SAP BusinessObjects BI Pattern Books

BI Platform 4.x Performance Testing Pattern Book

ABSTRACT

This pattern book provides comprehensive and step-by-step instructions for creating and executing performance tests on BI platform. It also demonstrates how to monitor, collect detailed performance information, and tune the environment based on the data

Disclaimer

- This pattern book is for informational purpose only and may not be copied / reproduced without the permission of SAP
- The information provided in this book are based on the SAP BI Pattern Books project for a specific set of patterns / use cases applied within SAP lab environment. Hence, make sure to review and apply the steps / workflows that are applicable to your use cases / patterns, based on your SAP BusinessObjects BI landscape
- Contents of this, or any related document and SAP's strategy and possible future developments, products and or platforms directions and functionality that are discussed in this book all subject to change and may be changed by SAP at any time for any reason without notice. Therefore, read the latest official product guides, release notes to understand the differences and act accordingly

For further comments and questions, email to SAPEnableBI@sap.com

Contents

Abbreviations 4
Planning and Scoping 4
Architecture 7
Deploying the Pattern 9
Creating test plans 11
Capturing the web request using Fiddler 11
Recording the workflow with Apache Jmeter 14
Moving the request to a test plan 17
Customizing test plan for dynamic data 22
Login Transactions 22
Browse Folder Transaction 29
View Report Transaction 30
Refresh and Prompt Transaction 33
Final Configuration Components 35
Initial Run 38
Determining Benchmarks 41
Performance testing machine resource consumption 41
Ramp-up Period 42
Repeating Test 42
Timers 42
Data Sources 43
Pattern source data details 43
Semantic Layers 44
Data Model 44
Web Intelligence Performance Testing 44
Web Intelligence Documents 45
Workflow Examples 45
Web Intelligence Test Plan and Results 46
Resources for Test 9 ..... **Error! Bookmark not defined.**
Test 1- Out of the box baseline and example 54
Test 2- 50 Active Concurrent Users 69
Resources for Test 2 79
Test 3- 150 Active Concurrent Users 87
Test 4- 150 Active Concurrent Users 88
Resources for Test 4 - 150 ACU 91

Test 5- 150 Active Concurrent Users 99
Resources for Test 5 - 150 ACU 108
Test 6- 100 Active Concurrent Users 117
Resources for Test 6 - 100 ACU 121
Test 7 to10 - Investigating Bottleneck Root Cause 129
Resources for Test 7 134
Resources for Test 8 141
Resources for Test 9 148
Test 11- 150 Active Concurrent Users - After ESX and Load Balancer Changes 155
Resources for Test 11 161
Test 12- Screenshots and Data 168
Resources for Test 12 169
Test 13- 300 Active Concurrent Users 177
Resources for Test 13 185
Performance Tuning Resources and Tips 194

So far, SAP has released pattern books for BI 4.x release, covering the BusinessObjects deployment scenario on Linux and Windows, and the upgrade scenario of existing environment. As installation and deployment are documented well and understood, we have decided that the latest version of the SAP BusinessObjects BI Platform 4.x pattern books must cover the performance of the infrastructure.

Performance coupled with consistency, positively impacts the adoption of BI platform and these are the critical factors that is crucial for the success of your BI deployment.

The aim of Performance Testing Pattern Book is to provide comprehensive and step-by-step instructions, for creation and execution of performance tests on BI platform. Performance test also demonstrates how to monitor, collect detailed performance information, and tune the environment based on the data.

By validating your sizing, you can perform the load test, performance test, and set your own benchmark.

- Planning and Scoping
- Architecture
- Deploying the Pattern
- Creating test plans
- Determining Benchmarks
- Data Sources
- Web Intelligence Performance Testing
- Web Intelligence Test Plan and Results
- Performance Tuning Resources and Tips

Abbreviations

The keywords and their acronyms used in this pattern book are as follows:

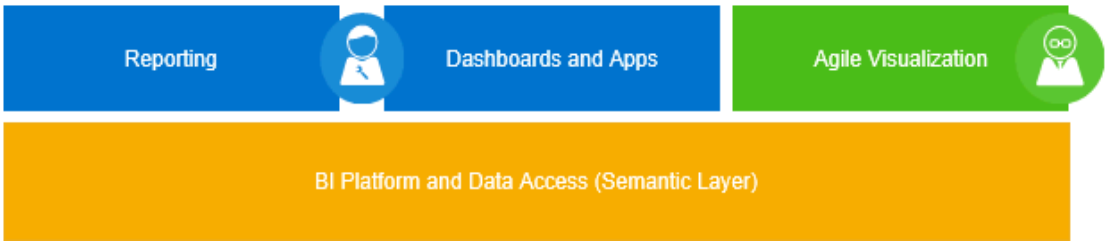
Acronym	Keyword
ACU	Active Concurrent User
WEBI	Web Intelligence
APS	Adaptive Processing Server
DSL	Dynamic Semantic Layer
ART	Average Response Time

Planning and Scoping

The first step towards planning is to create a conducive atmosphere, which represents real-world scenario. This scenario should enable us to use the BI content for testing. We use a combination of three different client segments representing Business Intelligence, as identified by SAP. The components identified under each client segments are as follows:

Client	Components
Web Intelligence	Reporting & Analysis
Lumira	Agile Visualization
Design studio	Dashboards & Applications

Figure: Client Segments and Components



Furthermore, Design Studio and Lumira are fairly new additions to the BI platform family, and there is minimal detail on performance testing of these clients.

The versions included in testing are as follows:

- BI Platform/Web Intelligence 4.1 SP6
- Lumira Server for BI Platform 1.28 **
- Design Studio 1.5 **

Warning: **The first version (current version) of pattern book covers the performance and load testing details around Web Intelligence. SAP Design Studio and SAP Lumira will be covered in the next version of this book.

After selecting clients, we identified SAP HANA as the data source that we use for testing. The decision of selecting SAP HANA data source is to highlight the performance of the BI platform, when the database response time is effectively zero. Selecting HANA database allows us to clearly illustrate the overhead associated with BI platform servers, and causes the iteration to be a simple process. Also, the content of this pattern book is relevant to other databases (SAP and Non-SAP database).

Finally, the need is to determine the number of users and workflows targeted for the test. The sizing deployment guide and Lumira are leveraged to identify the typical workflows for each client:

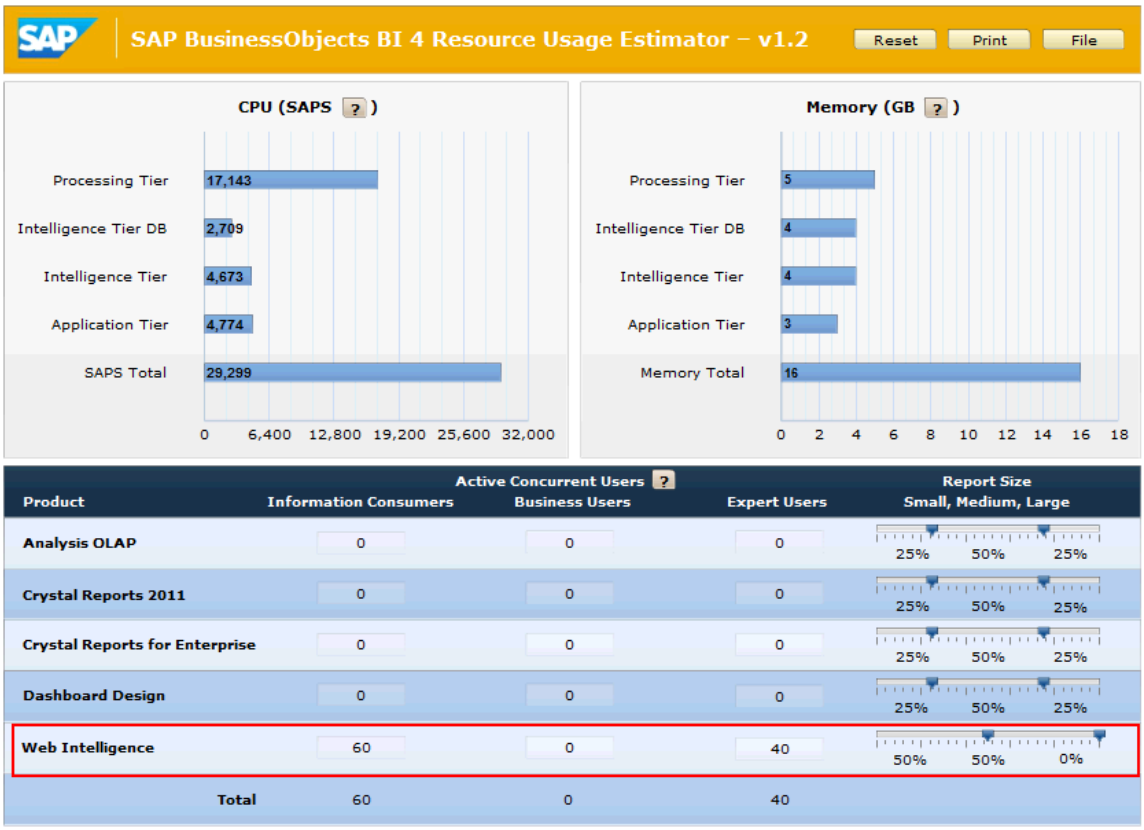
http://scn.sap.com/docs/DOC-33126

http://scn.sap.com/docs/DOC-53374

Apache JMeter is selected as the performance testing tool based on the fact that it is an open source tool and freely available for use. The documentation of Apache JMeter is available on SCN for reference. The test started with a target of 100 active concurrent users (ACU), ramping up to a certain level over a period of time. The testing team pushed platform beyond the level of 100 ACU as a part of stress testing.

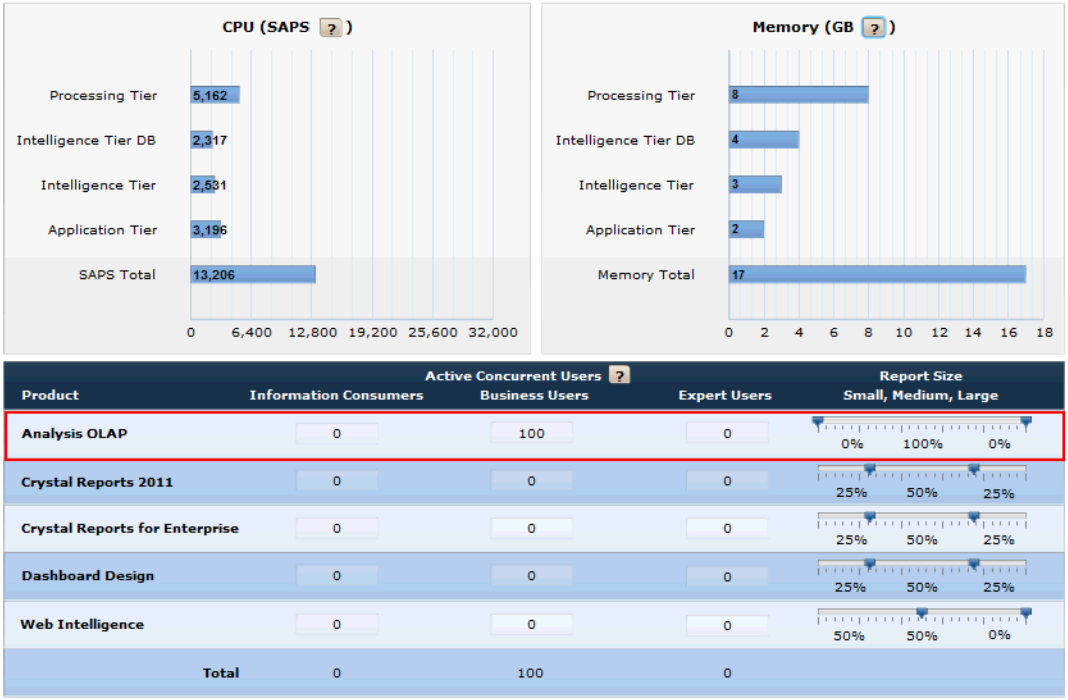
The 100 ACU is split as 60 information consumers and 40 expert users (2/3rd of information consumers) for Web Intelligence, and this ratio is maintained throughout the test. By using the Sizing Estimator, the following details are derived:

Figure: Resource Usage Estimator for WEBI



The sizing estimator provided the estimation of ~8 CPU cores, and 5 GB of memory for processing tier. The estimate is increased to 8 GB of memory for each processing tier node. This increase is to accommodate splitting of APS and increased memory for DSL Bridge process, responsible for the connection to HANA via UNX universe. Furthermore, it is understood that ~3 GB of memory is required for Tomcat application server.

Figure: Resource Usage Estimator for Design Studio



This indicates that we need ~4 CPU cores and 8 GB of memory for Design Studio tests.

At this time, sizing estimator does not include Lumira server for BI Platform, but the Lumira Sizing Guide provides suggestions for resources.

Figure: Suggestions for Lumira sizing guide

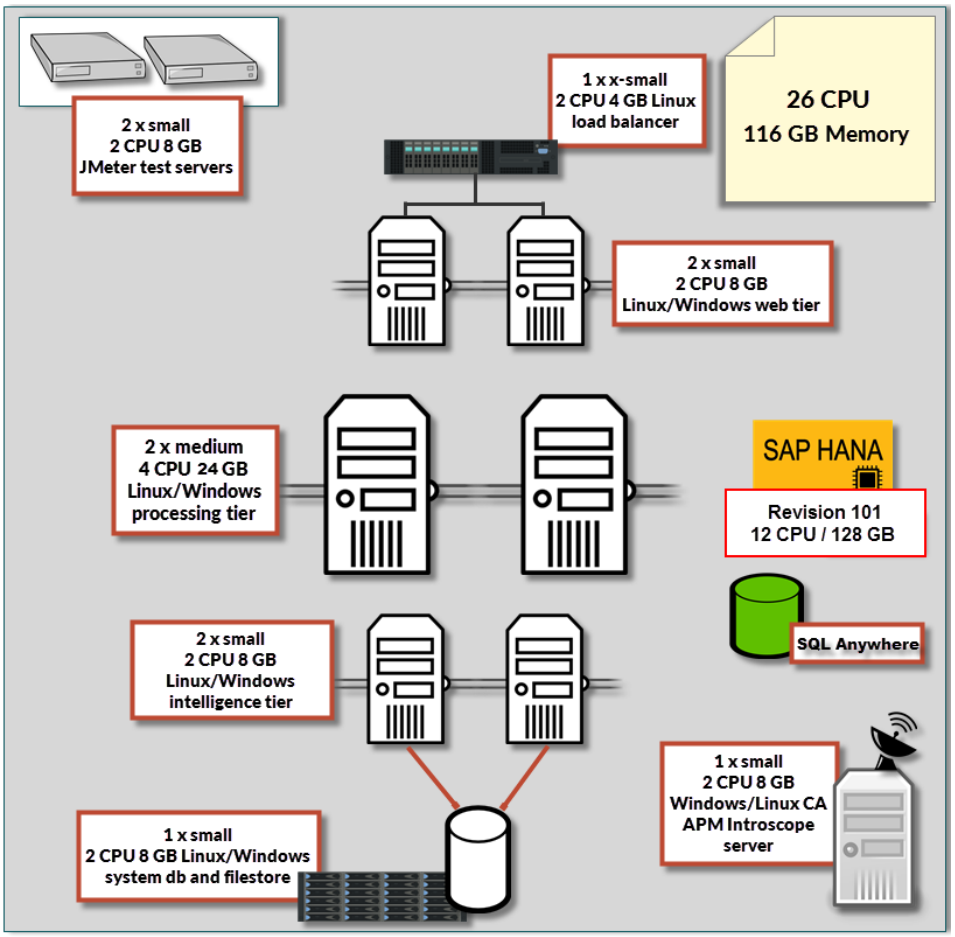
The sizing estimate below should only be used as a starting point as it is based on the workflows outlined in Table 1. We strongly recommend volume testing to validate your sizing estimate based on the expected usage in your deployment.

Maximum Cell Size (no. of rows * no. of columns)	Recommended Maximum Active Concurrent Users		
60,000,000	10	12	15
700,000	15	25	35
	Recommended Configuration Scenario		
Minimum memory required (RAM in GB)	**32**	**48**	**64**
Minimum CPU Cores	**8**	**24**	**24**

Architecture

The architecture used for this pattern book is illustrated in the following image.

Figure: Architecture



The machines that map to this configuration are as follows:

Machine	Operating System Platform	Deployment
LNXPLPB01	SUSE Enterprise Linux 11.3	Apache 2.2 (Load Balancer)
WINPLPB01	Windows 2012 R2	JMeter 2.13
WINPLPB02	Windows 2012 R2	JMeter 2.13
WINPLPB03	Windows 2012 R2	CA APM Introscope 9.5.1
WINPLPB04	Windows 2012 R2	SAP SQL Anywhere 12 (Reporting)
WINPLPB05	Windows 2012 R2	SAP SQL Anywhere 12 (System Database & FRS)
WINPLPB06	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Web Application Tier)

Machine	Operating System Platform	Deployment
WINPLPB07	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Web Application Tier)
WINPLPB08	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Intelligence Tier)
WINPLPB09	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Intelligence Tier)
WINPLPB010	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Processing Tier)
WINPLPB011	Windows 2012 R2	SAP BusinessObjects BI Platform 4.1 SP6 (Processing Tier)
hanatestfra	SUSE Enterprise Linux 11.3	HANA Revision 101 (AB1)

Based on the above-mentioned sizing estimates, it is decided to allocate 24 GB of memory for processing tier to provide minimum recommended memory for Lumira Server. Although 32 GB is the minimum requirement, our data sets maxed out at around 200,000 cells so we determined this to be sufficient. It also provided us with ample room to scale up more than 100 active concurrent users with Web Intelligence and Design Studio.

Warning:

- Two Apache JMeter servers are provided in order for a single load generator to avoid the bottleneck. JMeter supports parent/slave relationships for distributed testing.
- BI Platform provides full redundancy and fault tolerance, which is fronted by an Apache web server to handle the load balancing aspect.
- Although an SQL Anywhere reporting database is provided, in this iteration of the pattern book we are testing using HANA as the backend database only.
- The pattern is deployed on VMware ESX servers using virtual machine technology.

Deploying the Pattern

The focus of this pattern is performance testing, so deployment of the pattern is performed according to existing resources available from SAP. The relevant instructions are found at the following locations:

1. To install and configure an Apache Load balancer on LNXPLPB01 follow the instructions described under the topic “Apache” in the BI 4.x Linux Pattern Book.

