Setting Up a Cluster of Virtualize Servers Behind a Load Balancer

This topic provides recommendations on how to configure a cluster of Parasoft Virtualize servers to achieve benefits such as:

- **Horizontal scaling (horsepower)**: Commonly used to optimize performance testing and cloud-based infrastructures for Continuous Integration / Continuous Delivery.
- **Fault tolerance**: Commonly used to increase uptime and/or facilitate disaster recovery for business-critical backend systems.

Sections in this topic include:

Prerequisites

- Multiple Virtualize servers, all with the exact same version (including the same service pack, if applicable) and all licensed for Virtualize command line (virtualizecli)—clustering is not supported by the container-deployable Virtualize server
- A shared file system
- Multiple servers to host multiple Virtualize instances

Load Balancer Hardware and Software Configuration

In all circumstances, you need to configure your load balancer to go to each Virtualize server.

If you are using Parasoft CTP (or any other tool that provisions/copies/modifies/… assets through the Virtualize REST API), you will also need to perform the additional configuration outlined in the following sections.

Note that the following instructions apply to any load balancer. F5 Local Traffic Manager (LTM) examples are provided to give you an idea of how these general guidelines might be carried out on a specific load balancer.

Configuring Source Address Affinity Persistence

This section explains how to configure source affinity persistence for Parasoft CTP. The same principles can also be applied to configure other any other tool that provisions/copies/modifies/… assets through the Virtualize REST API.

Source address affinity persistence will ensure that all calls from Parasoft CTP go to the same node. This session affinity is also known as "sticky sessions."

You'll want to enable source address affinity persistence (session stickiness) for the following ports:

- 9080 - Virtualize default HTTP connector (Required)
- 9443 - Virtualize default SSL connector (Required)
- 2424 - Data Repository (Required if a Data Repository exists on this node)
- 9617 – The built-in event monitor provider service (Active MQ)
- 9618 – The built-in server hit statistics provider service
- 9619 – The built-in JDBC provider service

With F5 Local Traffic Manager (LTM), source address affinity persistence is configured by enabling "Match Across Services" and "Match Across Virtual Servers." For example, if you wanted to implement source address affinity persistence with F5s LTM, you could use the default source_addr profile or create a custom profile. The following table shows the settings and values for the default source_addr profile.

Setting	Description	Default Value
Name	Defines a unique name for the profile. Required.	No default value
Persistence Type	Defines the type of persistence profile. Required.	Source Address Affinity
Match Across Services	Indicates that all persistent connections from a client IP address which go to the same virtual IP address should also go to the same node.	Enabled
Match Across Virtual Servers	Indicates that all persistent connections from the same client IP address should go to the same node.	Enabled
Match Across Pools	Indicates that the LTM system can use any pool that contains this persistence entry.	Disabled

Timeout	Indicates the number of seconds at which a persistence entry times out.	180
Mask	Defines the mask the LTM system should use before matching with an existing persistence entry.	0.0.0.0
Map Proxies	Enables or disables proxy mapping.	Enabled

Using Priority-based Member Activation

Priority-based member activation ensures that all calls from any origin will go to the main machine first (if possible). To configure this, set the first machine as the highest priority. For example, the following is a sample pool configuration file from F5 documentation for Local Traffic Manager:

