Lab 4 Using dynamic clusters to automatically allocate resources to an overloaded application

Dynamic clusters are intended to provide autonomic environment management, allowing your environments to automatically heal themselves and allocate resources where those resources are most needed. In this exercise, you will learn how to create a new dynamic cluster and then use the built-in CPU overload capabilities of your dynamic cluster to automatically add extra processing capacity to your cluster in response to load. In this exercise you will:

- Configure your cluster with a low CPU utilization threshold for overload protection
- Install an application in your cluster
- Create a chart to use to track CPU utilization in the application server instances in your dynamic cluster
- Use JMeter to drive load against your installed application and watch the intelligent controllers in your environment start the second application server instance to even out the CPU load

1.1 Create a new dynamic cluster

We will use the pre-existing dynamic cluster we created in lab 1 named DynamicClister01

We want to set the heap size for the cluster to be significantly low to cause a trigger of cluster expansion.

- Go to **Servers → Clusters → Dynamic clusters**
- Select **DynamicCluster01**
- Select **Server Template** from the right hand side, under **Additional Properties**
- Select **Java and Process Management → Process definition**, under **Server Infrastructure**
- Select **Java Virtual Machine** from the right hand side, under **Additional Properties**
- Set the **Maximum heap size** to **256 MB**
- Select **OK**
- Click the **Save** link and wait for the node synchronization to complete, then click the **OK** button.

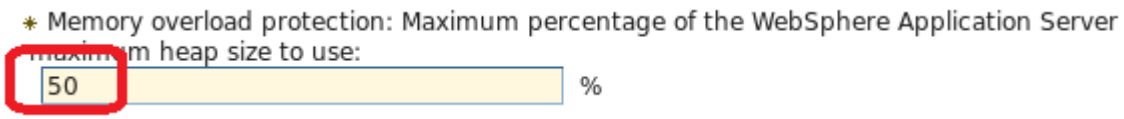
1.1.1 Update the dynamic cluster configuration and install an application

For this exercise, you want to make it easy to trigger the dynamic cluster to allocate additional resources to your environment, so you are going to set the CPU utilization threshold artificially low as well as the Memory overload protection.

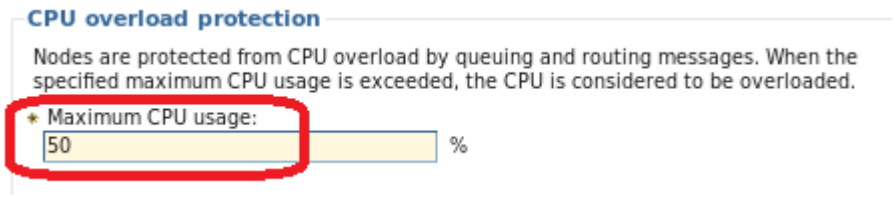
__1. Reduce the CPU utilization threshold for overload protection.

__a. Go to **Operational policies > Autonomic Managers > Autonomic Request Flow Manager**.

__b. Set the Memory overload protection to **50**.



__c. Set the Maximum CPU usage to 50%



__d. Scroll down to the bottom of the display and select **OK**. Choose to **Save** the changes to the master configuration repository. When node synchronization completes, select **OK**.

__2. Install a sample application that you will use to drive load against your cluster.

__a. Go to **Applications > New Application**, then select **New Enterprise Application**.

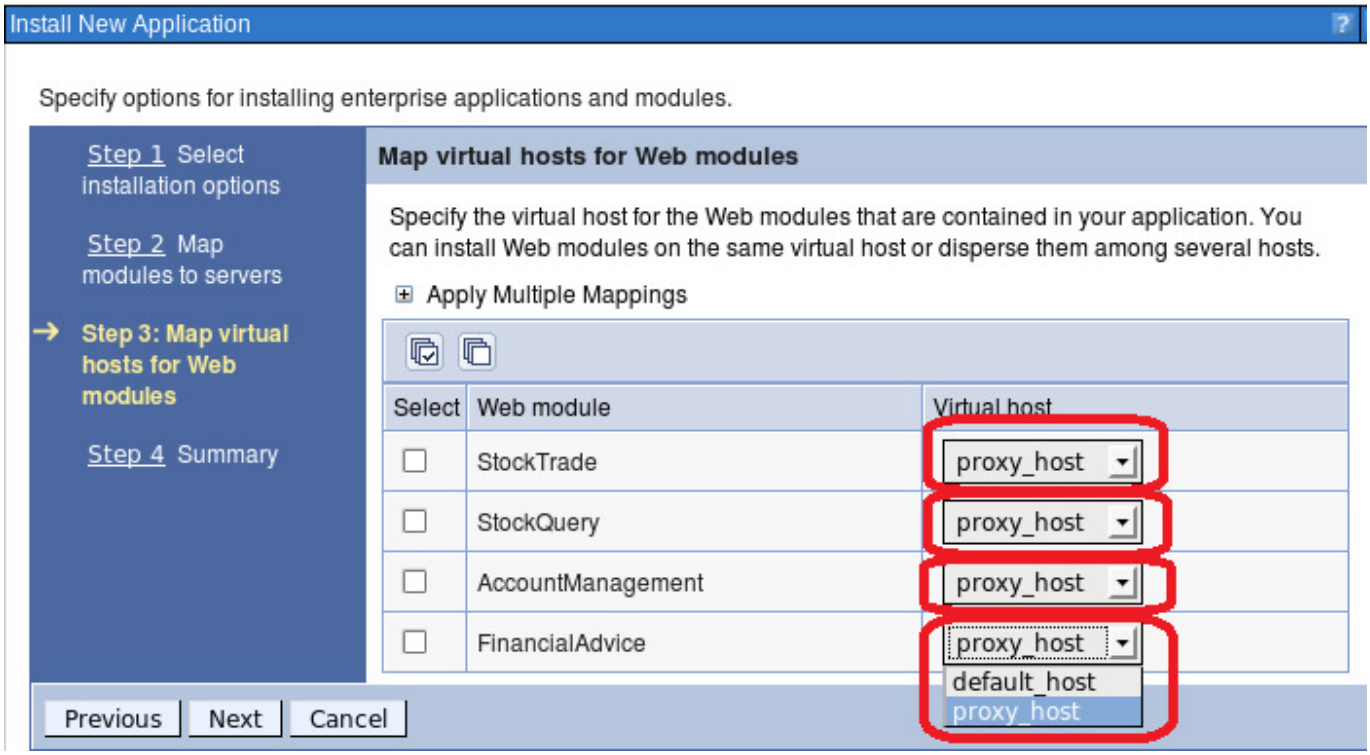
__b. Verify that **Local file system** is select, then choose **Browse…**. In the file dialog, navigate to **/home/wasadmin/workshop/labs/PerfMgmt/IMStock.ear**. Select **Open**, then select **Next**.