2. To install Apache JMeter on WINPLPB01 and WINPLPB02 follow the instructions described in SCN document:http://wiki.scn.sap.com/wiki/display/BOBJ/Introduction+to+Apache+JMeter
3. To install and configure CA APM Introscope 9 on WINPLPB03 follow the instructions described in SCN document:http://scn.sap.com/community/bi-platform/blog/2014/01/31/a-complete-guide-to-setup-ca-apm-introscope-9-for-monitoring-sap-bi-platform-41-on-windows
4. To install SAP SQL Anywhere 12 on WINPLPB05 follow the instructions described in SCN document: Using SQL Anywhere for a CMS/Auditing database in BI 4.1
5. To install SAP BusinessObjects BI Platform SP6 web tier installation on WINPLPB06 and WINPLPB07 follow the instructions described in the "BIP on Windows with Mobile and Explorer Pattern Book".
6. To install SAP BusinessObjects BI Platform SP6 full Installation on WINPLPB08, WINPLPB09, WINPLPB10, and WINPLPB11 follow the instructions described in the "BIP on Windows with Mobile and Explorer Pattern Book".
7. To install SAP BusinessObjects Design Studio 1.5 on WINPLPB06, WINPLPB07, WINPLPB10, and WINPLPB11 follow the instructions described in SCN document: http://service.sap.com/~sapidb/012002523100014716212015E/ds_15SP02_admin_bip_en.pdf
8. To install SAP Lumira Server for BI Platform on WINPLPB06, WINPLPB07, WINPLPB10, and WINPLPB11 follow the instructions described in SCN document: http://help.sap.com/businessobject/product_guides/lumS4BIP1/en/lumS4BIP_128_admin_en.pdf
9. SAP HANA Interactive Education (SHINE) was used as the foundation for this pattern. You can deploy the analytic and calculation views that were used in the performance tests by importing the delivery unit under the topic "Deploying the Pattern" in the BI Platform 4.x Performance Testing Pattern Book.
10. The documents used in the performance tests were exported, along with universes and connections, to an LCMBIAR file. You can contact the authors to download the LCMBIAR file.

Creating test plans

A test plan describes the series of steps Apache JMeter executes when run. A complete test plan consists of one or more thread groups (users), controllers, samplers, or configuration elements. You can follow the below mentioned wiki resource to build your first test plan against BI 4.1, consisting of a simple logon/logoff operation.

http://wiki.scn.sap.com/wiki/display/BOBJ/Creating+your+First+Test+Plan

To become familiar with the application of JMeter against Web Intelligence reports, you can utilize this SCN document to build a simple refresh test plan using a sample report:

http://scn.sap.com/docs/DOC-45507

The process below is used to record numerous test plans against SAP BI.

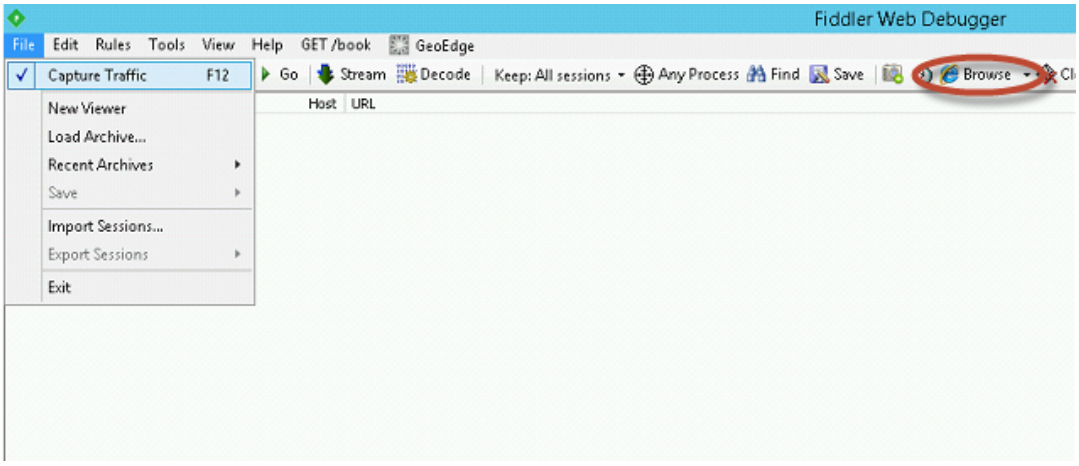
- Capturing the web request using Fiddler
- Recording the workflow with Apache Jmeter
- Moving the request to a test plan
- Customizing test plan for dynamic data
- Final Configuration Components
- Initial Run

Capturing the web request using Fiddler

To record the workflow interactively, use Fiddler, a web debugging proxy.

The procedure is as follows:

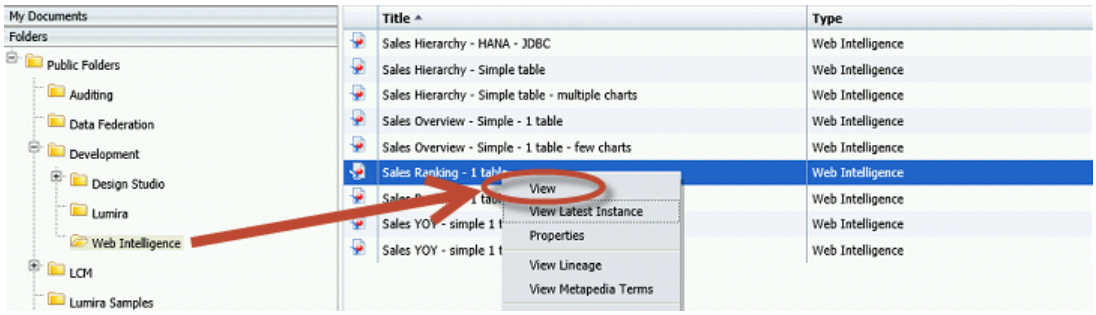
1. Launch Fiddler, ensure Capture Traffic is enabled.
2. Click the Browse button.



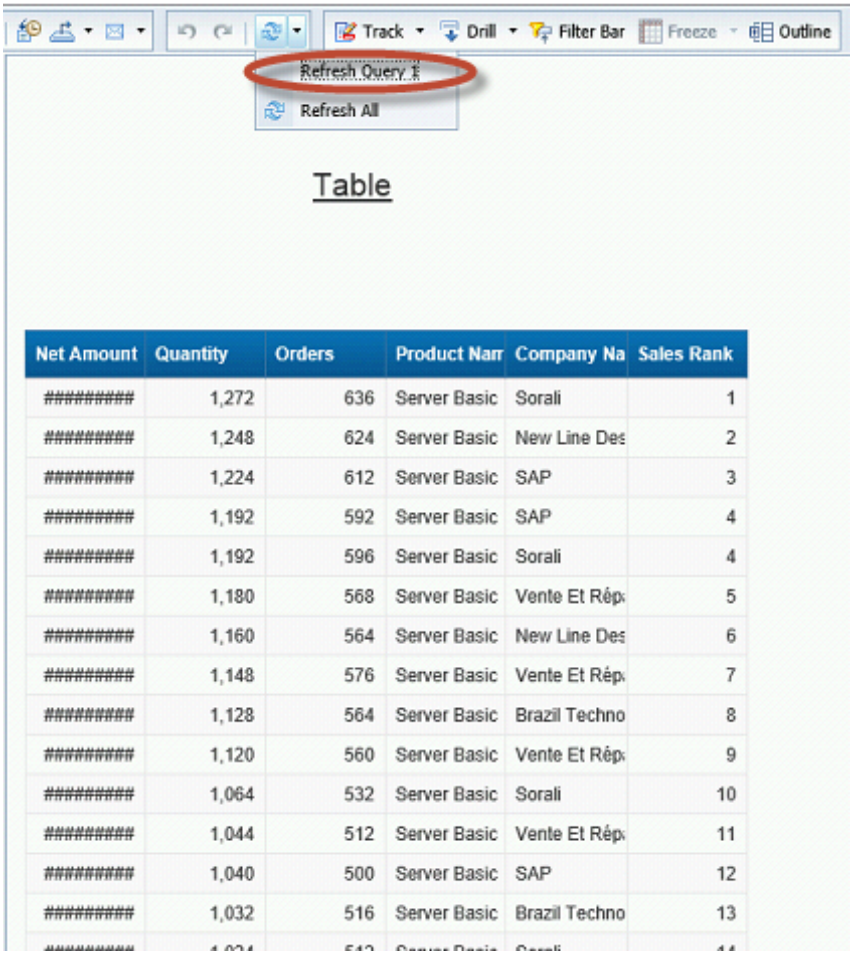
3. Perform the workflow.

   For example: Refer the workflow that is required to record Workflow 1 – View Webi Document "Sales Ranking - 1 Table" on demand below.

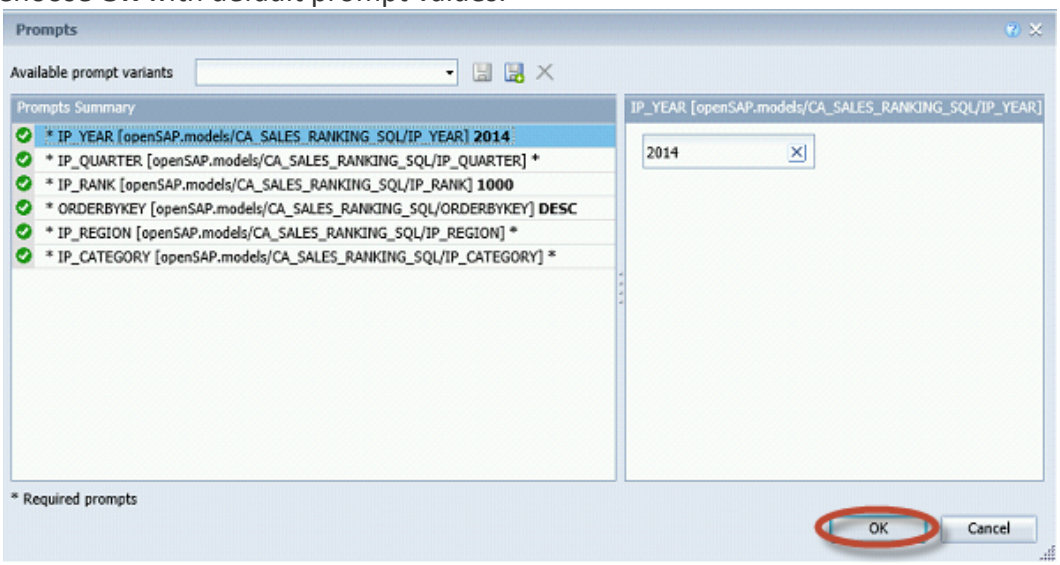
In BI launch navigate **Public Folders > Development > Web Intelligence** > **Sales Ranking – 1 table** and Select **View**.



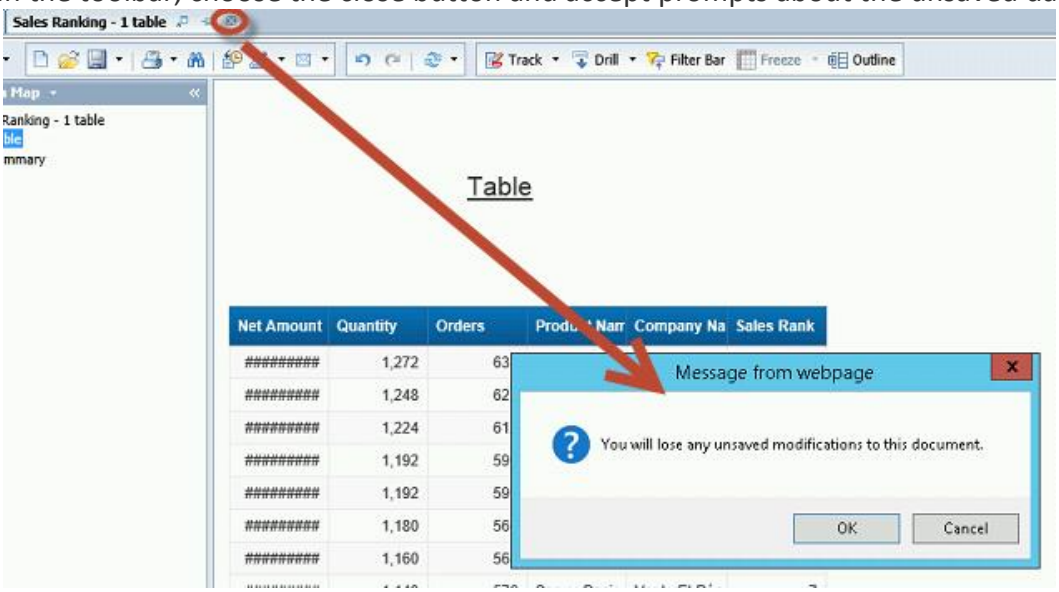
4. Choose the **Refresh** icon and choose **Refresh Query 1**.



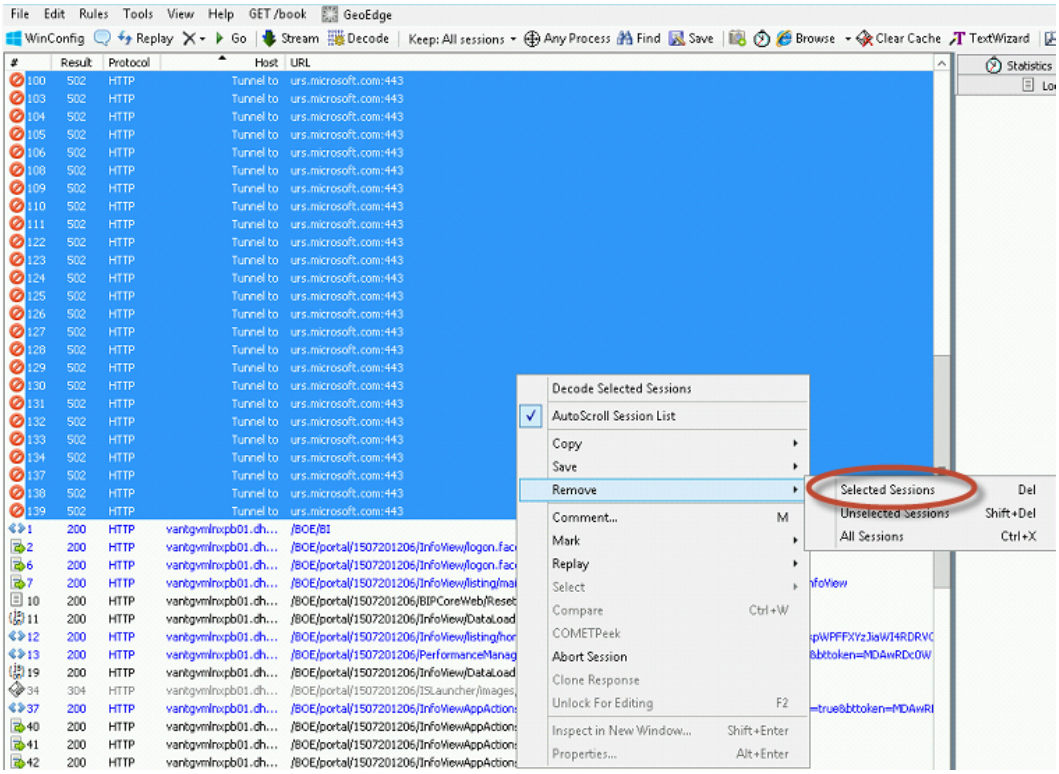
5. Choose **OK** with default prompt values.



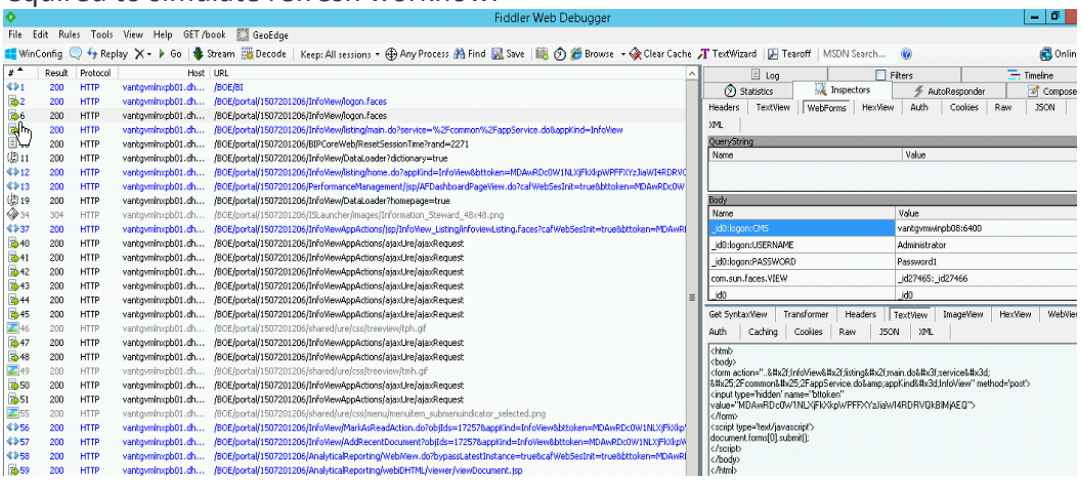
6. In the toolbar, choose the close button and accept prompts about the unsaved data.



7. Log off to complete the process.
8. Return to Fiddler and note the HTTP requests added to the output. Sort the output by host and delete any requests not associated with LNXPLPB01.



9. On the left column, sort the request based on number to produce a list of requests required to simulate refresh workflow.



