A load-balancer is required to distribute incoming requests to instances
throughput. It need to detect instance's health and route requests only to healthy instan
availability.

In most production environment, a web-server tier frond-ends a application-server tier. S
static content from web-server tier itself, sending only requests for dynamic content to
increasing the system throughput. So a load-balancer which can be installed and configu
be a ideal choice.

To provide high availability in true sense, session data need to be replicated to another
failure scenarios. Replica will hold session information in case current instance handlin
information can be retrieved by any instance in cluster, however to further optimize rec
failed over to replica instance itself. This will provide higher throughput even in case of in

3.2. Justification

High Availability(HA) is one the release driver for GlassFish 3.1 and load-balancer is one

4. Technical Description

4.1. Details

4.1.1. Load-balancer plugin configuration

The load-balancer acts as a front-end to GlassFish cluster/instances. It will distribute lo
instances for applications deployed on them. It gathers information on back-end GlassFis
as load-balancer xml. The load-balancer xml provides following information :

1. The clusters front-ended by this load-balancer. It can front-end multiple clusters hav
   deployment of applications.
2. Instances in each cluster with following details :
    1. HTTP/HTTPS listeners for the instance
    2. State of the instance - enabled/disabled
    3. Weight associated with each instance. This is useful in case of deployment ov
3. Applications deployed in each cluster with following details :
    1. Context root of the web application
    2. State of the instance - enabled/disabled
4. Health checker used to detect instance recovery
    1. HTTP request-uri used to ping failed instance
    2. Interval between health check pings
    3. Response time-out for health check ping

To generate a load-balancer xml, a load-balancer view need to be defined in domain.
association between load-balancer and cluster/standalone instances among other deta
domain.xml from GlassFish 2.1.1. Same elements and structure will exist for GlassFish 3.

The load-balancer configuration element for cluster and standalone instances is prov

```
<lb-configs>
  <lb-config https-routing="false" monitoring-enabled="false" name="cluster-
interval-in-seconds="60" response-timeout-in-seconds="60" route-cookie-enabl
    <cluster-ref lb-policy="round-robin" ref="cluster2">
      <health-checker interval-in-seconds="30" timeout-in-seconds="10" url="/
    </cluster-ref>
    <cluster-ref lb-policy="round-robin" ref="cluster1">
      <health-checker interval-in-seconds="30" timeout-in-seconds="10" url="/
    </cluster-ref>
  </lb-config>
  <lb-config https-routing="false" monitoring-enabled="false" name="standal
poll-interval-in-seconds="60" response-timeout-in-seconds="60" route-cookie-e
```

<server-ref disable-timeout-in-minutes="30" enabled="true" lb-enabled="tr
instance">
<health-checker interval-in-seconds="30" timeout-in-seconds="10" url="/
</server-ref>
</lb-config>
</lb-configs>

Note :The load-balancer config can either have server reference or cluster reference both. It is assumed that information on instances within cluster and deployed applica referencing the cluster configuration.

The load-balancer element having reference to associated lb-config is provided belo
<load-balancers>
<load-balancer auto-apply-enabled="false" lb-config-name="cluster-http-lb-
http-lb">
<property name="device-host" value="test.com"/>
<property name="device-admin-port" value="8080"/>
</load-balancer>
</load-balancers>

This will be required for automatically pushing load-balancer xml over the wire to v element is not needed if command ***apply-http-lb-changes*** is not supported.

Also each server-ref will have ***lb-enabled*** attribute, to indicate whether an instance i load-balancer perspective.
<server-ref disable-timeout-in-minutes="30" enabled="true" lb-enabled="true"

Similarly all application-ref will also have lb-enabled attribute, to indicate whether a disabled from load-balancer perspective.
<application-ref disable-timeout-in-minutes="30" enabled="true" lb-enabled="t
application"/>

Also user can assign weight to a particular instance. This will be stored as attribute
<server config-ref="cluster1-config" lb-weight="400" name="instance1" node-

Below is list of commands to create load-balancer view in domain.xml :

Commands	Details

create-http-lb-config	Creates the lb-config element with provided values.
create-http-lb-ref	Creates cluster-ref or server-ref under lb-config element with provided valu
create-http-health-checker	Creates health-checker element with provided values
enable-http-lb-server	Sets the lb-enabled flag to true for the given instance. If cluster name is use cluster lb-enabled flag is set to true.
enable-http-lb-application	Sets the lb-enabled flag to true for the given application.
delete-http-lb-config	Deletes the given lb-config
delete-http-lb-ref	Deletes the given cluster-ref or server-ref. All instances need to be disabled successfully.
delete-http-health-checker	Deletes the given health-checker
disable-http-lb-server	Sets the lb-enabled flag to false for the given instance. If cluster name is use cluster lb-enabled flag is set to false.
disable-http-lb-application	Sets the lb-enabled flag to false for the given application.
configure-lb-weight	Configures the weight for a particular instance.
list-http-lb-configs	Lists all the lb-config elements
create-http-lb	Creates the load-balancer element. It can create lb-config, cluster-ref/server-instance/application by running a single command. This command will be r changes command is supported.
delete-http-lb	Delete the given load-balancer. This command will be needed if apply-http- supported.
list-http-lbs	Lists all the load-balancer elements. This command will be needed if apply- is supported.