__c. Choose to use the **Fast Path** installation option, then select **Next**.

__d. In **Step 1: Select installation options**, keep all of the default options, then click **Next**.

__e. In **Step 2: Map modules to servers**, verify that all four modules are mapped to the **DynamicCluster01** dynamic cluster. Click **Next**.

__f. In **Step 3: Map virtual hosts for Web module,** Change the virtual host to proxy_host. Not we set the proxy host port to 1024 since that is the port the ODR is listening on. If you haven't done this, then follow the steps <TODO LINK TO STEPS>



__g. Review the **Summary** data, then click **Finish**. After you see the message that your application installed successfully, click **Save**.

__h. After node synchronization completes, click **OK**.

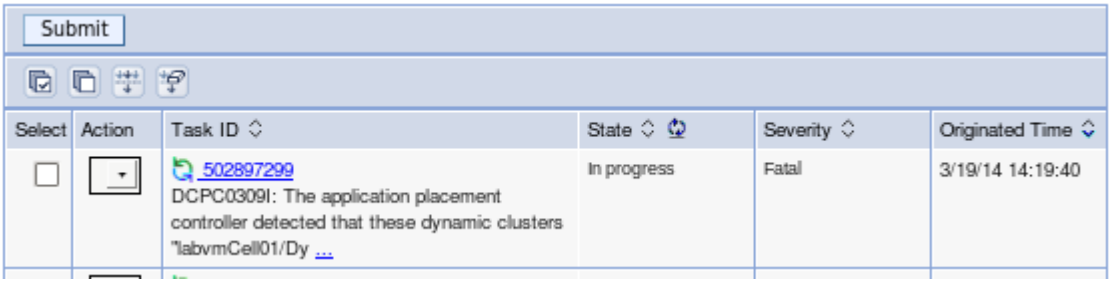
__3. Switch your dynamic cluster to **Automatic** mode and watch one of your servers start.

__a. Navigate to **Servers > Clusters > Dynamic clusters**.

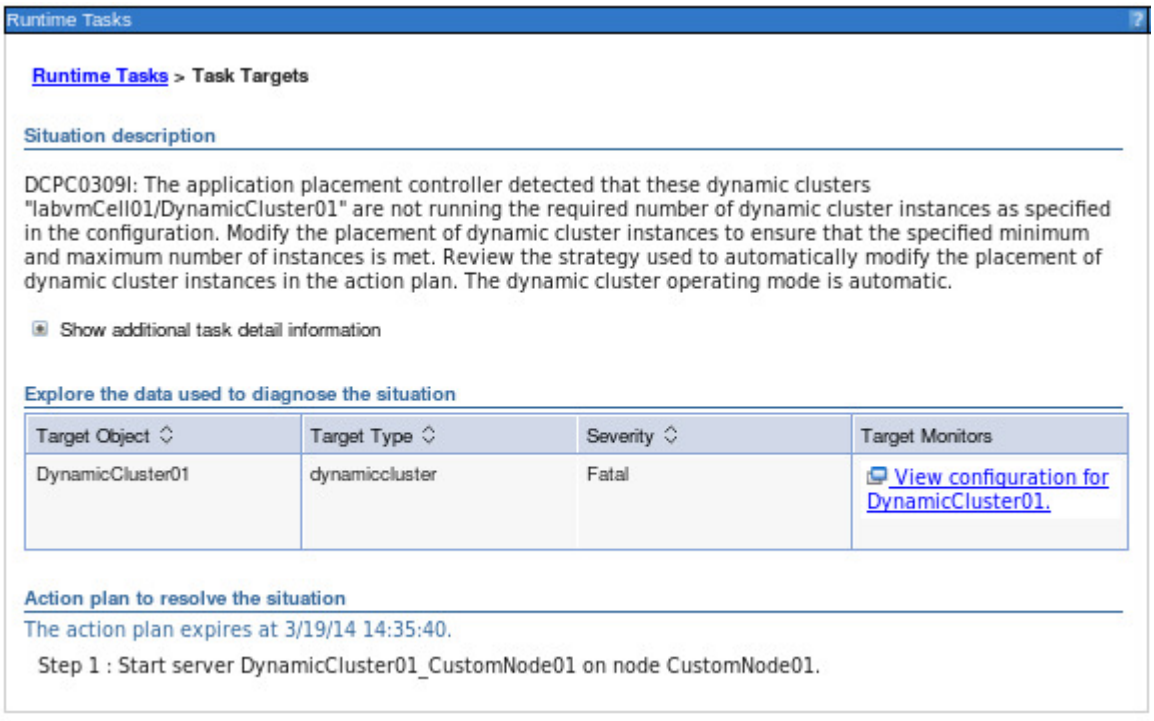
__b. Select the box next to **DynamicCluster01**. Set the mode menu to **Automatic**, then click the **Set Mode** button. Now your cluster is in Automatic mode.