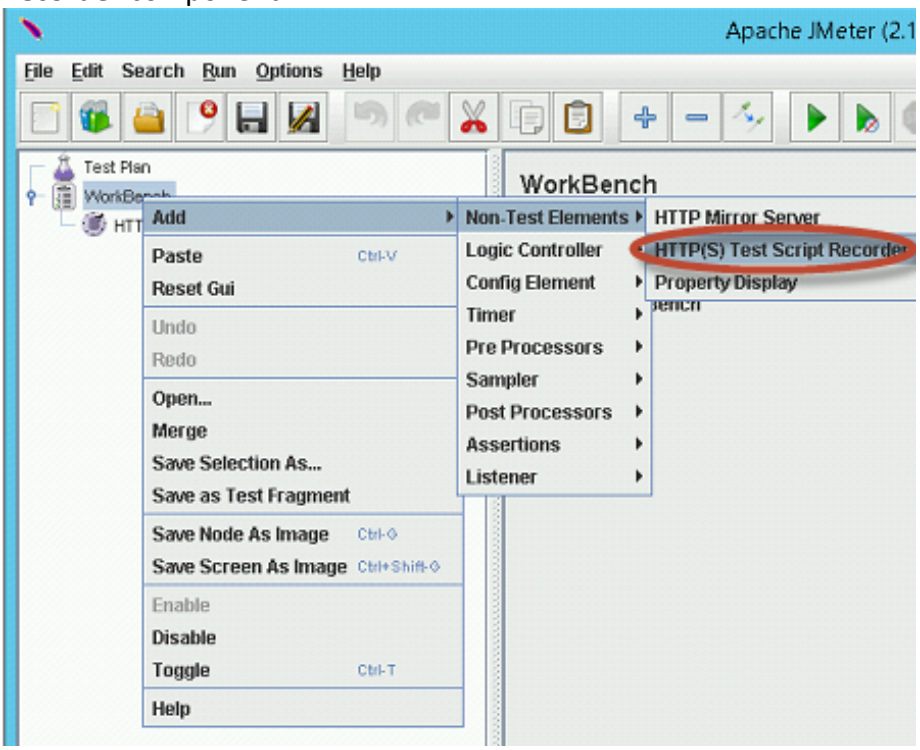
10. Save the sessions in the .SAZ archive type in order to use for future reference.

Recording the workflow with Apache Jmeter

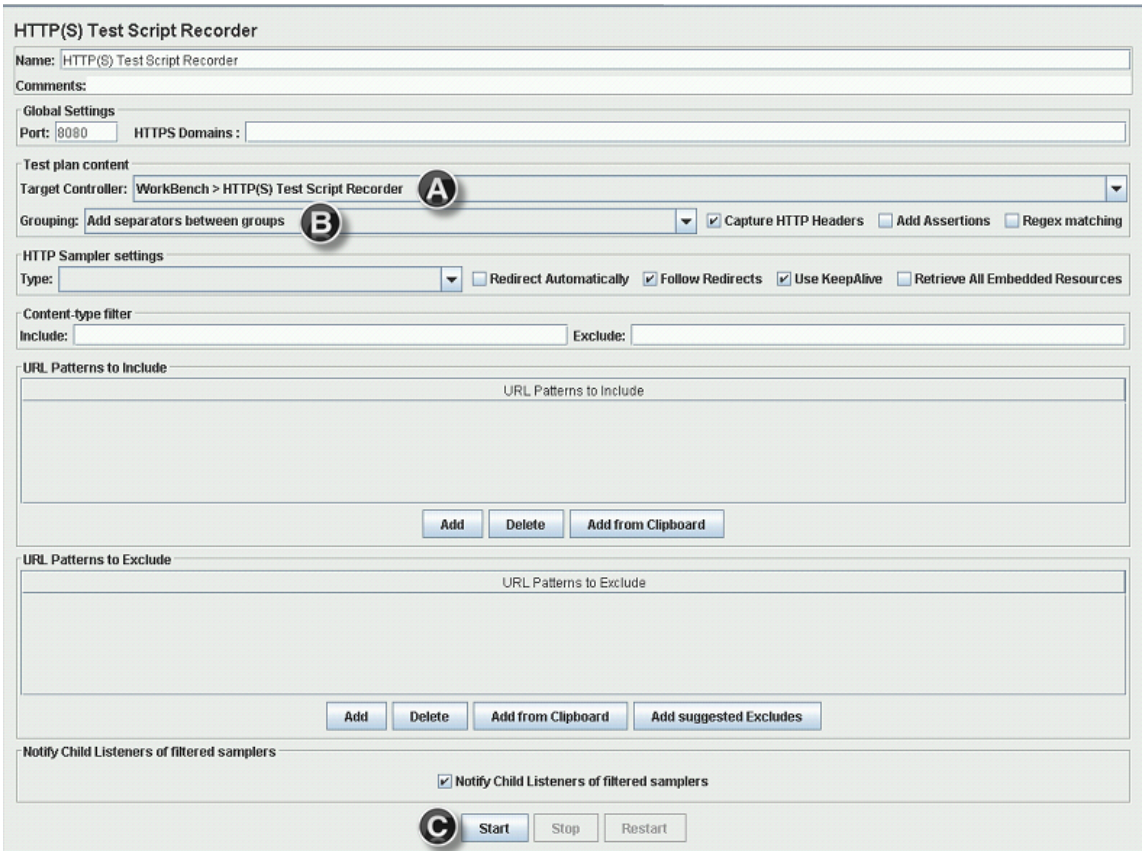
After you have recorded a workflow with Fiddler, you can record the same workflow again in Apache JMeter. Apart from practicing the workflow, this provides a side-by-side comparison of a functioning test execution, which is useful for future when you customize your test plan for repeated use. JMeter provides an HTTP test script recorder which functions almost exactly like Fiddler. This component allows us to interactively perform the workflow used in the previous sections, and to have JMeter capture each of the requests.

To record the workflow using Apache JMeter, perform the following:

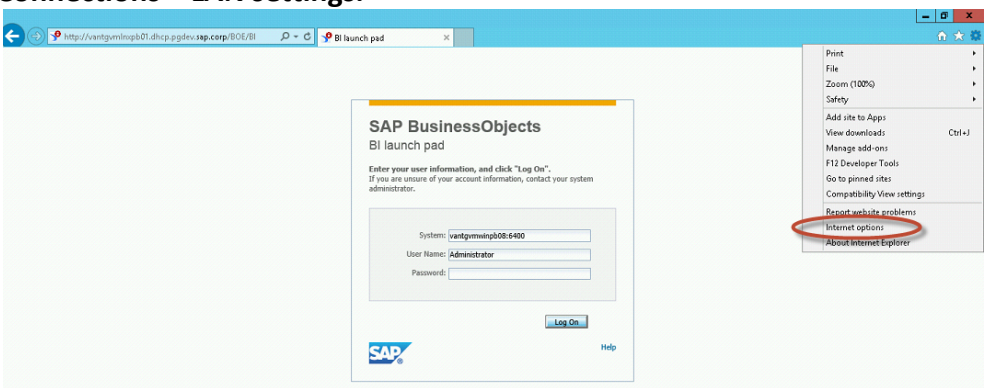
1. Launch **jmeter.bat** from the JMeter installation directory.
2. Right-click on the workbench component and add an **HTTP(S) Test Script Recorder** component.

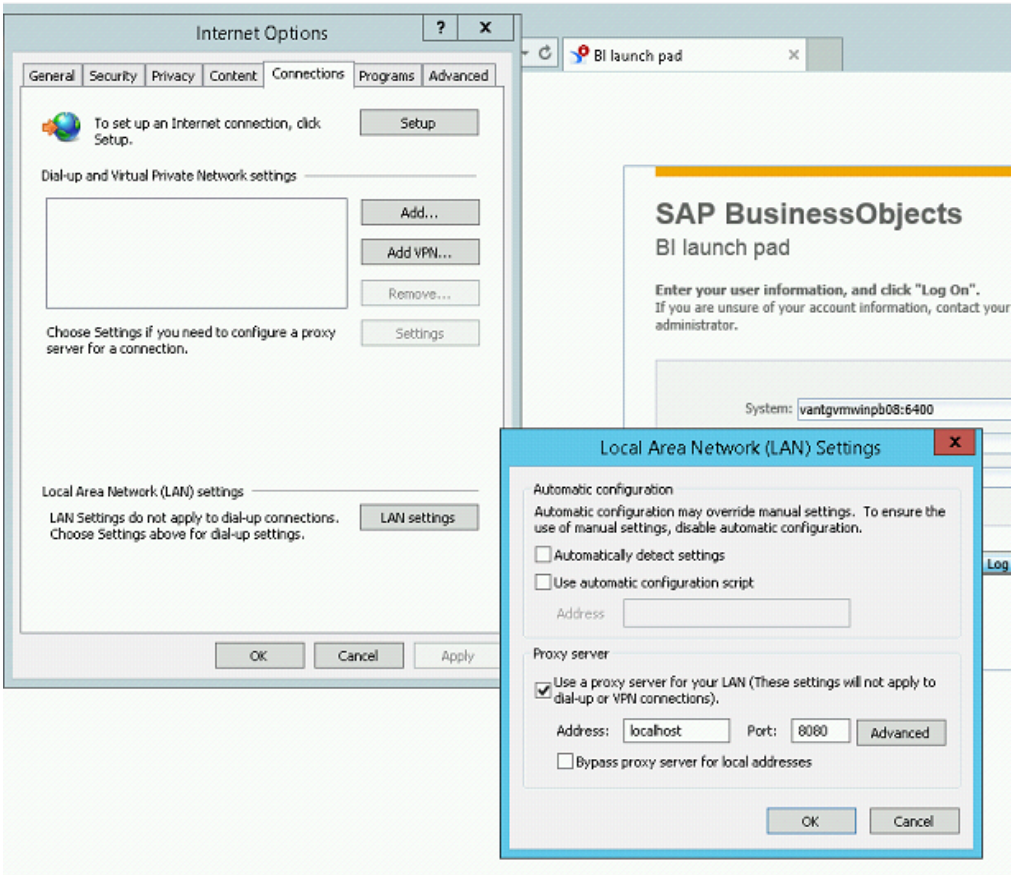


3. Configure the following options on the **HTTP(S) Test Script Recorder**.
    a. **Target Controller set to WorkBench > HTTP(S) Test Script Recorder**– This option stores the recorded URL requests under the workbench so you can easily cut and paste them into the Test Plan itself
    b. Grouping set to **'Add separators between groups' –** This option adds a simple controller in between requests where there is a pause of 1-2 seconds or longer. This can be helpful to logically determine where certain transactions begin/end (i.e. logon, navigate, view/refresh, etc)
    c. Choose **Start** the component so that it begins listening on port 8080 (as defined in the port setting).

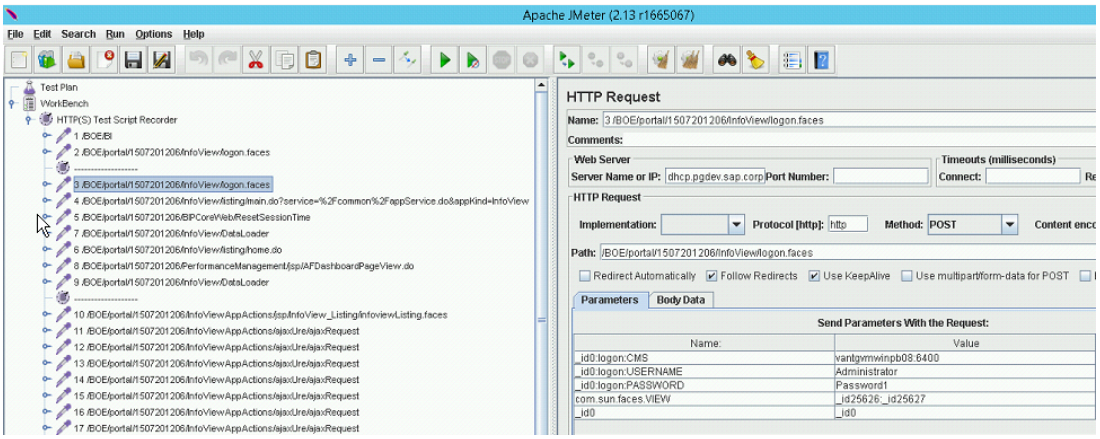


4. Launch **Internet Explorer**.
5. To access the proxy server configuration, navigate to **Internet Options > Connections > LAN settings.**





6. Set the proxy server to localhost: 8080 which routes all requests through JMeter.
7. Repeat the steps documented in Workflow 1 – View Webi Document "Sales Ranking - 1 Table" on demand.
8. Return to JMeter and ensure that the request has been recorded successfully.

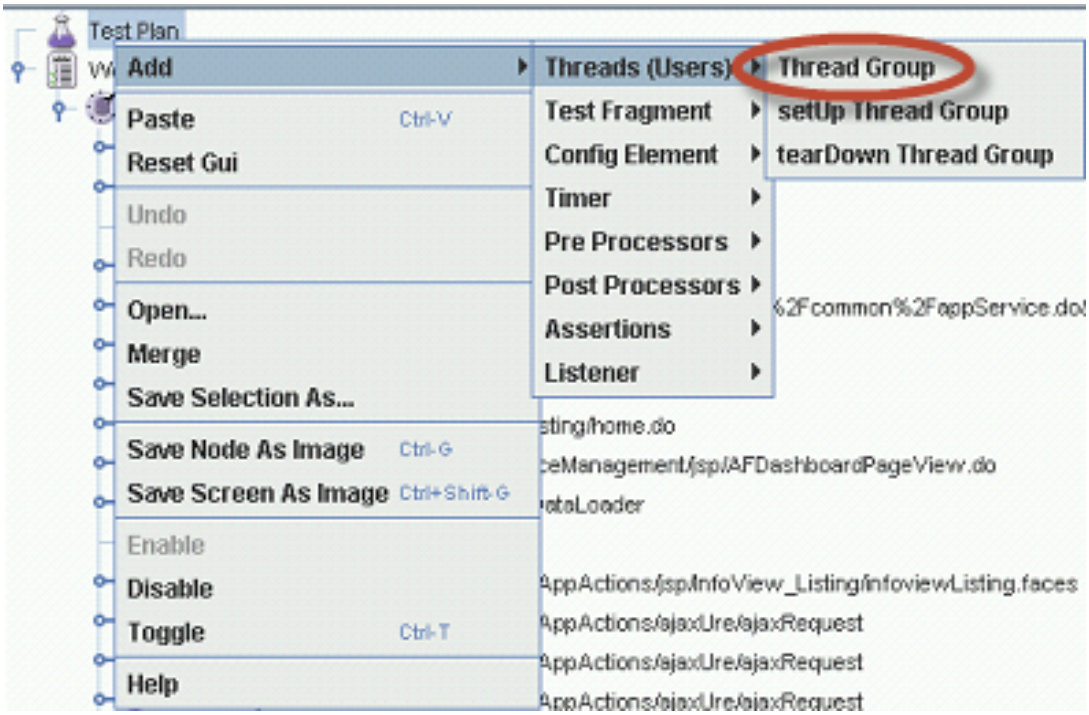


Moving the request to a test plan

After the Test Plan is recorded, we move the requests to thread groups and controllers to create a meaningful structure.

The procedure is as follows:

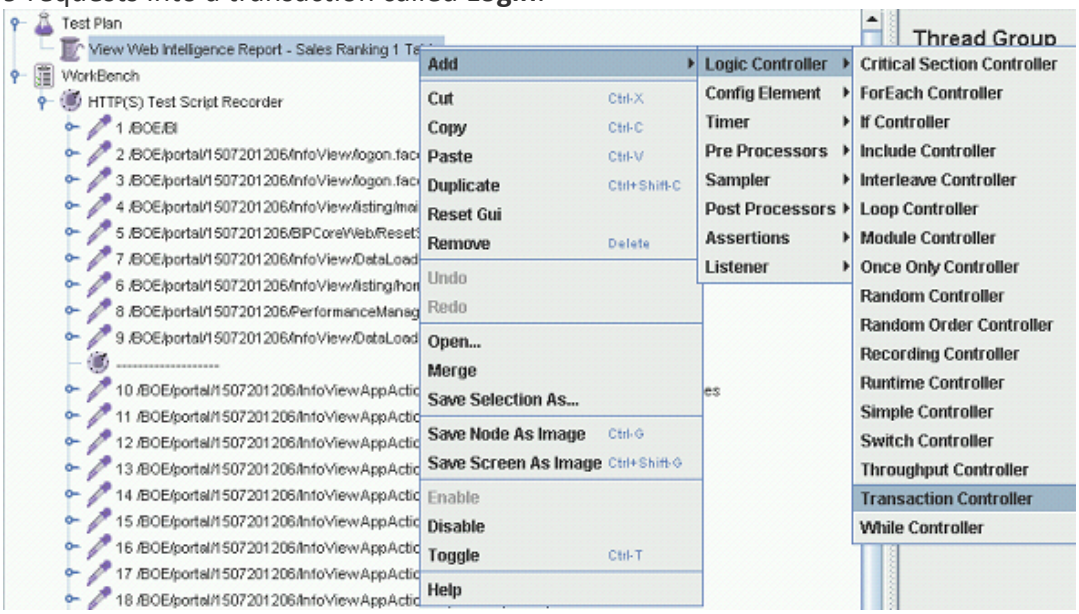
1. Right-click on the Test Plan and on the context menu Select **Add > Threads (Users) >**



2. In the thread group window, enter the Name field as "View Web Intelligence Report – Sales Ranking 1 Table".

3. Right-click the **Thread Group** and select **Add > Logic Controller > Transaction Controller**.
   We use 2 different kinds of controllers for this test, **transaction controllers** and **simple controllers**. The **transaction controller** allows us to measure the time a series of requests take to complete. In this fashion, we can group the first

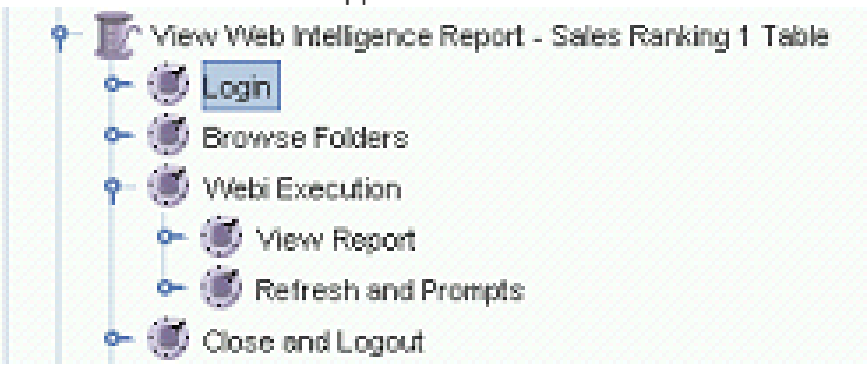
9 requests into a transaction called **Login**.