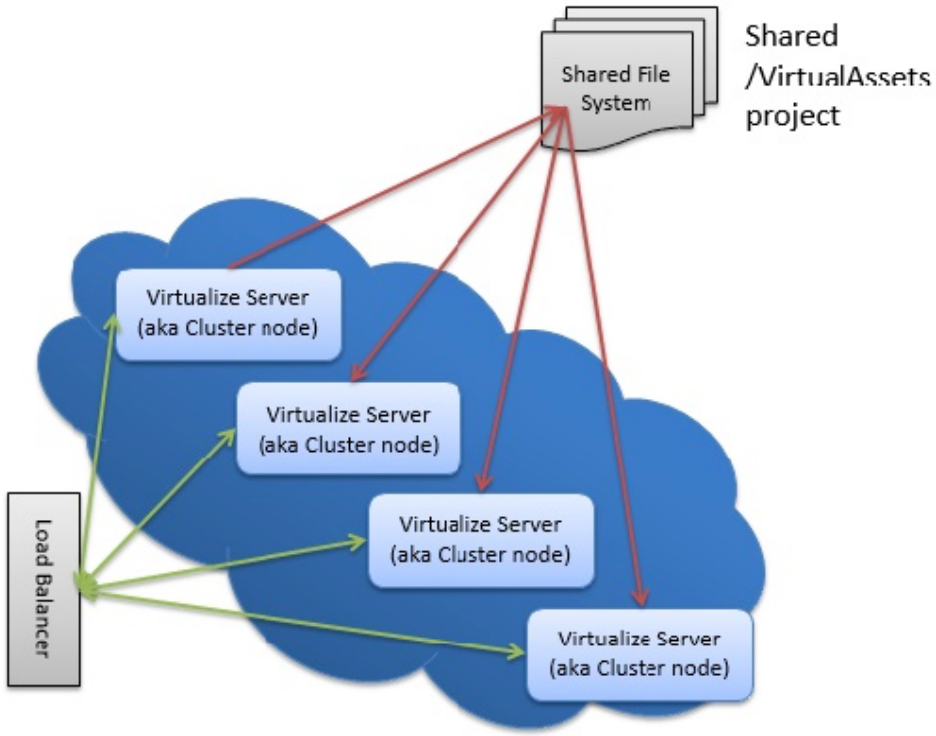
```
pool my_pool {
         lb_mode fastest
         min active members 2
member 10.12.10.7:80 priority 3
member 10.12.10.8:80 priority 3
member 10.12.10.9:80 priority 3
member 10.12.10.4:80 priority 2
member 10.12.10.5:80 priority 2
member 10.12.10.6:80 priority 2
member 10.12.10.1:80 priority 1
member 10.12.10.2:80 priority 1
member 10.12.10.3:80 priority 1
}
```

Parasoft Virtualize Configuration

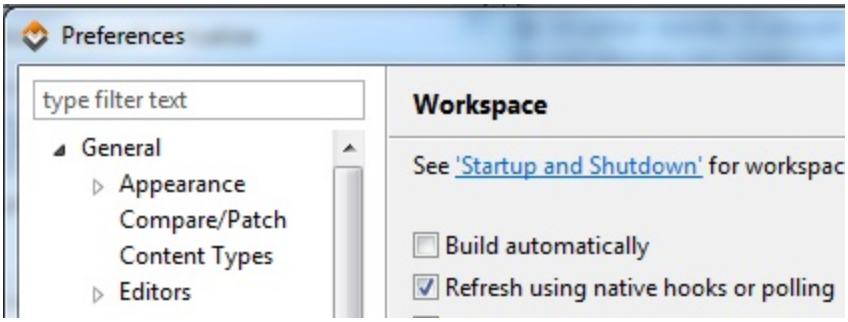
Before you start configuring Virtualize for load balancing, ensure that the load balancer is configured so that 'Changes' are propagated to only one node.

Next, do the following to ensure that the changes are then synchronized to the other nodes via the shared file system (SAN and/or NAS):

1. Share the VirtualAssets project and all of its associated content (.pmpdd, .pvadd, .pjcdd, VirtualAssets.xml, .git, etc.) across the cluster. A typical setup would be to mount an NFS folder as the VirtualAssets project in each node of the cluster.



2. Enable refreshing using native hooks or polling:
    - **From the GUI**: Navigate to **Window> Preference> General> Workspace**, check **Refreshusing native hooks or polling**, then restart the server.



- **From the command-line (for a headless launch)**: Add `refresh.enabled=true` in the `/.metadata/.plugins/org.eclipse.core.runtime/.settings/org.eclipse.core.resources.prefs` file, then restart the server. For example:

```
eclipse.preferences.version=1
version=1
refresh.enabled=true
```

3. Put the servers in "Cluster Mode":
    a. Add the following Java option to your startup command on all nodes in the cluster.

    ```
    -J-Dparasoft.auto.deploy.new=false
    ```

    b. Restart your servers.

Note that deployments in the Virtualize cluster will eventually be consistent. When a request deploy is sent, a single server processes the request and then returns a response. The file system will notify the other servers in the cluster to make the changes required for consistency. In some cases, this may take several seconds. As a result, there may be a brief time when some servers are not yet aware of a deployment that happened on another server.