__c. You will use the runtime tasks display to see that one of your servers is starting. Go to **System administration > Task Management > Runtime Tasks**.

__d. You should see a runtime task with a message ID of **DCPC0309I** in the list of tasks, its state will either be **In progress** or **Succeeded**. If you do not see a task at the **TOP** of the table with this message ID, periodically refresh the **Runtime Tasks** page until the task appears. This may take up to 5 minutes.

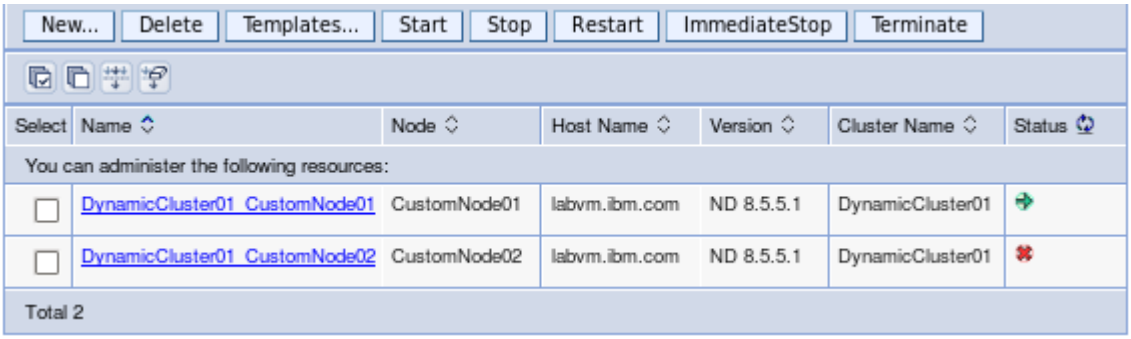


__e. Select the **Task ID** for your new runtime task (in the sample screen capture above, this means clicking the hyperlinked task ID text 502897299 in the first row of the table). This task is telling you that your dynamic cluster is not running the specified minimum number of instances (which is 1). The action plan for this task is to start one of the servers.



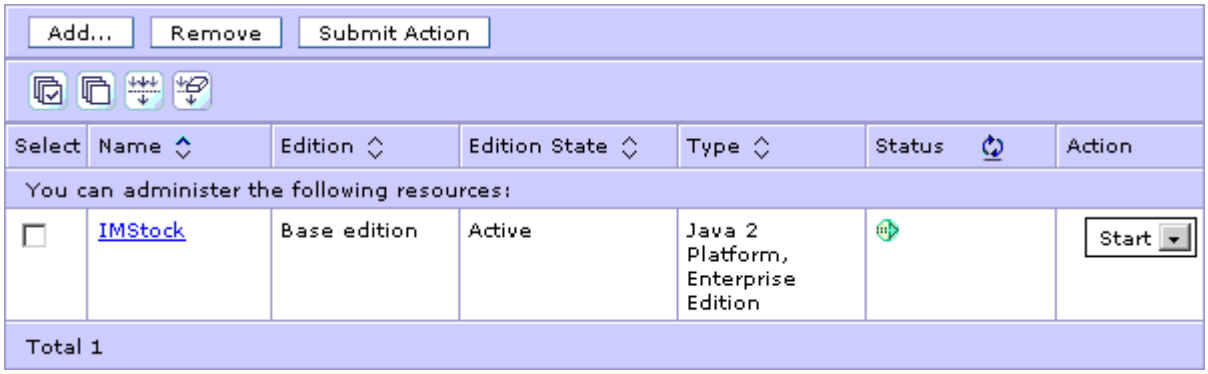
__f. Use the **Runtime Tasks** link at the top of the display to navigate back to your task list. If your task is still listed as **In progress**, periodically refresh the display until the status is shown as **Succeeded**.

__g. Navigate to **Servers > Server types > WebSphere application servers**, and verify that one of your server instances is listed in Started state.



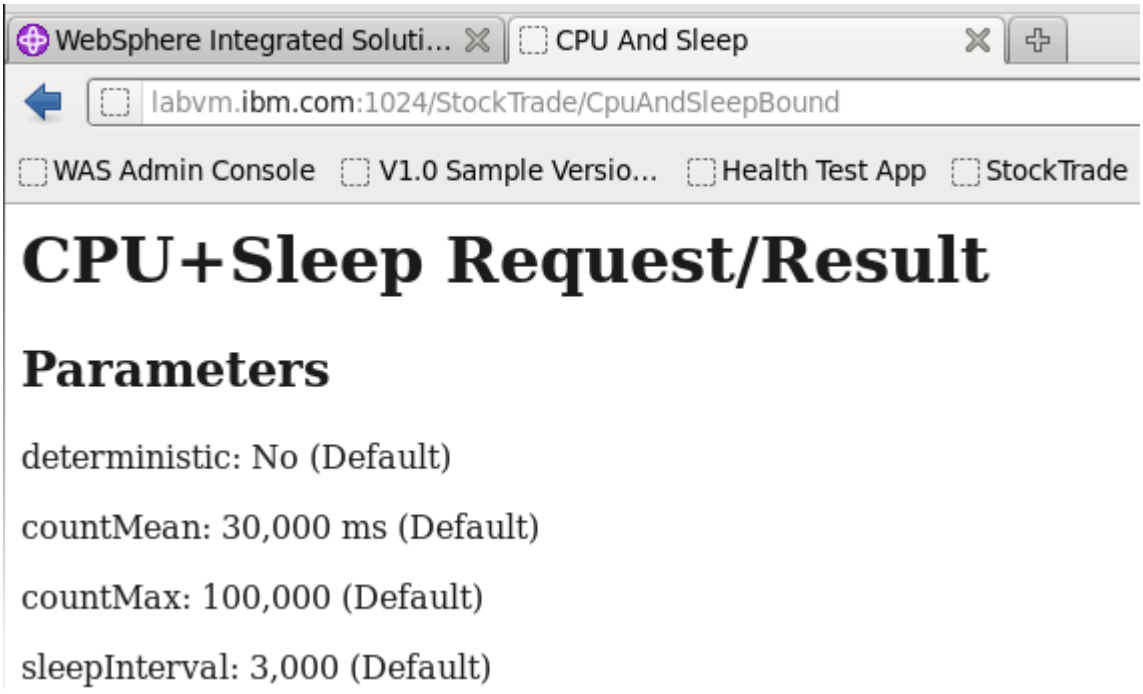
__4. Verify that your application is running.