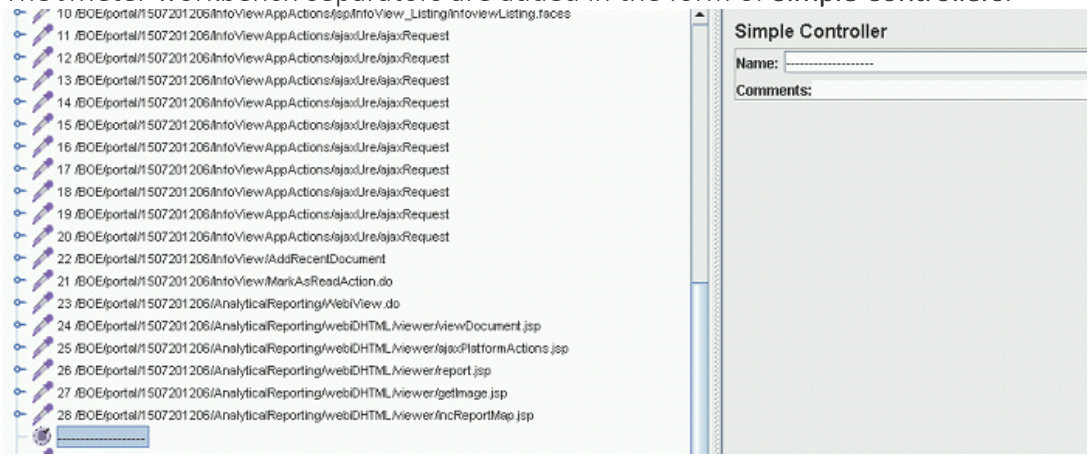
4. Name the **Transaction controller Login.**
5. Perform step 4 to add another **transaction controller** and name it **Browse Folders.** Add **Simple Controller** from the **Logic Controller** menu and name it **Webi Execution.**
6. A **simple controller** serves to organize other controllers.
7. Right-click the **Webi Execution** controller and add a new **transaction controller** and name it **View Report.**
8. Repeat step 4 and add another **transaction controller** and name it **Refresh and Prompts**.

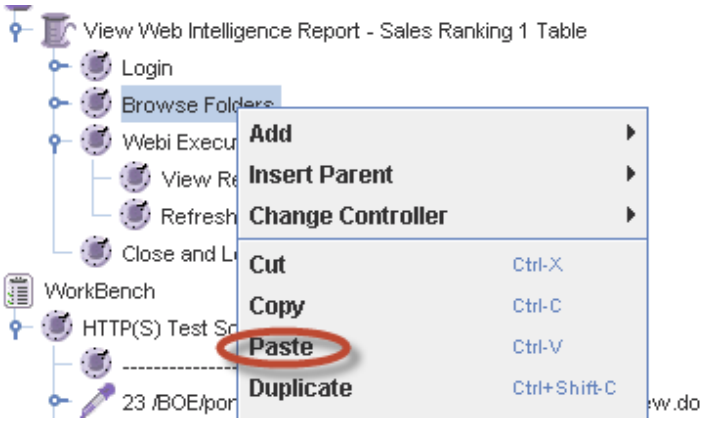
Information: Organize multi-step processes into Simple Controllers and Transaction Controllers for easier interpretation of test results. In this example, we divide the Webi Execution into 2 steps, opening the report, and refreshing the report. This way we can differentiate between the 2 transactions.

9. Repeat step 4 to add the last **transaction controller** and name it **Close and Logout**. The final structure should appear as shown below:



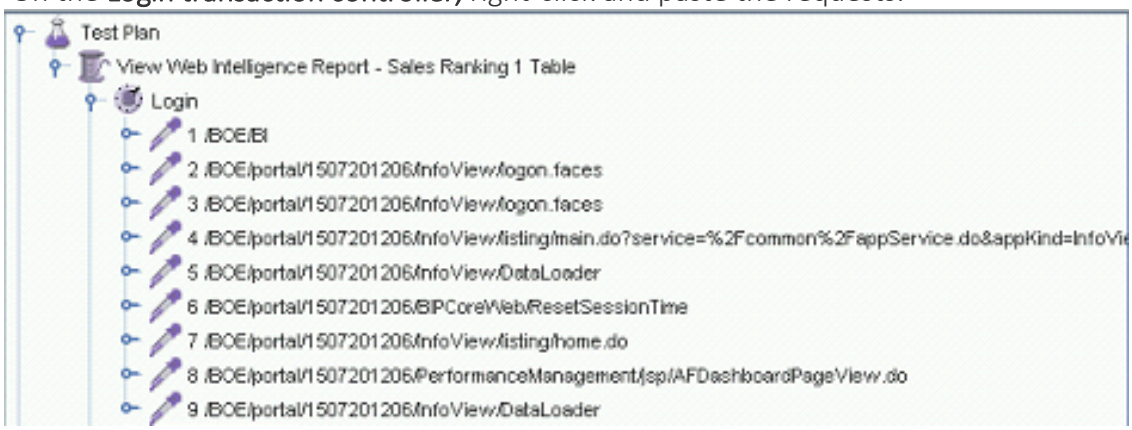
10. The JMeter workbench separators are added in the form of **Simple Controllers**.





The pause between transactions is JMeter's best guess. It is important to structure the requests perfectly. We can use our best estimate to move the requests to the test plan.

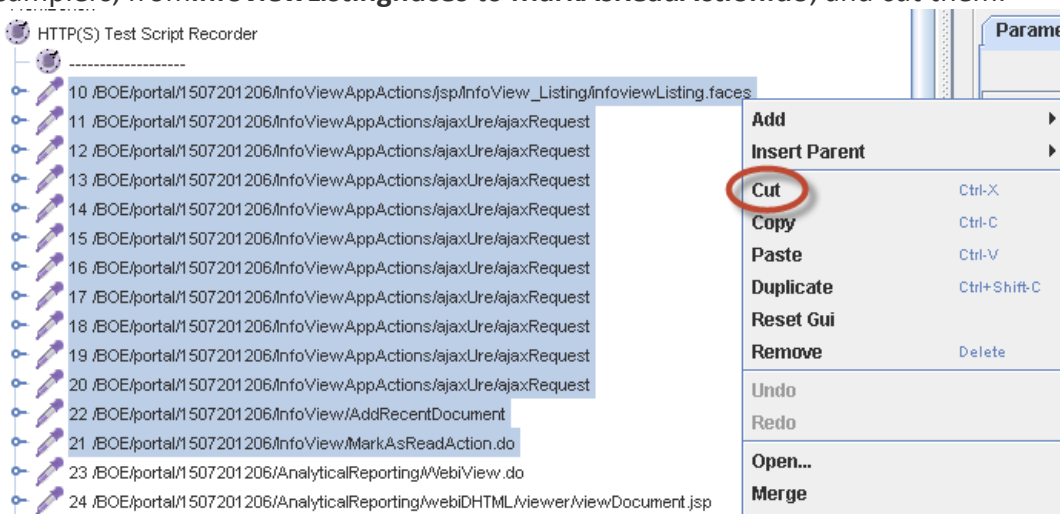
11. Select to highlight and cut the first 9 requests from **/BOE/BI** to **/InfoView/DataLoader**.
12. On the **Login transaction controller,** right-click and paste the requests.



13. Create the **Browse Folders** transaction.

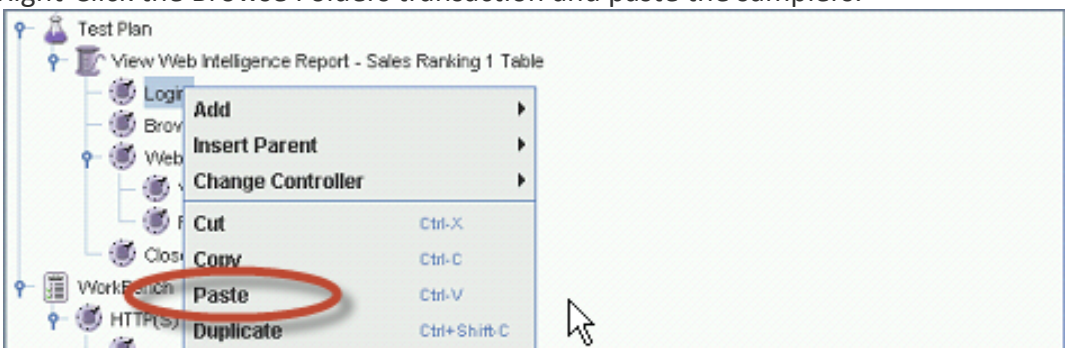
We will create the browse folder transactions as an example because JMeter do not pick up any separator during the recording process. In the below shown screenshot, the division of request is for our convenience. Highlight the next 13

samplers, from**infoviewListing.faces** to **MarkAsReadAction.do**, and cut them.



Warning: If you're not sure where to split a transaction, look for a change in web application directory as shown above. The change from InfoView to AnalyticalReporting suggests a natural break in the browse folders transaction.

14. Right-Click the Browse Folders transaction and paste the samplers.



15. Repeat steps 13-14 for the next set of samplers and paste it into the **Webi Execution > View Report** transaction.
    While JMeter did have separators into the recording, you can also see the **refreshDocument.jsp** sample which indicates the start of the refresh transaction.
16. Repeat steps 13-14 for the **Webi Execution > Refresh and Prompts** transaction, using the same web application directory change as before to identify the end.

The refresh transaction consists of the 8 requests between **refreshDocument.jsp** and **processClose.jsp**

17. Repeat steps 13-14 for the **Close and Logout** transaction by cutting and pasting the remaining samplers.

Customizing test plan for dynamic data

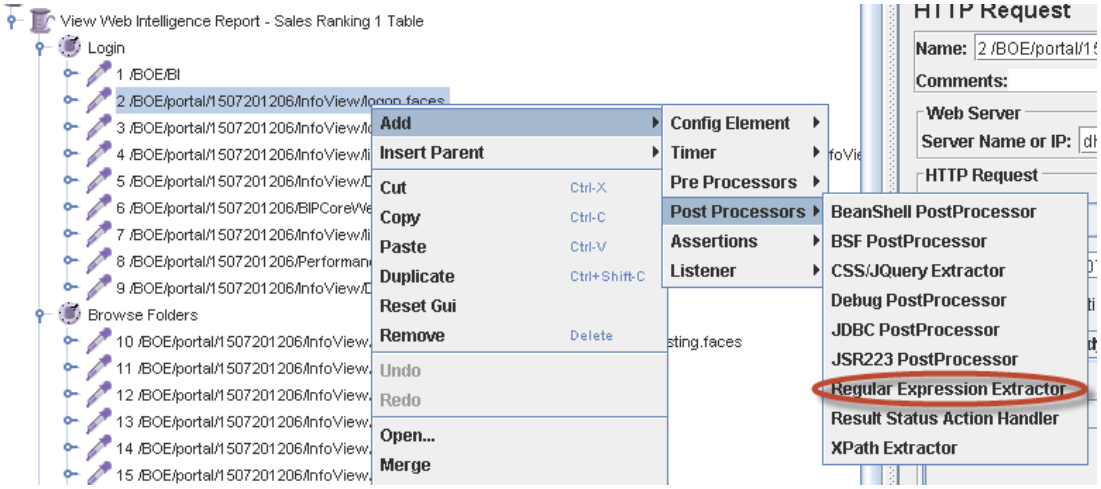
At this point, you have a test plan with the necessary HTTP samplers to open and refresh a Web Intelligence report. However, there is a significant amount of information changing every time you interact with the system, and some information changes only when different users access the system. Some of this data consists of usernames or passwords, session tokens, and temporary image files. Playing the recorded samplers again might result in errors from the BI Platform, so it is necessary to extract this dynamic data and re-use it in appropriate places.

The concept of extracting dynamic data using regular expressions is explained in the tutorials at the beginning of this section. In short, JMeter provides a component which you can attach to any URL that allows you to match content from headers or body of request or response, and pipe it into a local variable that JMeter replaces with the extracted value at runtime. We can use this to simulate multiple users, pass different prompt values into a parameter, and control folder navigation in the BI Launch Pad, just to name a few.

- Login Transactions
- Browse Folder Transaction
- View Report Transaction
- Refresh and Prompt Transaction

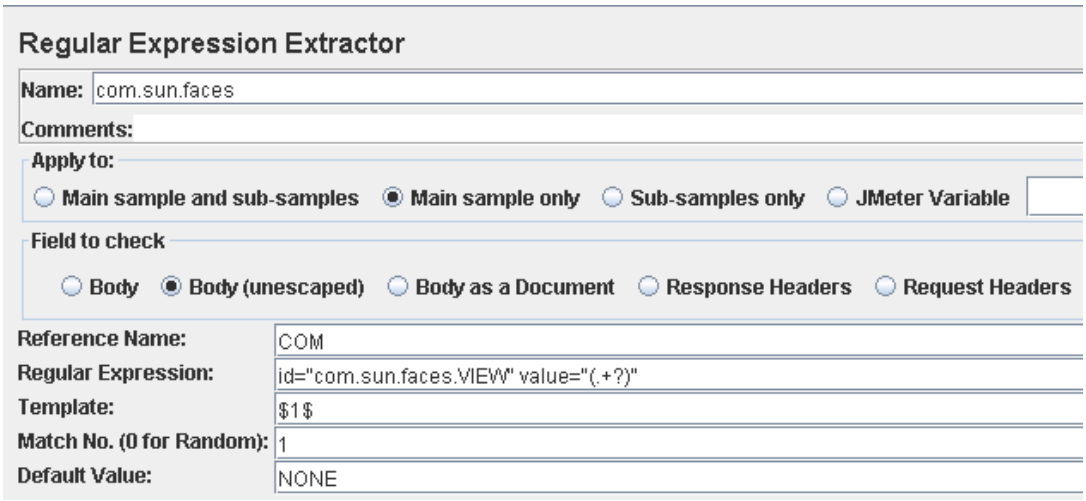
Login Transactions

1. Right-click the first sampler labeled **/InfoView/logon.faces** and select **Add > Post Processors > Regular Expression Extractor.**



2. Enter the details in regular expression extractor to configure using the information in the table below.

Field	Configuration Information
Name	com.sun.faces
Apply to	Main sample only
Filed to check	Body (unescaped)
Refrence Name	COM
Regular Expression	id="com.sun.faces.VIEW" value="(.+?)"
Template	1
Match No.	1
Default value	NONE

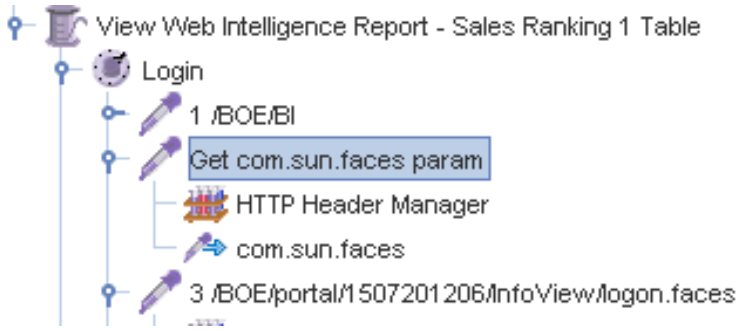


3. If we look at the same request in Fiddler, the response body in a functional request appears as shown below:

```
<input type="hidden" name="com.sun.faces.VIEW" id="com.sun.faces.VIEW" value="_id27465:_id27466" /><input type="hidden" name="_id0"
value="_id0" />
<script type="text/javascript">
<!--
function clearFormHiddenParams__id0(curFormName) {
  var curForm = document.forms[curFormName];
}
//-->
</script>
</form>
```

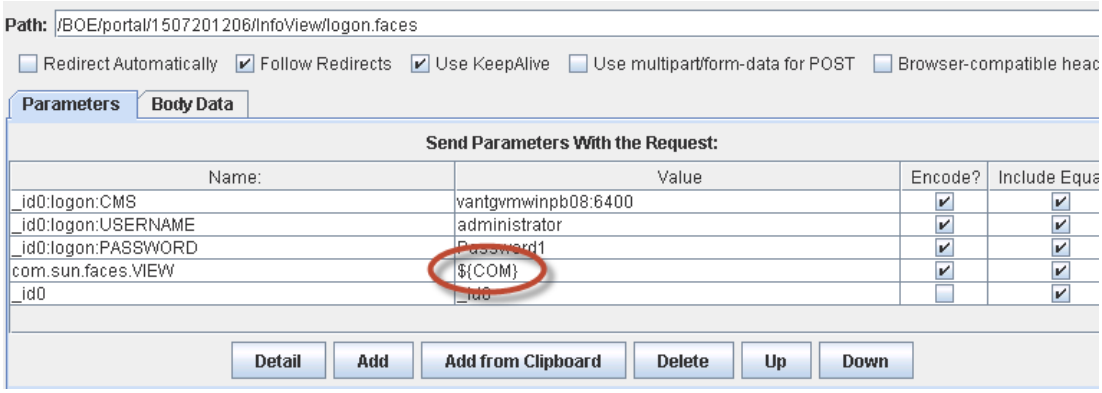
In this case, the regular expression is configured to extract all characters between the double-quotes in value property. Without this value the request does not run and the test plan generates errors. We will evaluate how to use the **View Results Tree** component to troubleshoot a test plan below.

4. Rename the sampler to **Get com.sun.faces param** to make it identifiable, even when collapsed, that something important is happening in this request.



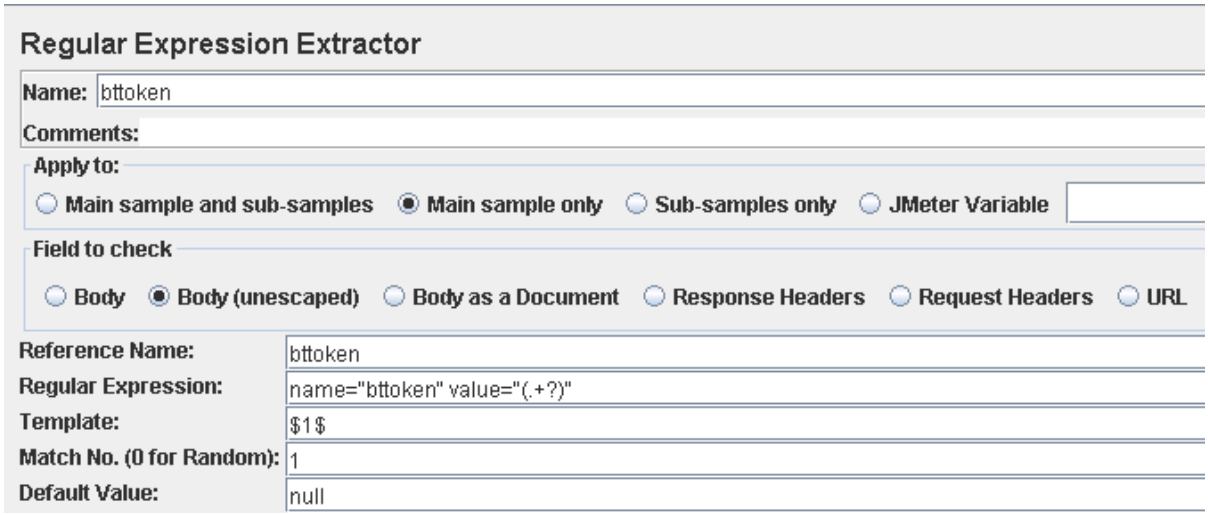
5. Select the next sampler, also called **/InfoView/logon.faces**, and change the **com.sun.faces.VIEW** parameter in the response body to **${COM}**

This syntax tells JMeter to replace the value inside of the brackets with the extracted variable COM.