Parasoft Data Repository Configuration

Parasoft Data Repositories, which are based on MongoDB, should be configured for clustering by identifying a primary Data Repository (the one that has the data you want replicated) and creating a replicate set. To do this, follow this procedure (taken and modified from the MongoDB documentation):

1. Stop all of the Data Repositories you want to become part of the replica set.
2. Create the key file each member of the replica set will use to authenticate servers to each other.
   To generate pseudo-random data to use for a keyfile, issue the following openssl command:

   ```
   openssl rand -base64 741 > mongodb-keyfile
   chmod 600 mongodb-keyfile
   ```

   You may generate a key file using any method you choose. Always ensure that the password stored in the key file is long and contains a high amount of entropy. Using openssl in this manner helps generate such a key.
3. Copy the mongodb-keyfile key file to each member of the replica set. Set the permissions of these files to 600 so that only the owner of the file can read or write this file to prevent other users on the system from accessing the shared secret.
4. Beginning with your primary data repository, start each member of the replica set with the –keyFile and –replSet command-line options (to specify the key file and the name of the replica set, respectively). To add these options, edit the Data Repository's server.sh or server.bat script file (at the line that calls mongodb). For example:

   ```
   mongod --keyFile /mysecretdirectory/mongodb-keyfile --replSet "rs0"
   ```

5. Connect to the primary Data Repository and authenticate as the admin user, created by the M_USER variable in the server.sh or server.bat script:

   ```
   "rs.add("mongodb1.example.net:2424")
   ```

6. (Optional) If you want to increase the write safety of the replica set, modify the primary data repository's write concerns. With the default setting, the client returns when one member acknowledges the write; you can change this so that a majority must acknowledge the write. See the MongoDB documentation for details.
7. On the primary data repository, initiate the replica set using rs.initiate():

   ```
   rs.initiate()
   ```

   This initiates a set that consists of the current member and that uses the default replica set configuration.
8. On the primary data repository, verify the initial replica set configuration by using `rs.conf()` to display the replica set configuration object:

```
rs.conf()
```

The replica set configuration object should resemble the following:

```
{
    "_id" : "rs0",
    "version" : 1,
    "members" : [
       {
           "_id" : 1,
           "host" : "mongodb0.example.net:27017"
       }
     ]
}
```

9. Add the remaining replica set members to the replica set with the rs.add() method. You must be connected to the primary data repository to add members to a replica set.
   rs.add() can, in some cases, trigger an election. If the Data Repository you are connected to becomes a secondary, you need to connect the mongo shell to the new primary to continue adding new replica set members. Use rs.status() to identify the primary in the replica set.
   The following example adds two members:

   ```
   rs.add("mongodb1.example.net")
   rs.add("mongodb2.example.net")
   ```

   When complete, you have a fully functional replica set. The new replica set will elect a primary.
10. Check the status of the replica set using the rs.status() operation:

    ```
    rs.status()
    ```

Tips and Tricks

Provisioning

- CTP and Virtualize Desktop can modify live assets. When nodes talk through the load balancers, CTP and Virtualize will treat the cluster as a single server. Note that the name of the server sent to CTP (as configured in the Virtualize server or localsettings.properties) must be the same on all nodes.
- Recording is not supported in a clustered environment. Recording should be performed on your staging infrastructure prior to asset promotion.

Asset Promotion

- Use source control to store ‘production grade’ assets and versioning information. Check out assets into the Shared File System for initial node configuration.

Consumption

- All AUT traffic is sent to the Load Balancer and filtered appropriately using the settings mentioned in Load Balancer Hardware and Software Configuration.
- To ensure that changes are sent to only one server, the load balancer for the Virtualize servers should be configured to send configuration/ ‘change’ messages to a single node in the cluster using the techniques defined in Load Balancer Hardware and Software Configuration. This should be done for the port (defaults are 9080/9443) and/or path (/axis2 SOAP or /soavirt/api/ REST) that handles ‘changes’ sent from the Virtualize desktop or CTP.
- To ensure consistent behavior from stateful assets, the Virtualize Server load balancer must direct all traffic for a single session to a single node. Note that:
  - Traffic for a stateful asset should only be sent to one node. In other words, it should not be run in "first available" to ensure that multiple assets do not change the state multiple times.
  - If stateful assets are intended to hold state across sessions, then the nodes will need to store the state in a synchronized data source, (i. e. a data repository).