__a. Navigate to **Applications > All applications**. You should see **IMStock** listed with a status of either **Started** or **Partial start**. If it is not started, then select the application, make sure the **Action** menu is set to **Start**, then click **Submit Action**.



Open a new tab in your browser. Use this URL to access the running application http://labvm.ibm.com:1024/StockTrade/CpuAndSleepBound

If the application page does not load for you correctly, check with your instructor before proceeding with the exercise.



1.2 Use Intelligent Management to add processing capacity to your cluster under load

Before you drive any load against your cluster, create a chart to track CPU utilization.

__1. Create a chart.

__a. Go to **Runtime Operations > Reports**. Scroll down in the panel and select the **Add Data…** button.

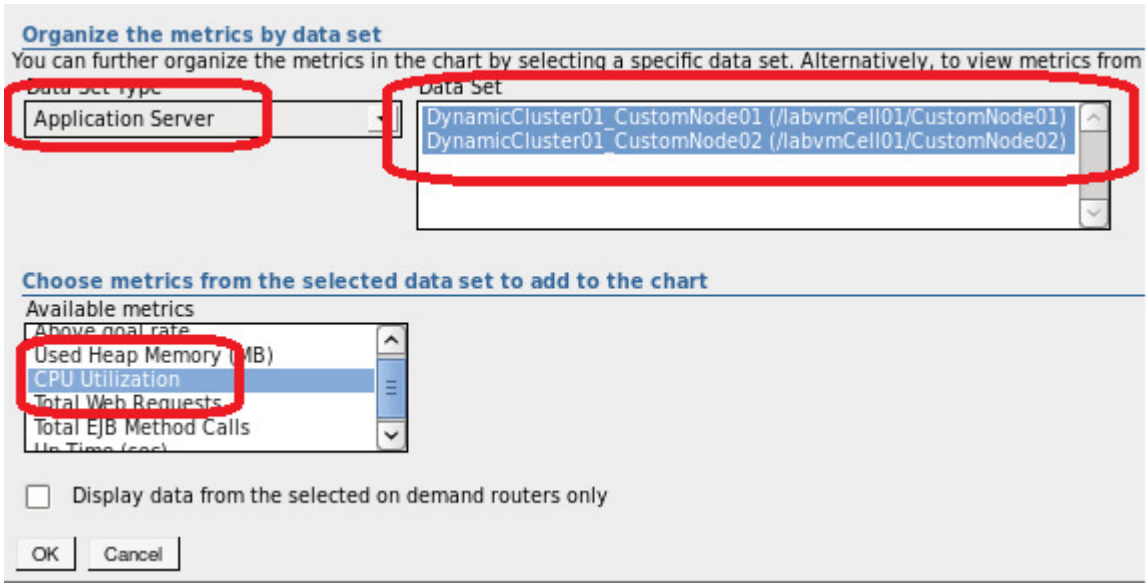
__b. You want to create a chart to track CPU utilization in the servers in your dynamic cluster. To do this, set these values in the **Organize the metrics by data set** panel:

__i. From the **Data Set Type** menu, choose **Application Server**.

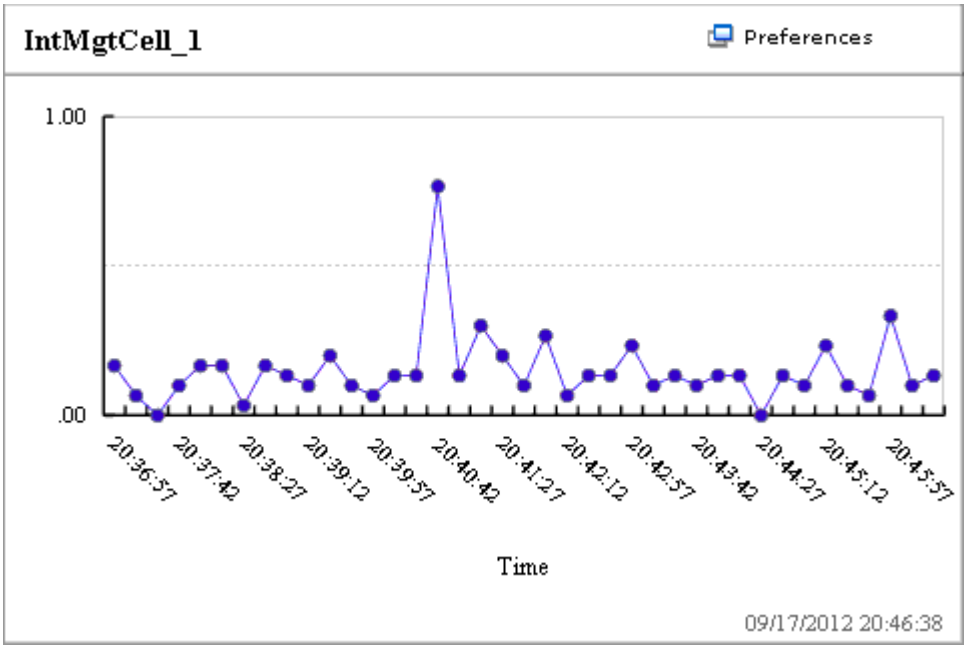
__ii. After the **Data Set** menu refreshes, choose **both** of the application servers that are listed (use **Ctrl+Click** to make multiple selections).