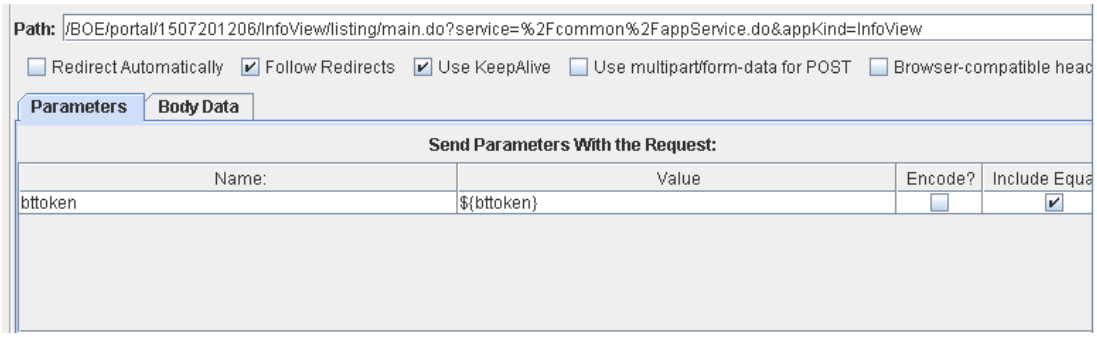
6. Right-click the sampler and add a regular expression extractor with the following values:

Field	Configuration Information
Name	bttoken
Apply to	Main sample only
Field to check	Body (unescaped)
Reference Name	bttoken
Template	1
|Match No.	1
Default Value	null

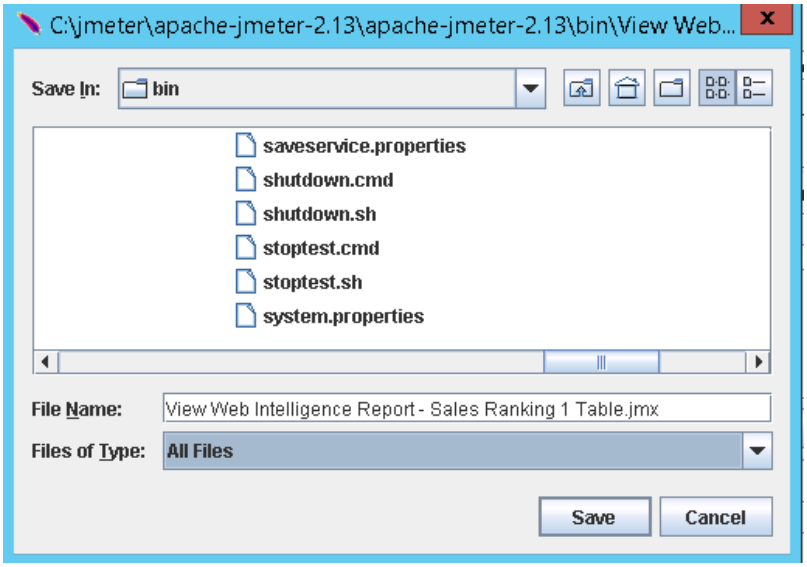


7. Rename the sampler to **pass credentials**.

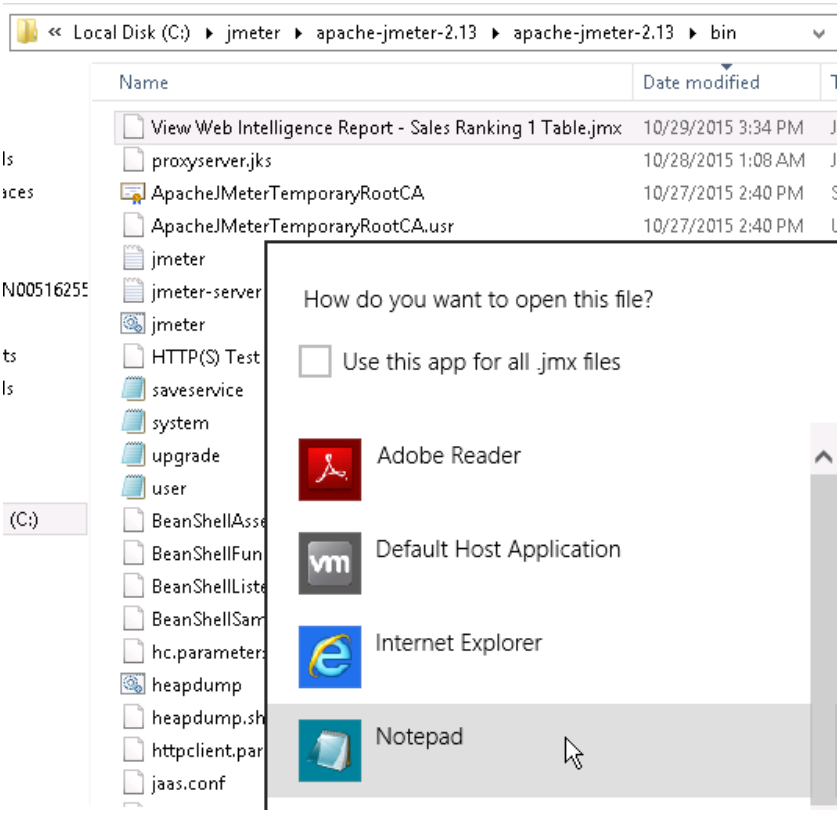
8. Select the **main.do** **sampler** and copy the **bttoken** value to your clipboard.
   The string is as shown: **MDAwRGRGbDJDZjY2bElKYTtuaWw0Zl9aTVNkYV9TNDAEQ**
   Once you have copied it, update the value to **${bttoken}** to reference the value you extracted in the previous step.



9. Save the test plan in the default **jmeter\bin** directory and call it **View Web Intelligence Report - Sales Ranking 1 Table**.

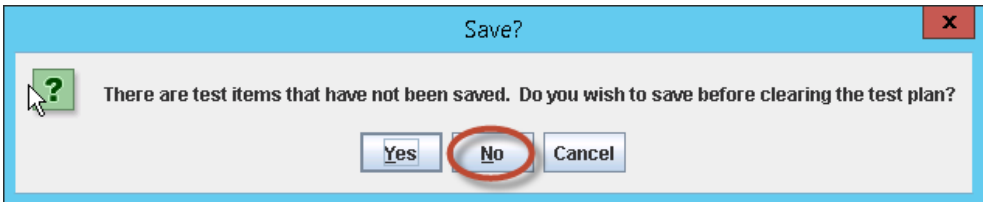


10. Browse to the directory you saved the file and open it using a text editor such as Notepad.

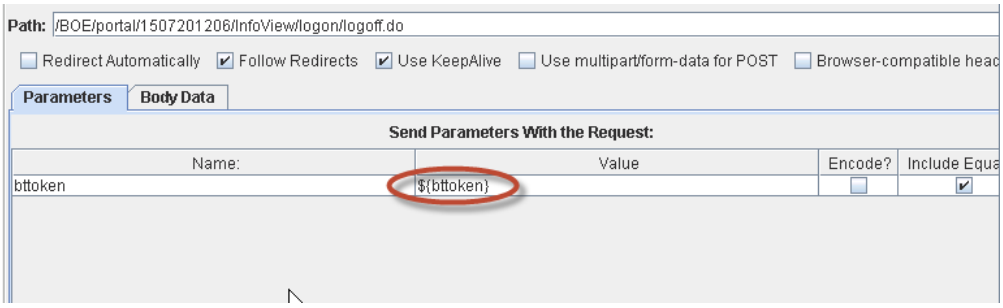


11. Use the **Edit > Replace** to replace **bttoken** value, **MDAwRGRGbDJDZjY2bElKYTtuaWw0Zl9aTVNkYV9TNDAEQ** with **${bttoken}**

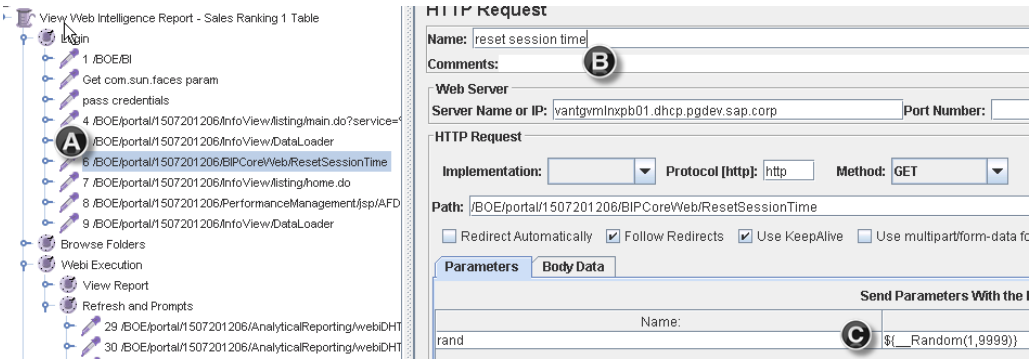
12. Save the file, return to JMeter, and reload the test plan. If you are prompted to save click No so that the changes are not overwritten.



13. Expand the **Logout** transaction and select logoff.do to observe that the bttoken is updated to reflect the JMeter variable.



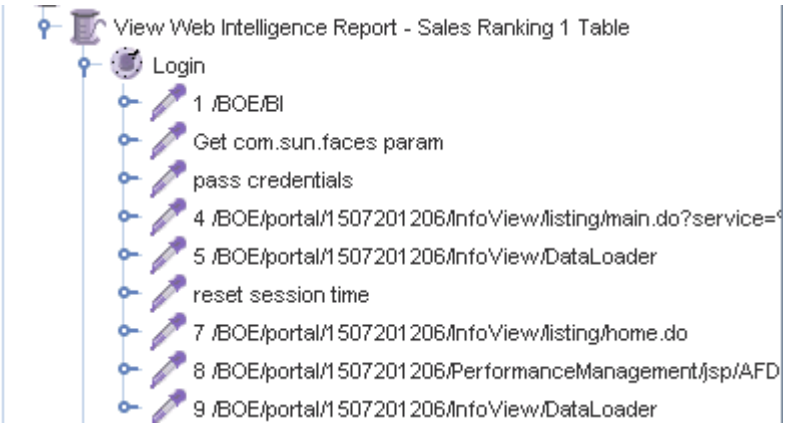
14. Return to the **Login** transaction and update the following details.



   a. Select the **ResetSessionTime** sampler.
   b. Rename the sampler to **reset session time**
   c. Modify the value of the **rand** parameter using a JMeter function. This function picks a random number between 1-9999 with each execution to pass to the servlet **${__Random(1,9999)}**

Information: There are variety of functions available for use within JMeter, all prefaced with the double underscore. You can find more details in the JMeter User's Manual: http://jmeter.apache.org/usermanual/functions.html

15. Confirm the Login transaction looks similar to the screenshot below and collapse the Transaction Controller.

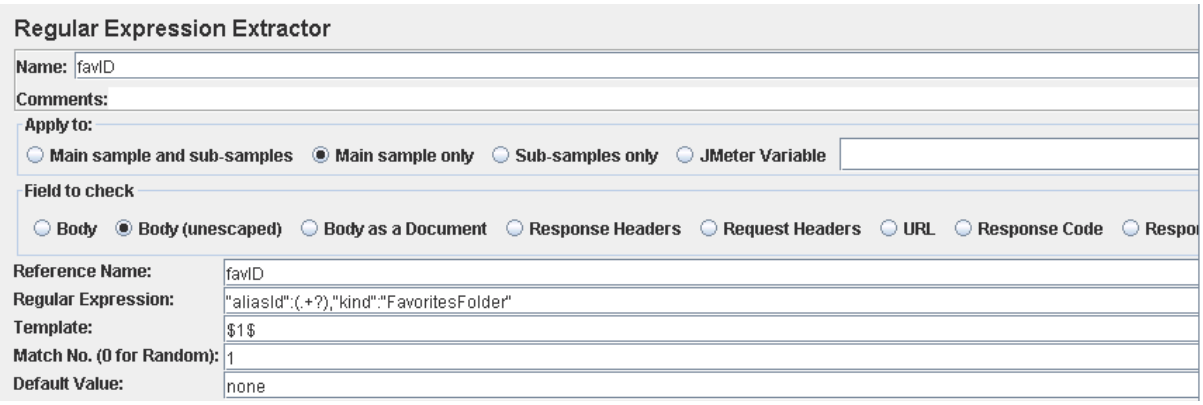


Browse Folder Transaction

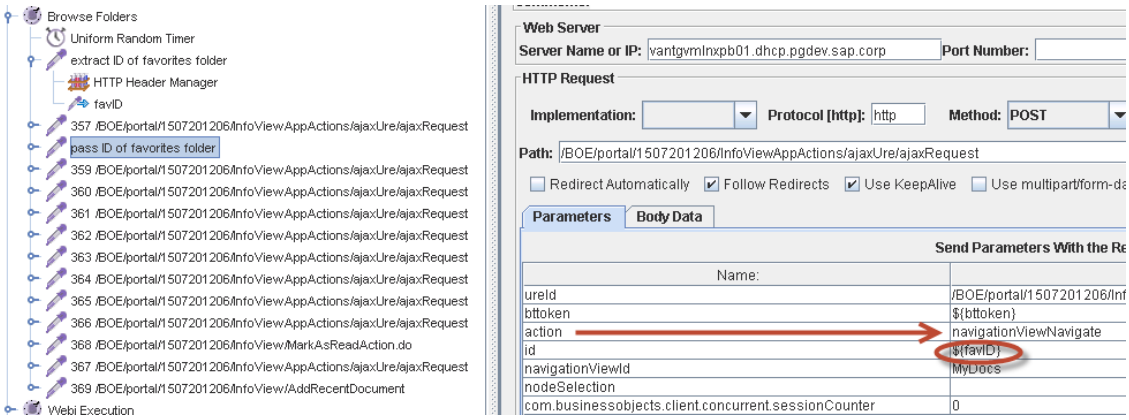
Browsing folder structure within the BI Launch Pad relies on AJAX call that return JSON describing the state of navigation. There is only one value required here, the SI_ID of the user's favorites folder.

1. Select the **infoViewListing.faces** sampler at the top of the transaction and rename it to **extract ID of favorites folder**
2. Right-click the renamed sampler and add a Regular Expression extractor post-processor component. Configure the sampler using the information from the below table.

Field Name	Configuration Information
Name	favID
Apply to	Main sample only
Field to check	Body (unescaped)
Reference Name	favID
Regular Expression	"aliasId":(.+?),"kind":"FavoritesFolder"
Template	1
Match No.	1
Default Value	none

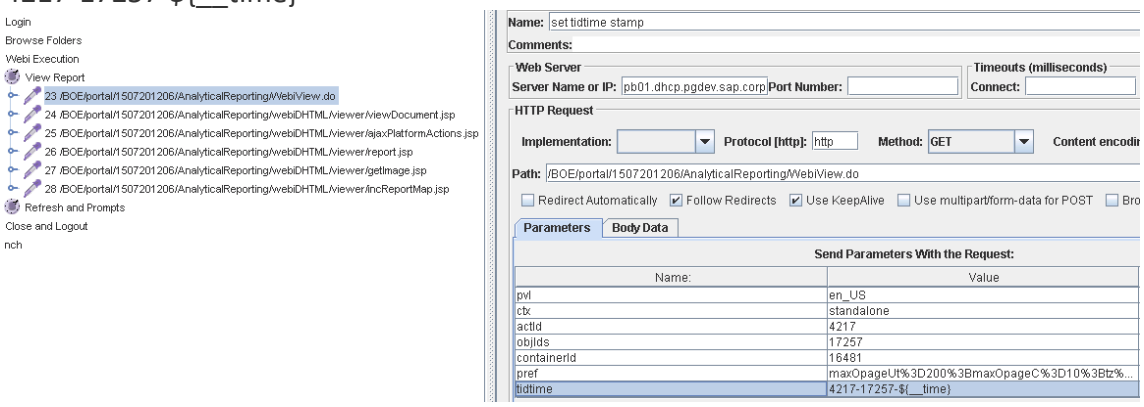


3. Select the 3rd instance of **ajaxRequest** which has a property of **action = navigationViewNavigate**. Rename the sampler to **pass ID of favorites folder.** Update the **id** property to use the extracted favorites ID **${favID}** from Step 2. Collapse the **Browse Folders** transaction when finished.