Main purpose of creating load-balancer view in domain.xml is to facilitate generation of lo
consumed by load-balancer plugin. Below is the list of commands to achieve the same :

Commands	Details
export-http-lb-config	Generates the load-balancer xml corresponding to given lb-config. User car load-balancer name. It will be exported to provided file name.
apply-http-lb-changes	Generates the load-balancer xml corresponding to given load-balancer and configured web-server host name and port number. Web-server require som this feature to work. This command may not be supported in GlassFish 3.1.

The generated load-balancer must conform to ***sun-loadbalancer_1_2.dtd***.

For more details on these command refer to GlassFish 2.1.1 documentation.

In addition to support for above commands, there will be following changes with respect t

1. The default value of ***lb-enabled*** attribute for a newly created instance will be ***true***. I default value was false.
2. The default value of ***lb-enabled*** attribute for a newly deployed application will be ***tr*** default value was false.
3. An additional parameter will be added to asadmin ***create-instance*** command to enal attribute to desired value.
4. An additional parameter will be added to asadmin ***deploy*** command to enable user t desired value.

4.1.2. Installer for load-balancer plugin

The load-balancer plugin need to be installed on web-server and then web-server need to
correctly, web-server will use the plugin to handle the requests. The installation and confi
error prone if done manually. To ease out this process, a tool will be provided to enable u
the loadbalancer plugin on the web server.

In GlassFish 2.1.1., a tool called GlassFish LoadBalancer Configurator was developed
capability. The same tool will be leveraged for GlassFish 3.1 as well. It is a IzPack based
to execute. It accepts user inputs and perform installation and configuration tasks based
installation and configuration in a two step process :

1. In first step, loadbalancer plugin is exploded and installed
2. In second step, web server is configured to use loadbalancer plugin

2. In second step, web server is configured to use loadbalancer plugin

The tool also provides Post Installation steps, if any. User can also generate a automation silent install at a later point of time. It also provides an uninstall script, which can be used configuration and plugin from the web server.

The tool will also provide support for upgrade. User will be able to perform upgrade fron 3.1 using this tool. It will detect that load-balancer configuration already exists in web-ser load-balancer plugin binary. In future, it can also be used to distribute new load-balance fixes or new features.

4.1.3. Preferred fail-over instance

Load-balancer detects instance failure and fails over requests being serviced by that instance thus providing high availability. This newly selected instance is called fa implementation of load-balancer plugin, the selection of fail-over instance is done using r round-robin algorithm is in general stateless, there is no preferred fail-over instance.

The session replication framework in GlassFish replicates session to a partner to provide The partner is know as replica. In case of instance failure, the session can be restored o When request is failed over to another instance, it needs to figure out the replica t replication now uses consistent hash algorithm to identify the replica. In case identifie session, it resorts to the broadcast mechanism to identify the replica. This will happen for a failed instance resulting in lot of network traffic and loss of throughput. However i intelligent decision, based on some information available in request, it can directly route The replica instance can directly load session from its local cache without a need for bro provide better performance and throughput even in case of instance failure.

Option 1 : A contract between session replication framework and load-balancer t session

The information of replica must be available in incoming request to enable load-balancer for handling session fail-over. This information can be present either as cookie or parame balancer plugin depends on web-container to stamp session stickiness information on resp subsequent requests. Now it will further depend on web-container to stamp replica inform

One important point to note here is that information of instance currently handling session Load-balancer plugin actually generates unique identification for each instance and use t name in clear text. Thus it will expect replica instance information being stamped to be must be same as one generated by load-balancer plugin for that instance. Due to th approaches to implement this feature.

Approach 1 - Load-balancer plugin selects replica partner : When load-balance

not belonging to any session), it will select an instance to service the request. It wil using same round-robin algorithm to act as replica partner. The replica instance na unique identification, will be added as proxy headers on request. The request GlassFish instance. The session replication framework can extract replica instanc and use it as replica partner. Also web-container can extract unique identificatic proxy header, and stamp that information either as cookie or parameter on uri(ur failure, load-balancer will select instance corresponding to replica instance inform act as a fail-over instance.. It will also select a new replica partner and add it as prox approach, a replica partner will be selected for all new requests, even those which d

Approach 2 - Session replication framework selects replica partner : The lc modify a new request in any manner. In case session is created by the new req framework will select an instance to act as replica. The information is made avail web-container then generates a unique identification for that instance using the balancer plugin. This will require that logic to generate instance identification is du well. Also it need to be ensured that both implementations remain same even in fu load-balancer will select instance corresponding to replica instance information av fail-over instance. The session replication framework will select a new replica partn

Option 2 : Using consistent hash algorithm in both session replication framework an
Using a consistent hash algorithm across load-balancer and session replication framework this scenario. There will be no contract between load-balancer and session replication fra independent of each other. However both of them need to use identical implementation of

This approach was used in SailFin and can be used here as well. The load-balancer uses distribute incoming traffic. The consistent hash algorithm is stateless in nature and thus y key. This implies for a given key, it will select same instance on load-balancer as framework. The session replication framework will use same algorithm to select a replica p

Main drawback of this approach is that distribution will not be as fair as round robin mecl it provided close to round robin distribution.