__iii. From the **Available metrics** menu, choose **CPU utilization**.

__iv. Click **OK**.



__c. You should see a graph with a single line, that looks roughly like this, with low-level CPU utilization for your server (only one of your application servers is currently running):



__d. Scroll down to the bottom of the page and locate the **Save current group of chart tabs configuration as chart group** field, and type in **Stock Charts**. Click **Save**. This will allow you to easily come back to your chart later if you get logged out of the environment.

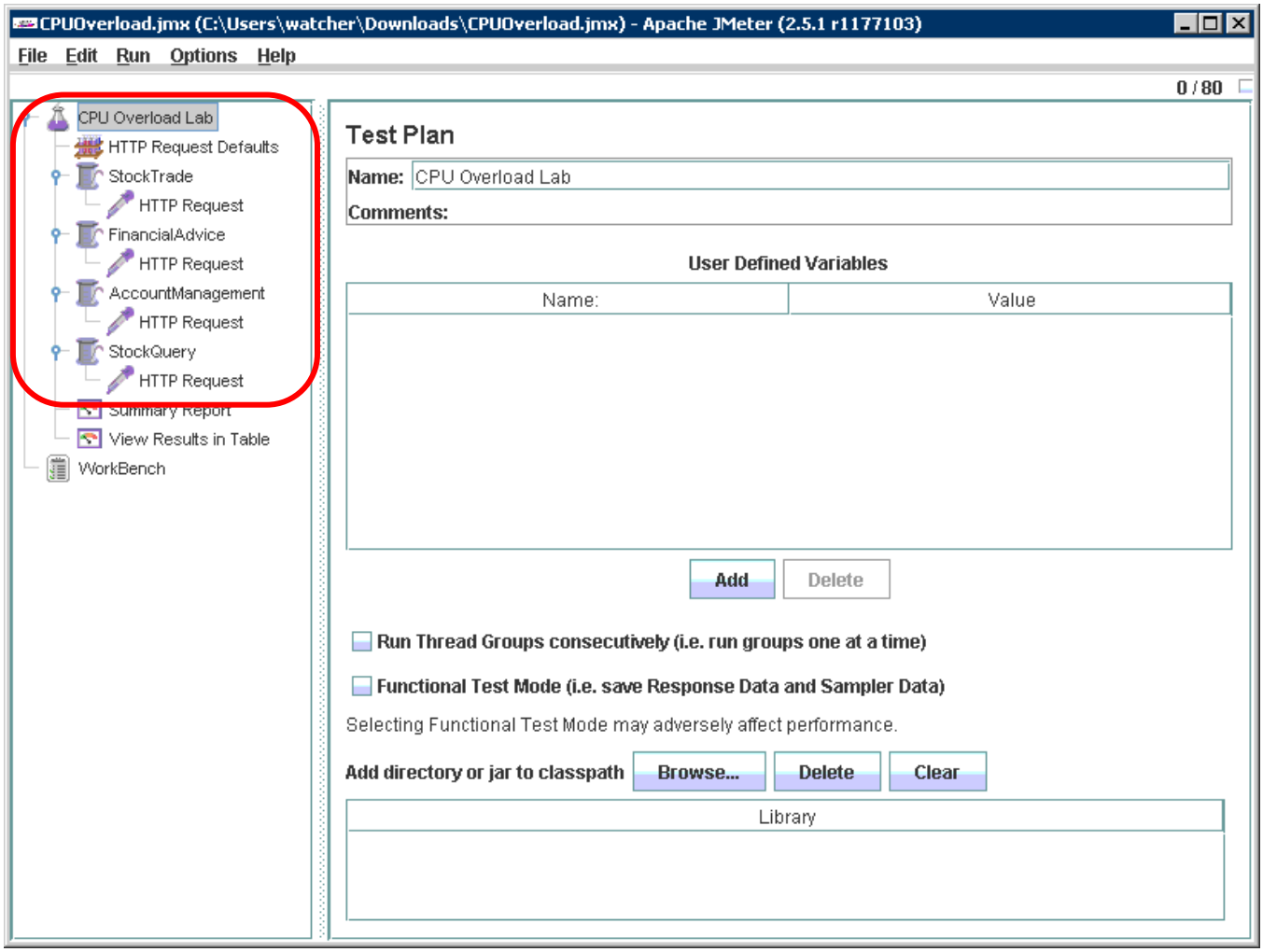
__e. When prompted, **Save** your changes to the master configuration. Click **OK** when node synchronization is complete.

__2. Launch JMeter to drive load against your application.

__a. Double-click on the **JMeter** shortcut on your desktop to open the local JMeter client. If your desktop does not have a JMeter shortcut, check with your instructor to determine where JMeter is installed and how you should launch it. It might take a moment for the application to load.