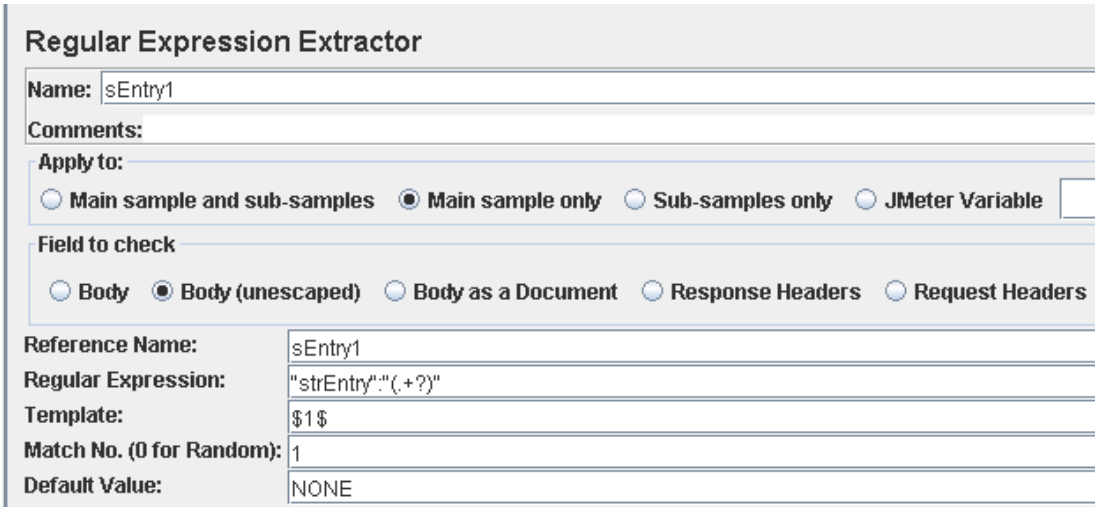
View Report Transaction

1. Select the WebiView.do sampler, and rename to set tidtime stamp.
2. Locate the tidtime property from the parameters list.The tidtime property contains a timestamp that is calculated every time the request is made. The format of the property is 4217-17257-1445982052106 where the first 2 values are ID(s) and the last value is a timestamp in the format of ms since the Epoch.
3. Update tidtime so that JMeter includes a function to calculate the current time: 4217-17257-${__time}



4. In the list, select the **viewDocument.jsp** sampler, and rename it to get sEntry1. Add a post-processor regular expression extractor and name it **sEntry1.**

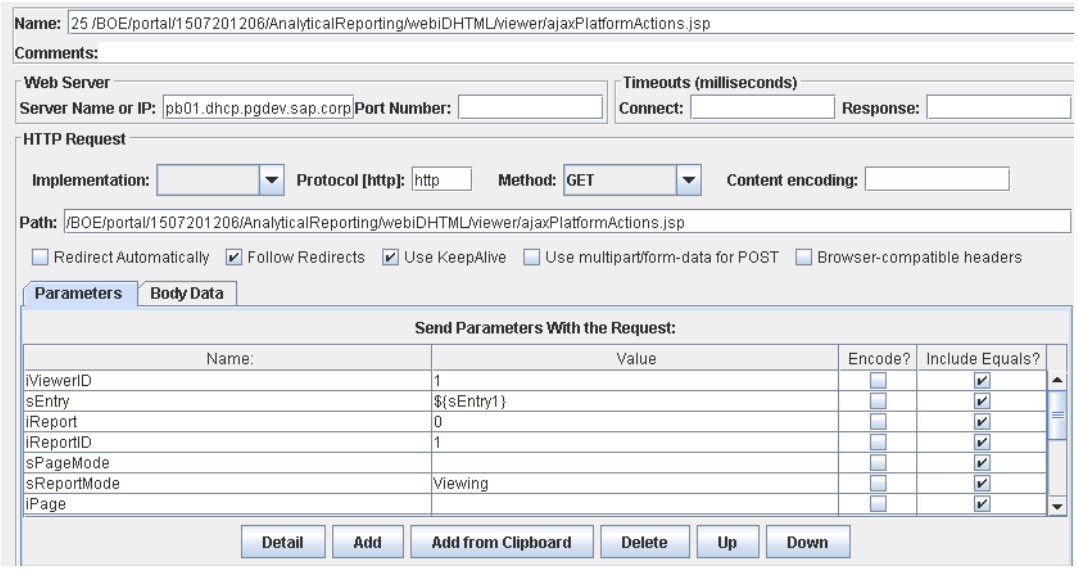
Field Name	Configuration Information
Apply to	Main sample only
Field to check	Body (unescaped)
Reference Name	sEntry1
Regular Expression	"strEntry":"(.+?)"
Template	1
Match No.	1
Default Value	NONE



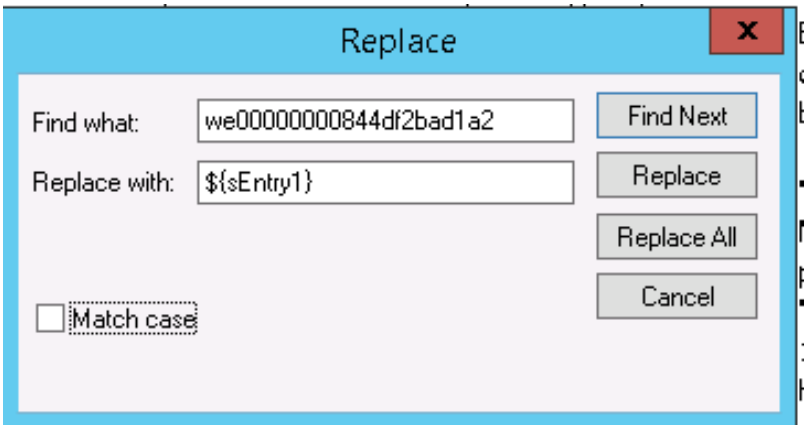
Information: An sEntry is a token that represents the state of a Web Intelligence document. In this case, the sEntry represents the state of the document when it is initially opened.

5. In the list select the **ajaxPlatformActions.jsp** sampler, and copy the **sEntry** value to your clipboard. The **sEntry** is likely to look like the string shown below.
   we00000000844df2bad1a2
   Replace the value with the JMeter variable **${sEntry1}** and save the test plan when

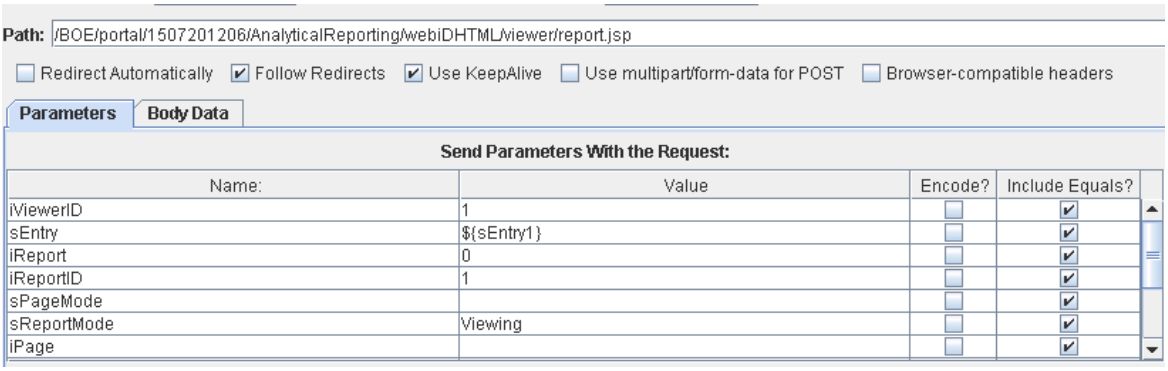
finished.



6. Open the test plan in Notepad.
7. Use the Replace options to replace all instances of the sEntry with ${sEntry1}.



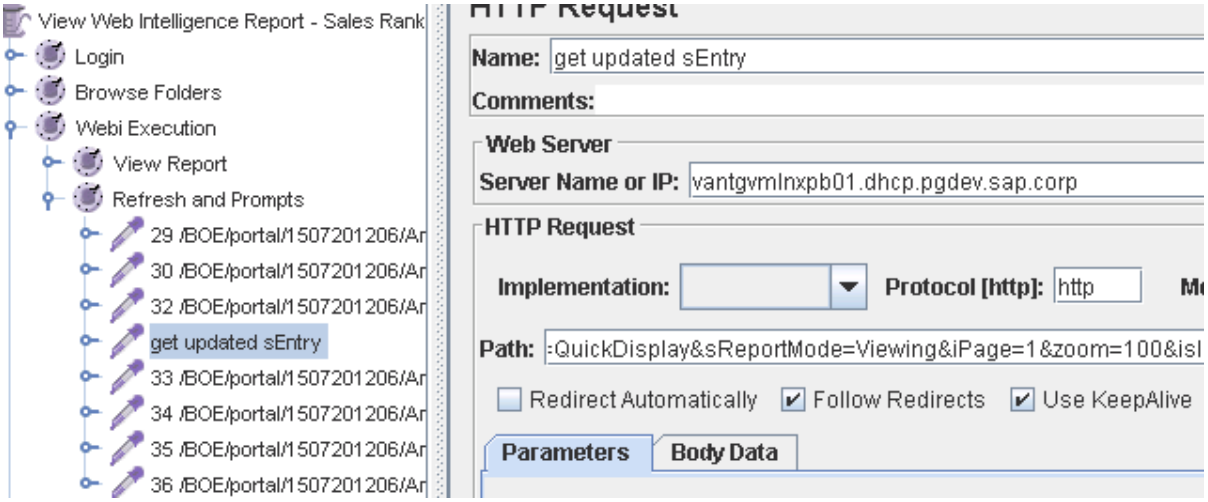
8. Save the test plan in Notepad and return to JMeter.
9. Reload the test plan in JMeter as in the previous step and note that successive samplers have been updated. Collapse the View Report transaction and continue to the next section.



Refresh and Prompt Transaction

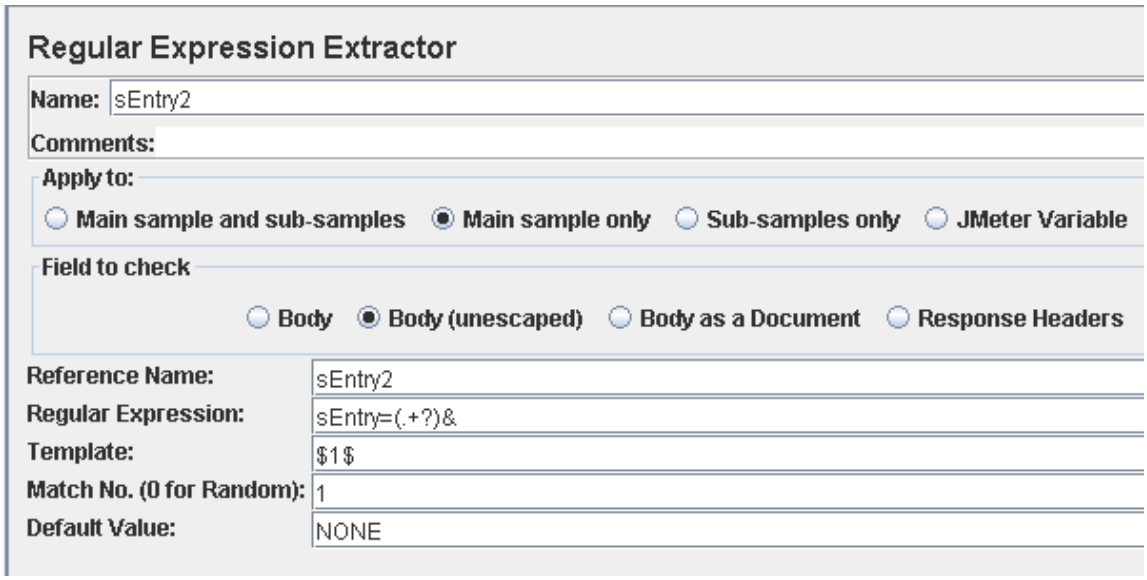
In the **Refresh and Prompts** transaction we extract a new **sEntry** that represents the change in state of the Web Intelligence document once it is refreshed. This is also the section that comprises the submission of the prompts panel.

1. Select the **processPrompts.jsp** sampler, and rename it to **get updated sEntry**. **processPrompts.jsp** is also the sampler where you could send dynamic values to a query prompt in a Web Intelligence document though we do not do it here.

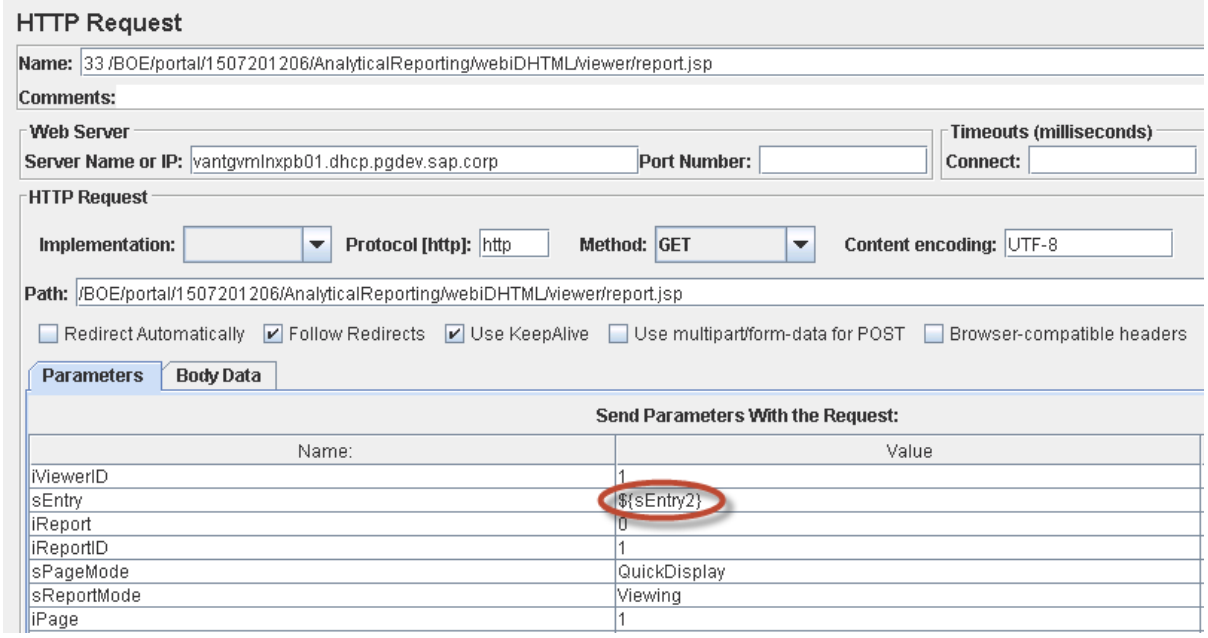


2. Add a Regular Expression extractor to the request as a post-processor and name it **sEntry2**. Configure using the below information.

Field Name	Configuration Information
Field to check	Main sample only
Reference Name	Body (unescaped)
Regular Expression	sEntry2
Template	1
Match No.	1
Default Value	NONE
Apply to	sEntry2

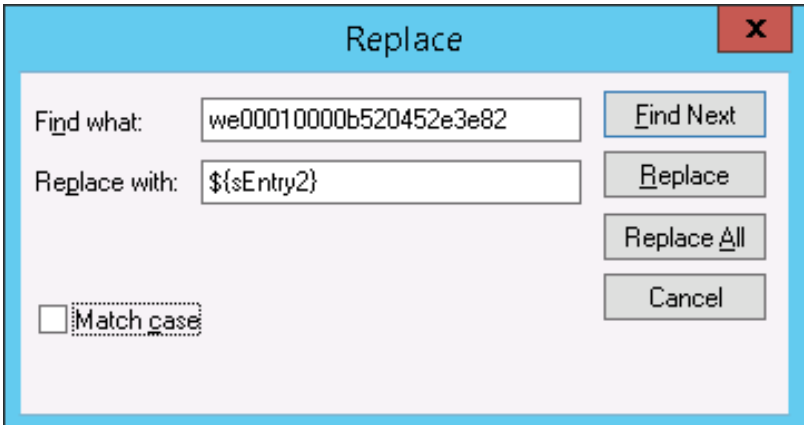


3. Select the report.jsp sampler, and copy the sEntry. Replace the value with ${sEntry2} and save the test plan:
   we00010000b520452e3e82



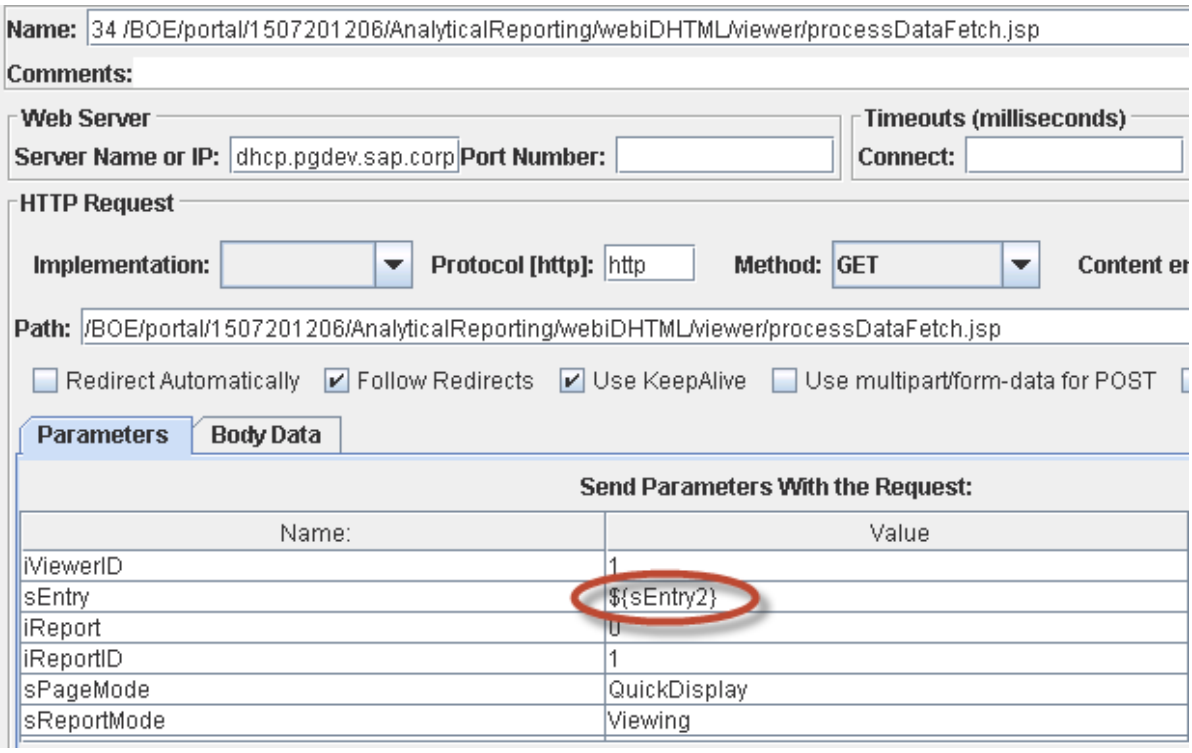
4. Open the test plan in Notepad

5. Use Replace options to Replace all occurrences of the latest **sEntry** with **${sEntry2}.**



6. Save the file.
7. Reload the test plan in JMeter and select the sampler **Webi Execution > Refresh and Prompts > processDataFetch.jsp**.

Warning: The sEntry parameter is updated to contain **${sEntry2}.** There are no samplers that require customizing in the Logout transaction other than to provide the correct bttoken value. This was done as part of the replace operation in the Login Transactions section.