4.1.4 Supported platforms

Supported platforms by load-balancer plugin will be subset of platforms supported by Gl to support platforms already supported in GlassFish 2.1.1. As of now there is plan to Below is the list of supported platforms.

- Solaris 9/10
- RHEL 3/4/5

- Windows 2003 Advanced Server Edition

4.1.5 Supported web-servers

It will continue to support web servers which were supported by load-balancer plugin in G
list of supported web servers :

- Sun Java System Web Server 6.1SPx/7.x
- Apache HTTP server 2.0.x/2.2.x
- Internet Information Services(IIS) 5.0/6.0

4.2. Bug/RFE Number(s)

N.A.

4.3. In Scope

All features described in this one pager are with-in scope of this document.

4.4. Out of Scope

N.A.

4.5. Interfaces.

4.5.1 Public Interfaces

Interface	Comments
asadmin commands	Newly added asadmin command create load-balancer view in domain.xml and also to generate load-balancer xml
load-balancer xml	Load-balancer xml to be consum by load-balancer plugin
sun-loadbalancer_1_2.dtd	DTD for load-balancer xml

4.5.2 Private interfaces

None

4.5.3 Deprecated/Removed Interfaces

None

4.6. Doc Impact

Load-balancer documentation will be part of high availability guide. Documentation existi
be reused. It will require man pages for admin commands(CLI as well as GUI). Need add
preferred fail-over instance feature.

4.7. Admin/Config Impact

A set of new commands are being added to GlassFish 3.1 for creating http load-balancer e
then to generate load-balancer xml based on that. These commands will be available in bo

4.8. HA Impact

Load-balancer is a core feature of high-availability.

4.9. I18N/L10N Impact

The i18n/l10n impact consists of making sure that the output from the new sub commands
already established for administrative commands.

4.10. Packaging, Delivery & Upgrade

4.10.1 Packaging

Load-balancer plugin will be packaged a jar file generated using IzPack.

4.10.2 Delivery

IzPack based bundle will be delivered as an add-on feature.

4.10.3 Upgrade and Migration

Since load-balancer is a standalone component outside GlassFish, there is no upgrade or n

4.11. Security Impact

Load-balancer plugin is installed on web-server and thus utilizes security framework of we

it adversely.

4.12. Compatibility Impact

This feature is compatible with older version of GlassFish. All features except newly intro
fail-over instance will continue to work.

4.13. Dependencies

4.13.1 Internal Dependencies

1. The GlassFish 3.1 administration framework will be used to write all admin comma
2. Dependency on web-container to parse proxy headers and populate request object a
3. Also web-container is responsible for stamping sticky information to maintain sessic
   be enhanced to stamp replica information as well.
4. For preferred fail-over instance feature to work correctly, there is a need to maintain
   balancer plugin and replication framework.

4.13.2 External Dependencies

GlassFish LoadBalancer Configurator is a IzPack based installer and configurator. Thus it

4.14. Testing Impact

The feature need to be tested in fashion similar to GlassFish 2.1.1. All functional test cases
to be executed.

The GlassFish Loadbalancer Configurator provided to install and configure load-balancer
exhaustively tested.

The preferred fail-over instance feature introduced in GlassFish 3.1 will require writing an
test cases.

5. Reference Documents

1. GlassFish 2.1.1 documentation on load-balancer plugin
2. GlassFish 2.1.1 documentation on load-balancer administration
3. GlassFish 2.1.1 documentation on load-balancer admin commands
4. White paper on GlassFish load-balancer
5. Blog on GlassFish load-balancer plugin

6. Schedule

6.1. Projected Availability

Item #	Date/Milestone	Feature-ID	Description	QA/Docs Handover?
1	MS1	N.A.	Load-balancer one pager describing features and implementation details	No
2	MS3	LBREC-001	Admin command to create load-balancer elements in domain xml	Yes
3	MS3	LBREC-002	Admin command to generate load-balancer xml	Yes
4	MS4	LBREC-004	Preferred fail-over instance	Yes
5	MS5	LBREC-003	Installer support for Load-balancer plugin	Yes
6	?	LBREC-005	Pushing load-balancer xml over the wire to web-server	Yes