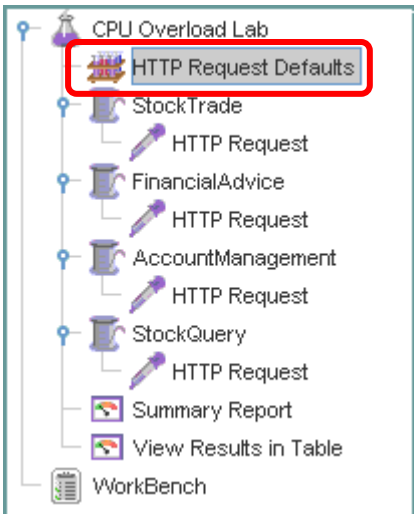
__b. A pre-built JMX file with load information has been provided for you to use with this exercise. Open it in JMeter by selecting the **File > Open** menu. Navigate to **/home/wasadmin/workshop/labs/PerfMgmt/CPUOverload.jmx**, then click **Open**.

__c. In your JMeter workspace, you should see the **CPU Overload Lab** test plan, with several sections on the left panel that describe the HTTP requests that will be sent to your application to drive CPU utilization. In this plan, you will be driving 10 threads against each of four web modules (**StockTrade**, **FinancialAdvice**, **AccountManagement**, and **StockQuery**), for a total of 40 threads. The application is designed to take HTTP parameters that describe how each thread will continue to eat up CPU resources as long as the threads are active.

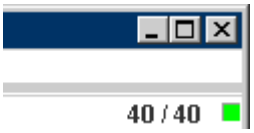


__d. Set your test plan to point to your on demand router.

__i. From the left navigation panel, select the **HTTP Request Defaults** area.



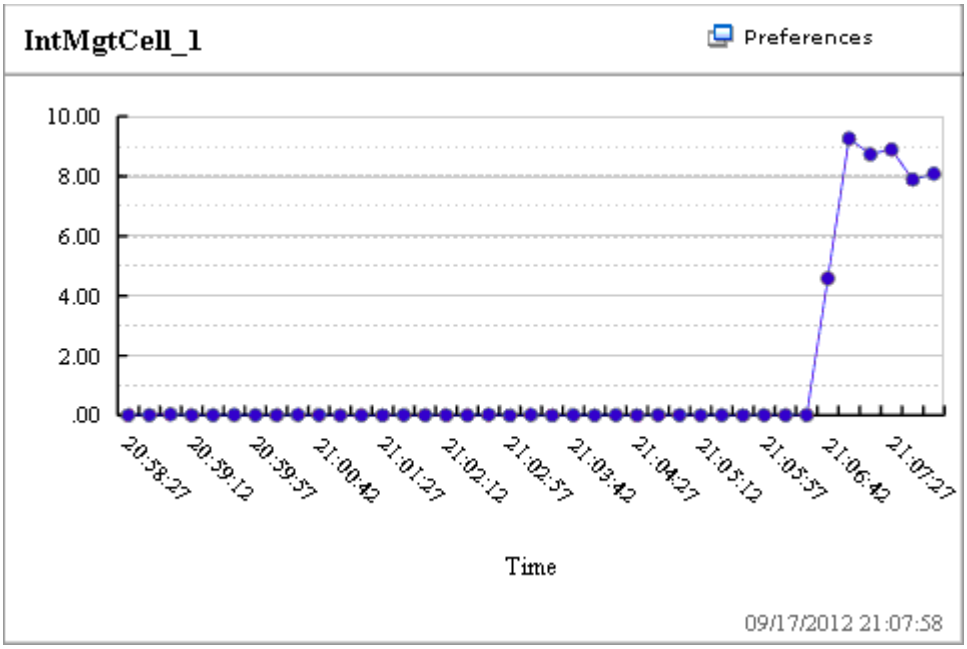
__e. Select **Run > Start** from the JMeter menu to kick off the test plan. You should see the upper right indicator spin up to **40** / **40** threads active, with a green status.



__3. Monitor your environment for CPU overload.

__a. Back in the WebSphere administrative console, go to your chart view (**Runtime Operations > Reports**, then open your **Stock Charts** report).

__b. You should see a sudden spike in CPU utilization, corresponding to the start of your JMeter test plan:

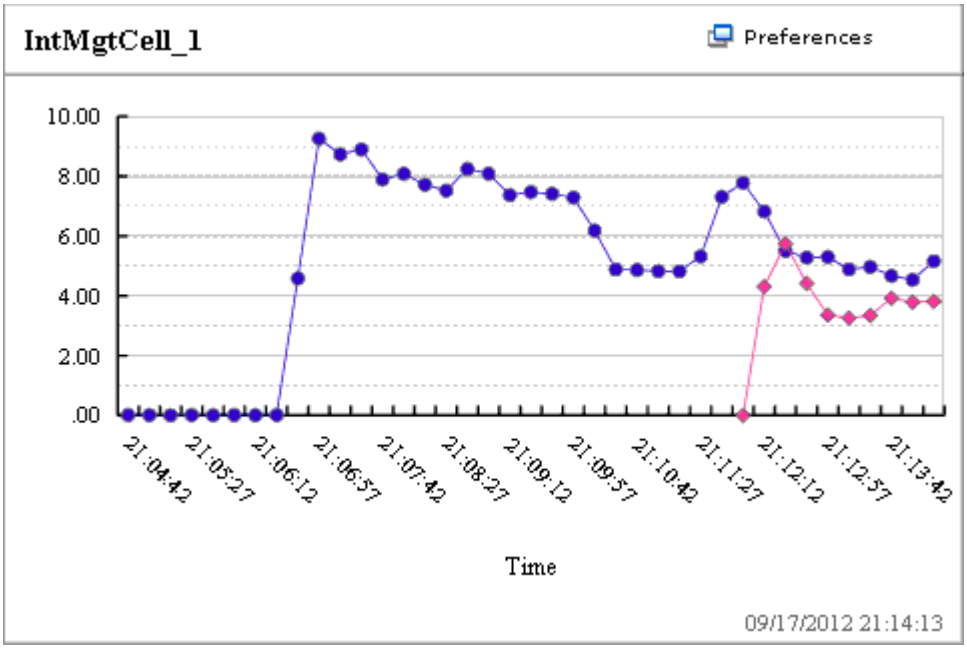


__c. Over time, the CPU utilization of your server may drop off for short periods, but it should be averaging over 60% utilization. You might need to refresh the Scale in the data table at the bottom of the graph to see up-to-date scale information so that you can accurately read your graph.