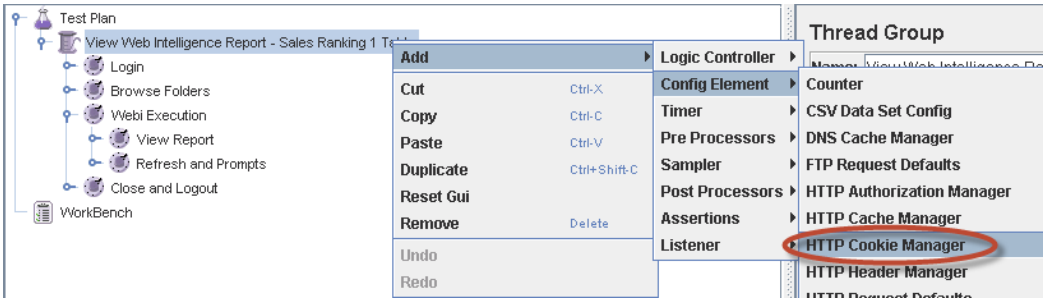
Final Configuration Components

The final step before running the test plan is to add some configuration and monitoring related components to the test plan. We add a component to manage cookies, one to

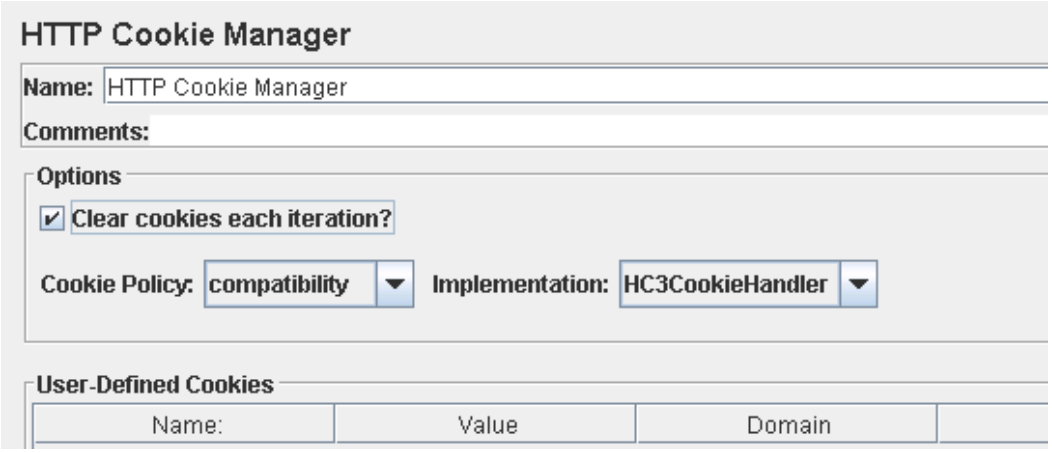
monitor the progress of the test plan execution, and a CSV configuration file to simulate different users.

The steps involved in final configuration are as follows:

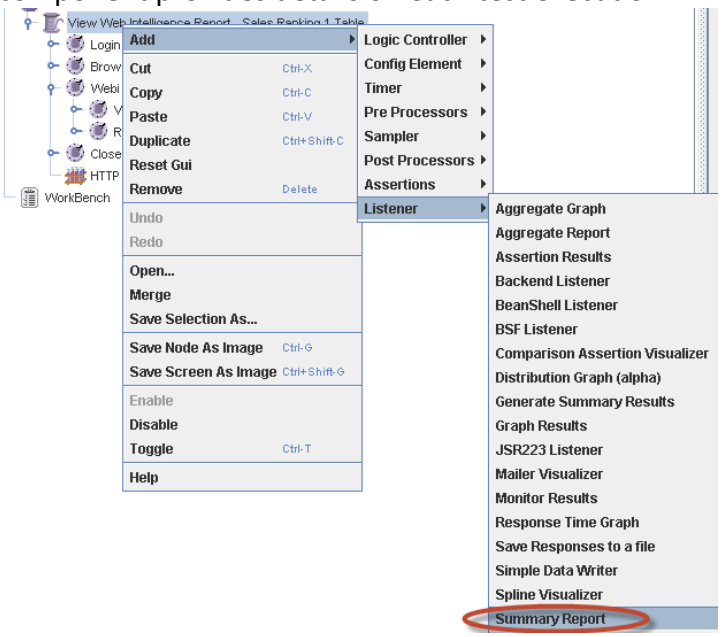
1. Right-click the thread group and select **Add > Config Element > HTTP Cookie Manager**



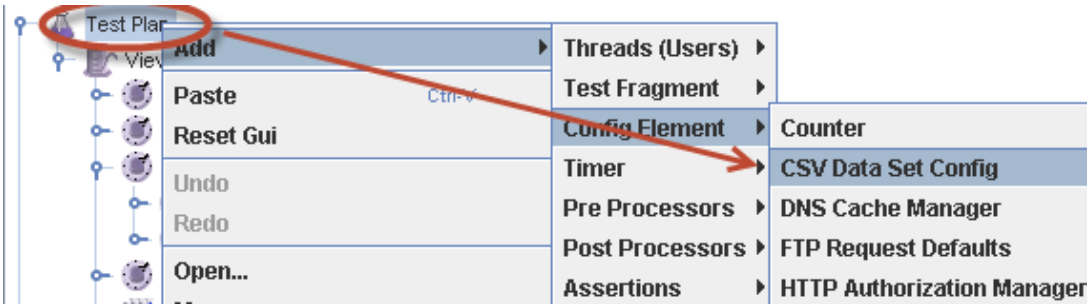
2. Check the **clear cookies each iteration?** option. It is very important to clear the cookies to ensure distinct sessions each time a thread runs.



3. Right-click the thread group and select **Add > Listener > Summary Report**. This component provides details on each test execution.



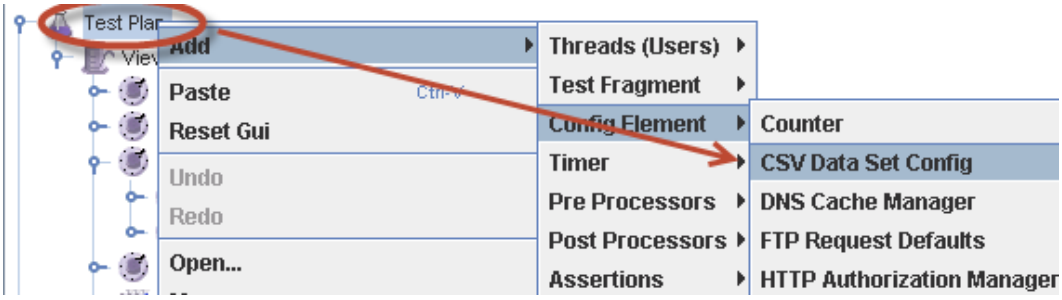
4. Right-click on the **Test Plan** and **select Add > Config Element > CSV Data Set Config**. This allows us to pass dynamic values into a JMeter test plan for each thread. In this Test Plan, we use it to simulate 10 distinct users in the BI Platform.



5. To create users in the BI Platform, use the information shown in the table below.

Username	Password
User01	Password1
User02	Password1
User03	Password1
User04	Password1
User05	Password1
User06	Password1
User07	Password1
User08	Password1
User09	Password1
User10	Password1

6. Copy the **users.csv** file to the same directory where the Test Plan has been saved (C:\jmeter\apache-jmeter-2.13\apache-jmeter-2.13\bin)
7. Configure the **CSV Data Set Config** component as shown below.



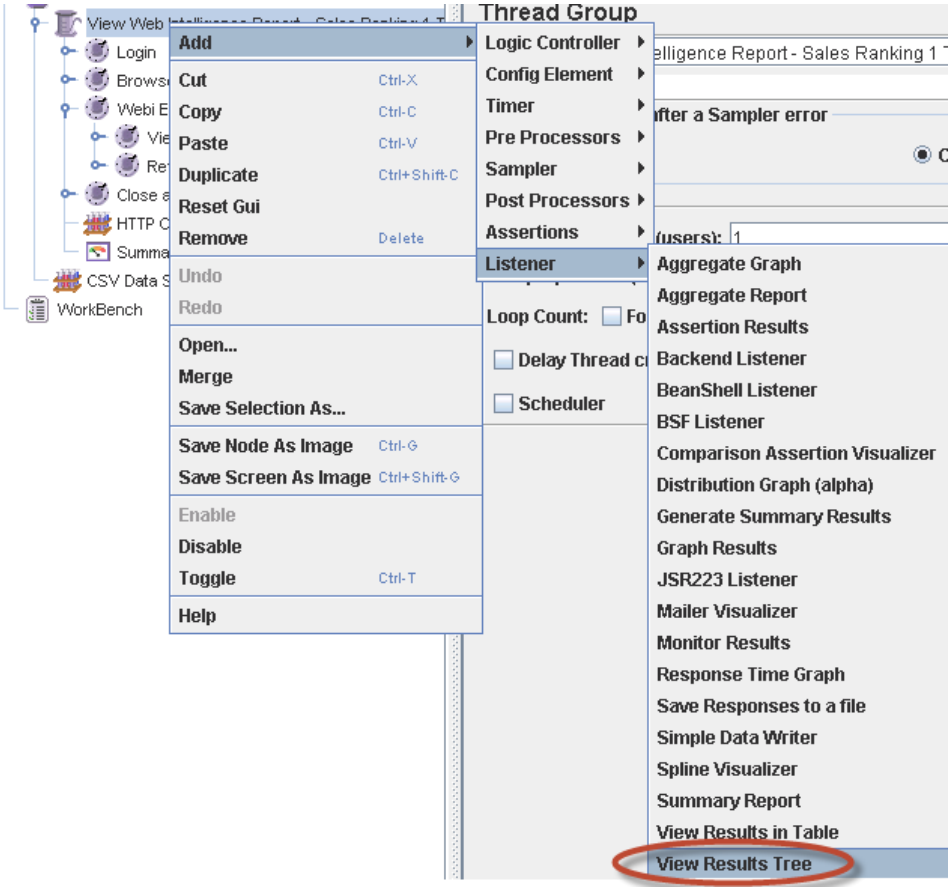
Expand the **Login** transaction, select the **pass credentials** sampler and update the **_id0:logon:USERNAME** and **_id0:logon:PASSWORD** property values to **${user}** and **${pass}** respectively.

Initial Run

To run a large-scale test plan, we recommend you to run the test for a single user and monitor the output to ensure that the test plan is working as expected. Even a small typographic error can result in a test failure. **View Results Tree** listener and a **Debug** sampler are added and the test is run with a single user thread, in order to test it.

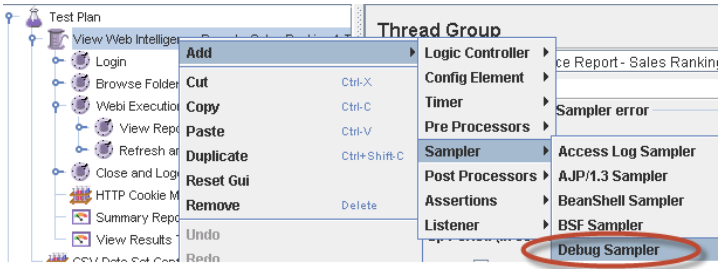
The procedure for initial run is executed as follows:

1. Right-click the thread group and select **Add > Listener > View Results Tree.**



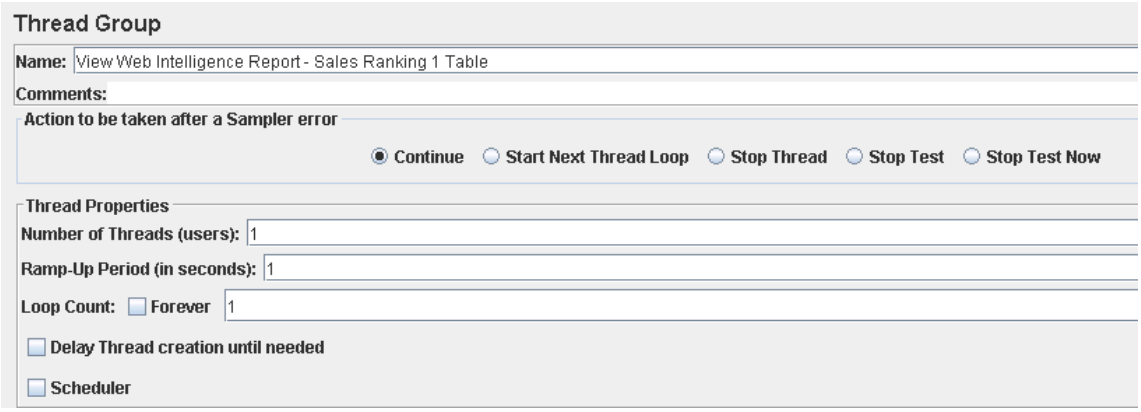
Warning: View Results Tree records every HTTP request or response associated with your test plan. It should be used in small debugging scenarios, otherwise it will have a negative impact on the integrity of test. You should delete this component before running tests with more than 10 users.

2. Right-click on thread group and select **Add > Sampler > Debug Sampler**.

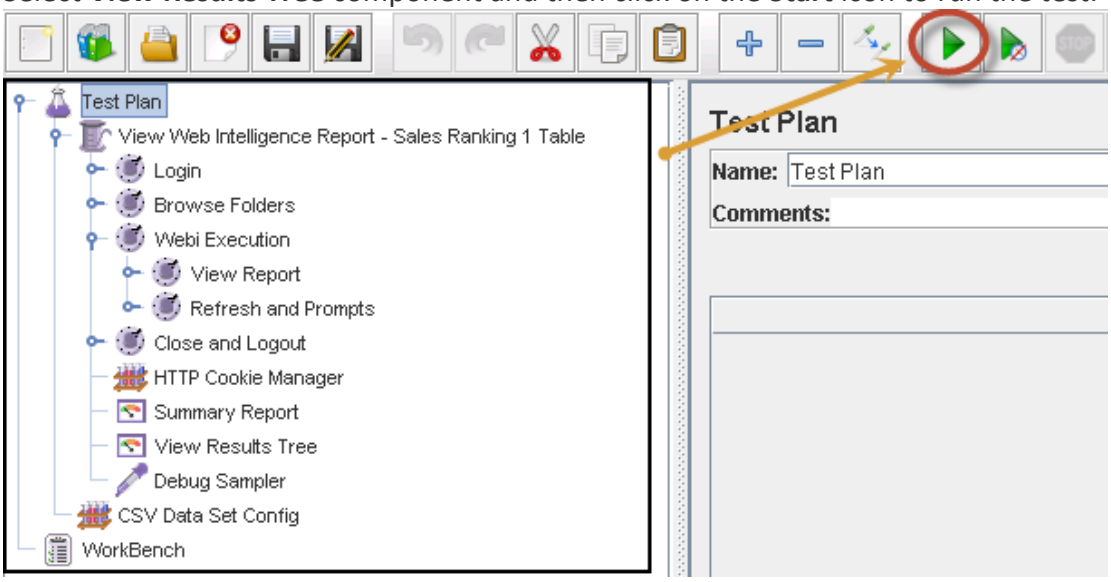


Warning: You can use the **Debug Sampler** in conjunction with **View Results Tree** to print the contents of all **JMeter** variables. By doing so, your regular expression extractors store the values you expect.

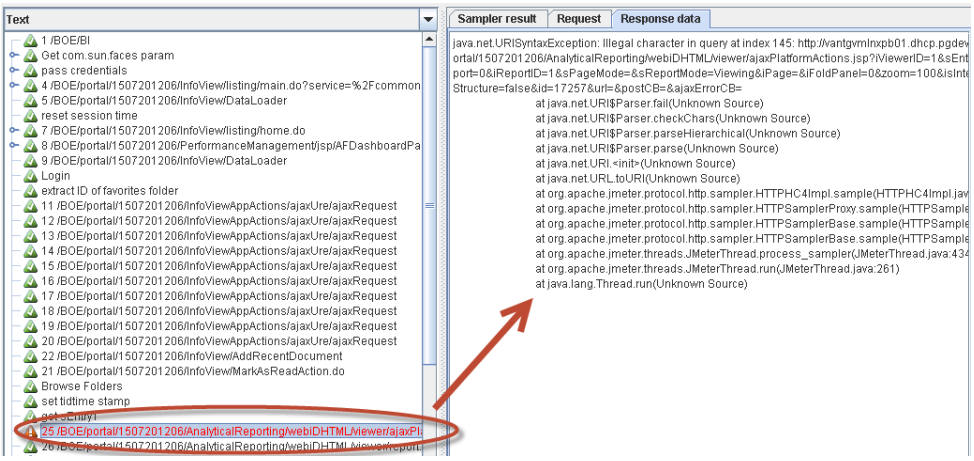
3. Select the thread group **View Web Intelligence Report - Sales Ranking 1 Table** and set all available properties to **1**. Ensure that your test plan looks like the image given below.

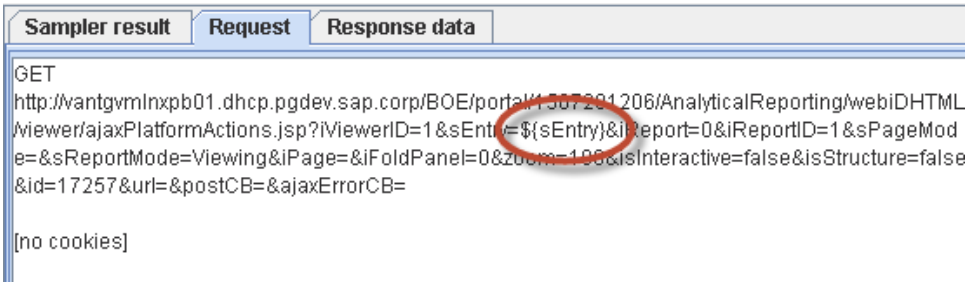


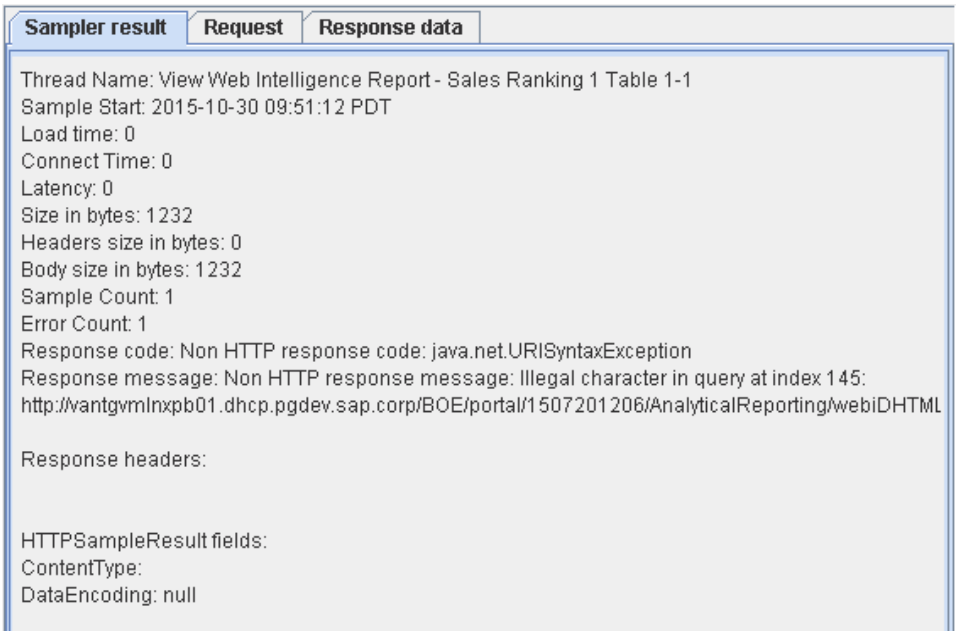
4. Select **View Results Tree** component and then click on the **Start** icon to run the test.