__d. If you see your utilization drop down below 60% for more than a couple of minutes, try adjusting the number of threads in your JMeter test plan upward to drive additional load.

__i. To change your JMeter load, go into JMeter and select **Run > Stop** to stop the current load. In the test plan navigation, select one of the modules that is driving load (for example, **StockTrade**). Increase the number of threads in small increments (perhaps initially moving from 10 to 15 threads), then try **Run > Start** to begin driving load again. You can continue increasing the number of threads until you see your load hovering around / above 60%. Work with your instructor if you have questions about how to proceed.

__e. After a short time (approximately 5-6 minutes), you will see a new line show up in your graph (in this case, in pink). This line is for the second application server in your cluster, which was automatically started to help offload work from the first overloaded server.



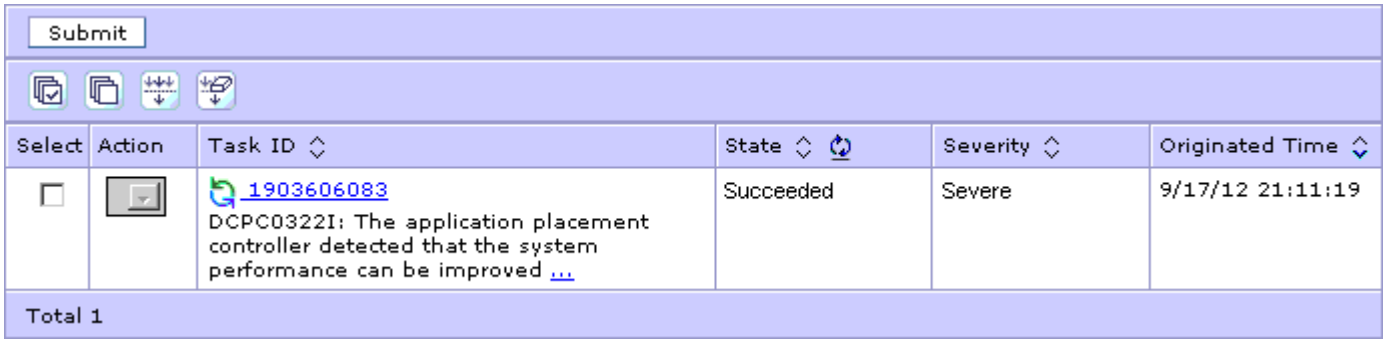
__f. Now that both servers are running (one on each custom node virtual machine), your graph shows them processing workload at a healthier utilization rate (less than 60% CPU utilization on each server).

Note: your VM may run very slow at this point. Ideally you would have your additional nodes on separate hardware, thus distributing load more appropriately. Starting a second server on the same machine won't necessarily increase performance, since both servers are sharing the same CPU usage, however for the purpose of this lab we can see the cluster expanding.

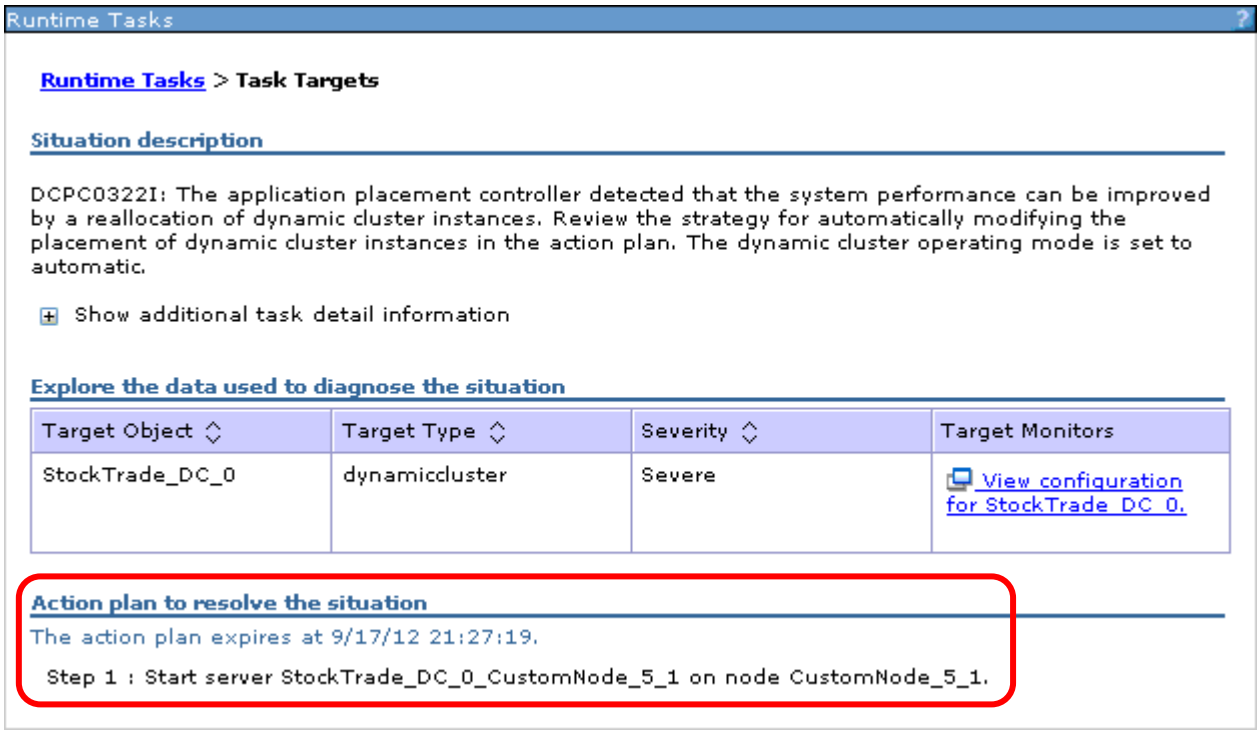
__4. Verify the configuration updates to your environment.

__a. Check the runtime task that was generated to start the second server in your environment, by going to **System administration > Task Management > Runtime Tasks** in the administrative console.

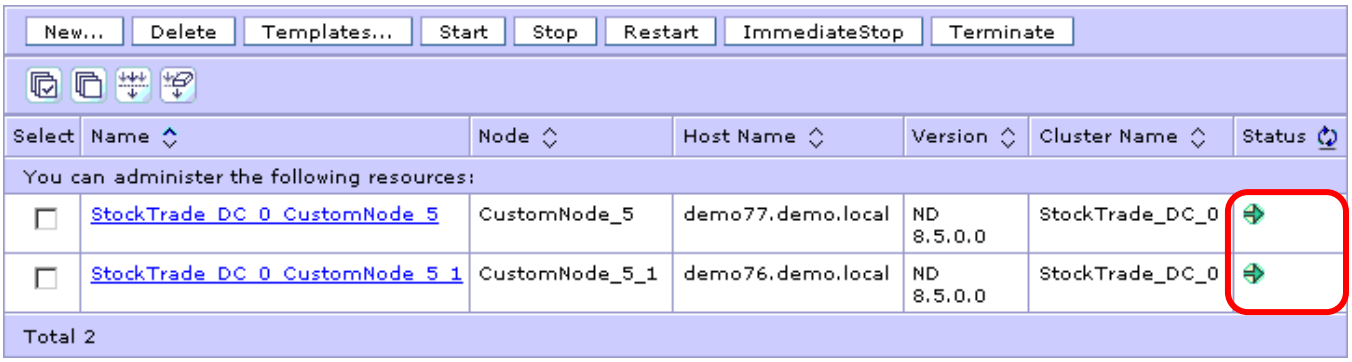
__b. You should see a runtime task with message ID **DCPC0322I** at the top of the table, with a status of **Succeeded**.



__c. Select the **Task ID** number to open the runtime task details. You should see that your cluster detected that it could improve the performance of your environment by starting another server instance.



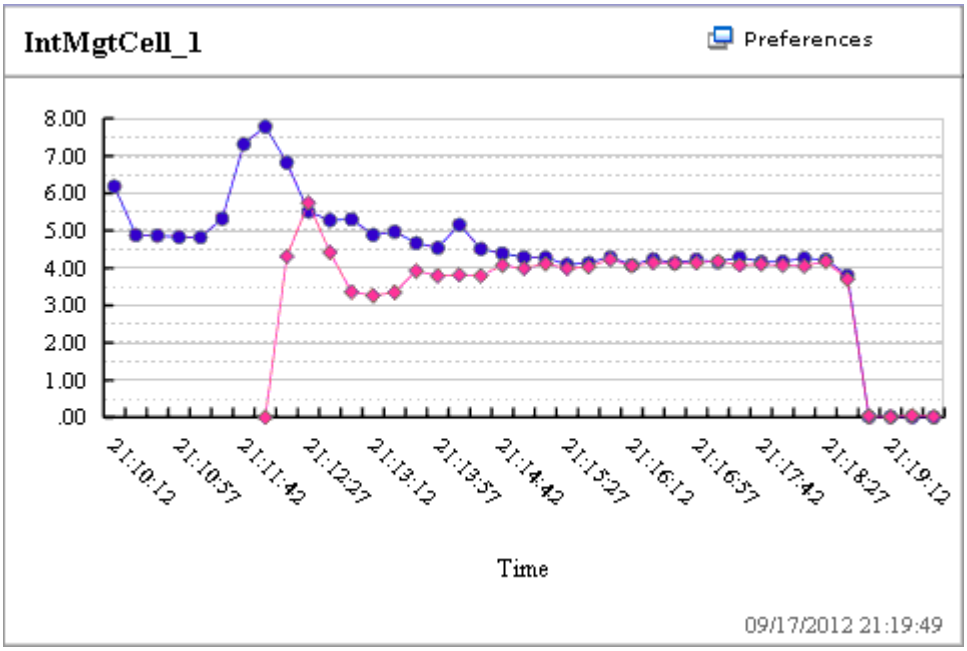
__d. Finally, verify that both of your servers are running on the **Servers > Server Types > WebSphere application servers**. You should see both servers listed in **Started** status:



__5. Stop sending requests to your environment.

__a. Go back into **JMeter** and select **Run > Stop**. All of your user threads should stop, so that the upper right corner of the interface shows **0** / **40** threads active.

__b. Back in the chart view, you should see that all of the workload on your servers has stopped.



__6. In this exercise, you saw how your Intelligent Management controllers were able to add extra processing capacity to your dynamic cluster, by starting another server in response to heavy CPU utilization.

1.3 Clean up

You have completed all of the lab exercises, so now you can delete your environment.

1.4 Summary

In this exercise, you learned how to create a dynamic cluster and configure it to run in Automatic mode. The nice thing about Automatic mode is that it enables your cluster to automatically take corrective action, without any operator intervention. In this case, you configured a low CPU utilization threshold for overload protection, then installed a simple sample application and used JMeter to drive load against it. When the load exceeded your CPU utilization threshold, the controllers in your Intelligent Management

topology automatically started another application server instance to offload some of the work, improving the overall performance of your environment.