5. If an error occurs in the test, it appears red in the **View Results Tree** component.

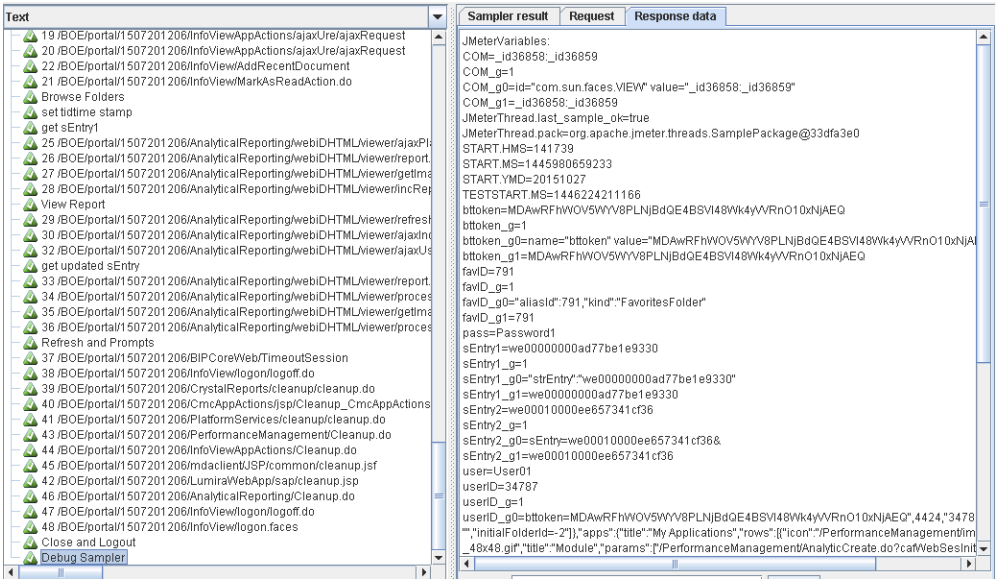






Warning: Here the syntax should be ${sEntry1} instead of ${sEntry}. You can fix this error on ajaxPlatformActions.jsp sampler.

6. Once the test plan runs without generating any exceptions, you can delete the **View Results Tree** and **Debug Sampler** components. You can now run additional tests under higher load.



Determining Benchmarks

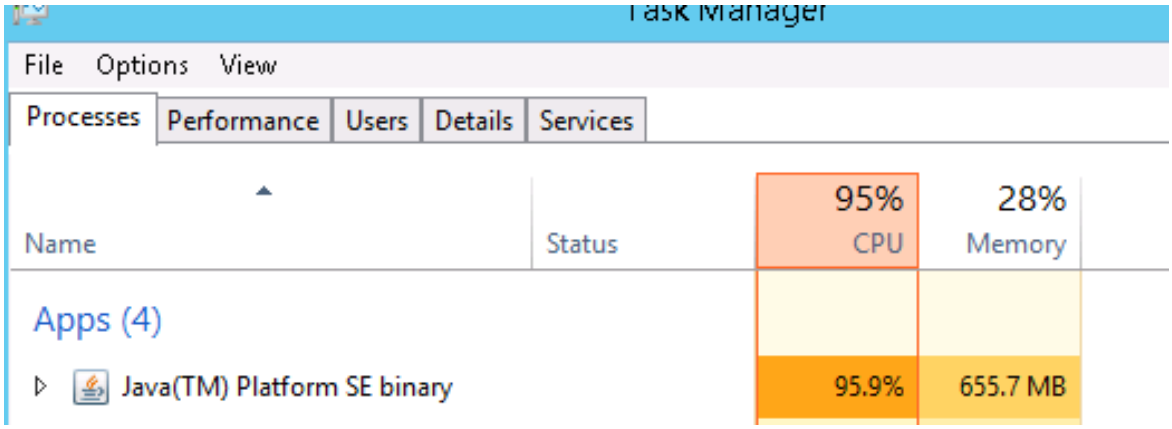
There are a few key points to consider while approaching a benchmark or performance test. The results and outcomes of generating load on your SAP BI platform environment are as good as the inputs.

- Performance testing machine resource consumption
- Ramp-up Period
- Repeating Test
- Timers

Performance testing machine resource consumption

We have discussed sizing of the SAP BI platform deployment but there is one more important aspect, which is about the server that is used to generate load. With hundreds of active concurrent requests, it is easy to overwhelm a small server running JMeter, and it is critical that we do not forget about this component. In early tests, we experienced test failures due to insufficient Java heap size. The steps on resolving that are included in the Web Intelligence test chapters below, but it is worth looking at how this manifests in our test executions.

The 95% CPU highlighted in the screenshot is due to the java process spending all of its time in garbage collection, due to the insufficient java heap size defined by default. This spike in CPU causes delays in the test execution, and ultimately results in hung threads throughout the system.



In addition to CPU and memory constraints, you should be aware of the network I/O that can be generated by hundreds of simultaneous requests. A network card can be quickly saturated, contributing to skewed numbers in the test. JMeter supports remote testing, in which commodity hardware can be used to distribute a test plan across several nodes. Complete instructions for configuring this process can be found in the Apache JMeter user manual. The link is as follows:

http://jmeter.apache.org/usermanual/remote-test.html

Ramp-up Period

The ramp-up period defines a period (in seconds) before all the users in a performance test are active in the system. This staggers requests and ensures that you do not take the environment from no utilization to 100% utilization instantly.

Flooding the system all at once is avoided for two reasons:

1. It does not simulate real usage. Performance testing is all about simulating legitimate business usage, and the chance of all users hitting the system simultaneously with the same request is impractical.
2. Process level thresholds are all exceeded at once, making it difficult to identify where a breakdown or bottleneck is occurring.

In this pattern, we use a relatively consistent ramp-up time between 120 seconds and 240 seconds. Consider increasing the ramp-up time as the number of users being tested increases. JMeter provides a setting at the root of every thread group that allows you to control ramp-up period.

Repeating Test

In each of the tests we execute for this pattern, we repeat, or loop through the test plan multiple times. This helps us create a sustained load on the system without exceeding the total number of concurrent users we target. In the example above, there would be at most 120 users on the system at any given time, based on the Number of Threads (users) property in the thread group. However, each of these 120 users would repeat the test 5 times, making a total of 600 transactions (120 X 5).

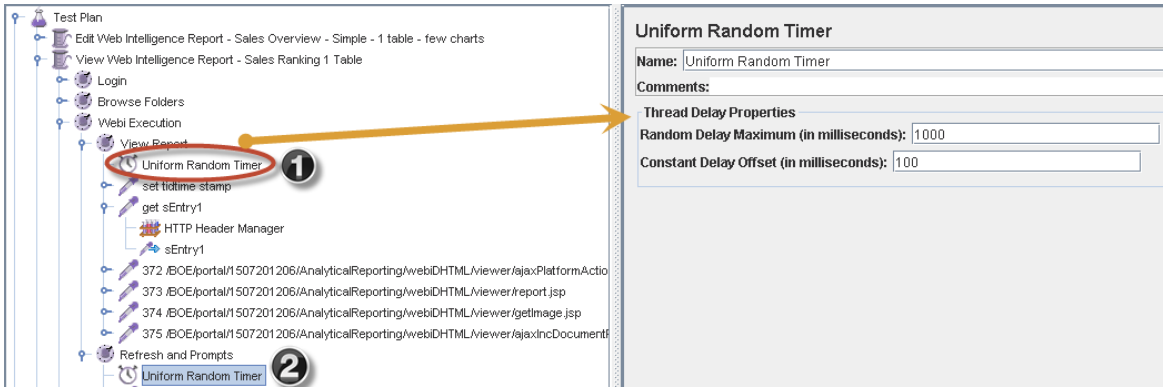
Timers

Another important aspect of performance testing is to implement “think time” between requests. Think time represents the amount of time a user is looking at information on the screen, or pausing to locate a folder item. Think time is important as we want to simulate real business usage. A user would not open a report and instantly close it without looking at something in the report.

There are a variety of timer components available in JMeter that allows you to incorporate think time in your test plans. For this pattern, we chose to use the **Uniform Random Timer** as it provides a simple mechanism for pausing between each request. Based on the settings below, with 1000 ms random delay maximum, and a constant delay offset of 100

ms, there will be a minimum of 100 ms pause and a maximum of 1100 ms pause between each request.

Figure: Uniform Random Timer



Information: Timers only affect samplers within the same transaction, so it is necessary to add them to each logic controller.

Data Sources

Almost every pattern will have a unique combination of data sources that are being used throughout the environment. For our Web Intelligence documents, we chose to focus on SAP HANA as it was readily available, had data we were familiar with and is a key component of SAP's strategy going forward.

Another reason we used HANA was to eliminate the database response time from our benchmarking. Database response time is a key component to tweaking and optimizing your report performance but for the purpose of this guide, we wanted to eliminate this factor for our initial tests and focus mainly on the BI components.

- Pattern source data details
- Semantic Layers
- Data Model

Pattern source data details

Details of the pattern data source are as follows:

Database Vendor	SAP
Database Name	SAP HANA Database 1.0
DB Server Version	1.00.101.00.1435831484 (HANA Rev 101)
DB Client Version	1.00.73 Build 0389160-1510
Network Layer	JDBC
DB Connection Type	Relational and OLAP connections

Semantic Layers

We create multiple Universes (UNX) in the Information Design Tool (IDT) for our Web Intelligence reporting needs. These universes are provided in the table below:

UNX Name	Description	Row Count
Sales Overview	A HANA view that shows Net Sales by Time, Product, Company and Location dimensions	51,289
Sales Ranking	A HANA View that ranks the Sales (top 10) by Net Sales, Quantity and Orders	10
Sales YOY	A HANA View that compares data Year over Year for the sales figures	40,899

Data Model

SAP HANA Interactive Education (SHINE) was used for the schema and data and can be downloaded for use in your own HANA based tests. More information on the SHINE demo application can be found on SCN: https://blogs.saphana.com/2014/03/10/shine-sap-hana-interactive-education/

Calculation and Analytic views were leveraged from the openSAP course, BI Clients and Applications on SAP HANA. The delivery unit containing these assets is provided as part of the pattern.

Web Intelligence Performance Testing

Web Intelligence is a very popular choice for Analytics in a BI Platform Deployment. It is a mature product that has a rich history with most customers. It is also a familiar tool that is robust enough to cover a wide spectrum of reporting and analysis needs. For these reasons, it is the most heavily utilized tool in the suite.

That being said, it is important that the right amount of focus be placed on performance testing and benchmarking Web Intelligence documents and workflows. A lot of the testing we did to validate our pattern was Webi focused.

In the following sections, we will give an overview of the Web Intelligence Documents that we used along with the workflow examples that we used in our Test Plans.

- Web Intelligence Documents
- Workflow Examples

Web Intelligence Documents

Below is a list of Web Intelligence documents that we used in this pattern with a brief description of the document:

Document Name	Description	Row Range
Sales Overview – Simple 1 Table with Charts	A Sales Overview report that lists out the detailed sales of products by year, region, product and time. This document has a table of details and a summary report tab that outlines the query summary information.	Based on parameters, it returns around 8,000 – 13,000 rows.
Sales Ranking – Simple 1 Table	A Sales Ranking report that shows the sales ranking based on net sales for the chosen year.	Depending on parameters, 1,000 to 12,000 rows.

Workflow Examples

There are two primary workflows in Web Intelligence documents, a simple view on-demand workflow, and a more complicated Power User workflow where the document is changed slightly in the process. They are explained below.

Workflow 1 – View Web Intelligence Document "Sales Ranking - 1 Table" on demand

This workflow is a simple Consumer end user test. We use Fiddler and the JMeter components to turn this workflow into a functioning test plan for our benchmarking.

The following are high-level steps for this test plan, which involves opening, refreshing and closing the document.

1. Launch browser (Internet Explorer) and go to the Load Balancer URL for BI Launch Pad.
2. Login as a User who has access to view the document.
3. Navigate to the folder that houses the document in BI Launch Pad.
4. Double-click on the **Sales Ranking - 1 Table** Document to open it in the HTML Interface.
5. Once loaded, click the **Refresh** icon on the tool bar.
6. Enter the Prompt Values and click **Ok** to run the Document.
7. Once the page loads, click on the **Summary Report Tab**.
8. On the Report viewer tab, choose the **Close** option to close the document.
9. Log off BI Launchpad.
10. Close Browser.

Workflow 2 – Edit and Refresh Web Intelligence Document "Sales Overview – Simple – 1 Table with Charts"

This workflow simulates a Power User to open a Web Intelligence document and make some minor changes before refreshing the data. We use Fiddler and JMeter to turn this workflow into a dynamic test plan.

Following are the high-level steps for this test plan:

1. Launch browser (Internet Explorer) and go to the Load Balancer URL for BI Launch Pad.
2. Navigate to the folder that houses the document in BI Launch Pad.
3. Right click on the **Sales Overview – Simple – 1 table few charts** document and choose **View**.
4. Click the **Design** button in the top-right corner of the tool bar.
5. Click the **Data Access** Tab and choose the **Edit** button.
6. Remove Quarter, City and Month from the Result objects.
7. Add **Net amount** to the Query Filter using **Greater Than**= 1000.
8. Run the query.
9. Refresh values and change year value in the prompt dialog to **2015**.
10. Click **OK** to run the report.
11. Click on **Summary Report Tab**.
12. Close **Sales Overview…** report.
13. Log off BI Launch Pad.
14. Close browser.

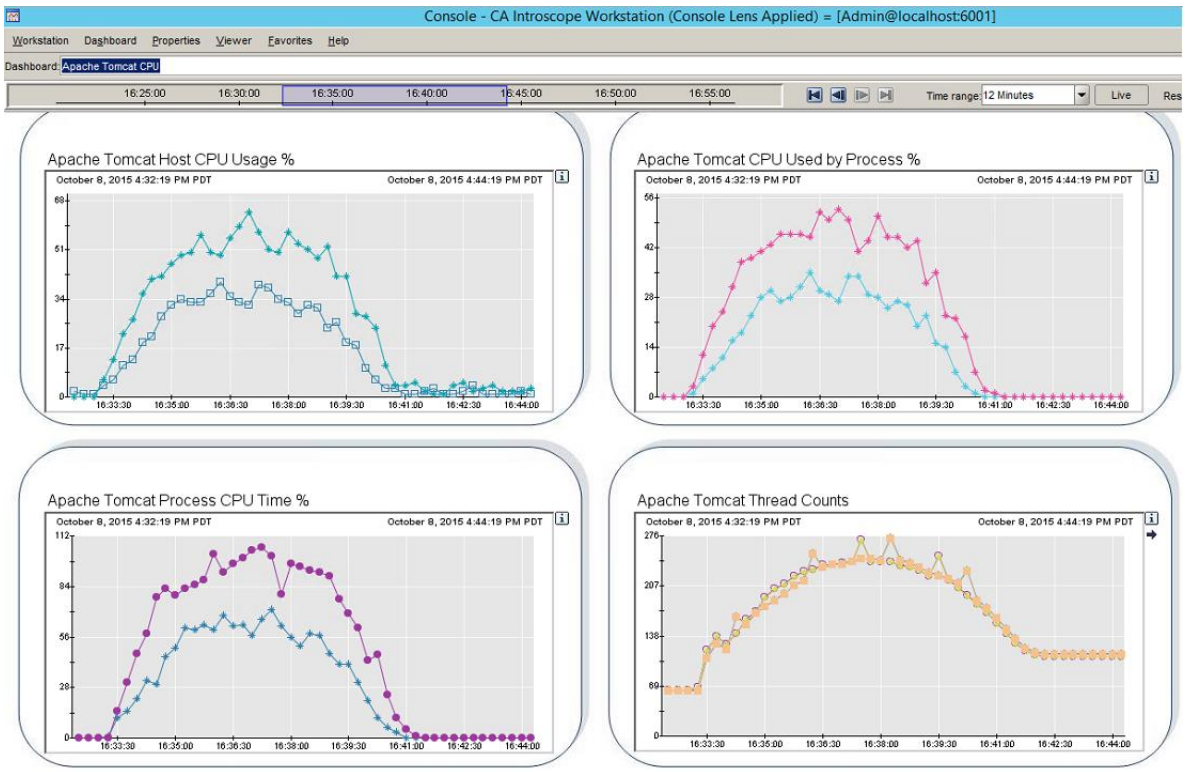
Web Intelligence Test Plan and Results

We ran several tests after deploying our own internal pattern. We wanted to document these as examples of how you can utilize this pattern to analyze your own system. Test 1 is the most comprehensive. The other tests all include some insight into what the JMeter and CA Introscope tools provided us during our analysis of each test.

- Test 1- Out of the box baseline and example
- Test 2- 50 Active Concurrent Users
- Test 3- 150 Active Concurrent Users
- Test 4- 150 Active Concurrent Users
- Test 5- 150 Active Concurrent Users
- Test 6- 100 Active Concurrent Users
- Test 7 to10 - Investigating Bottleneck Root Cause
- Test 13- 300 Active Concurrent Users

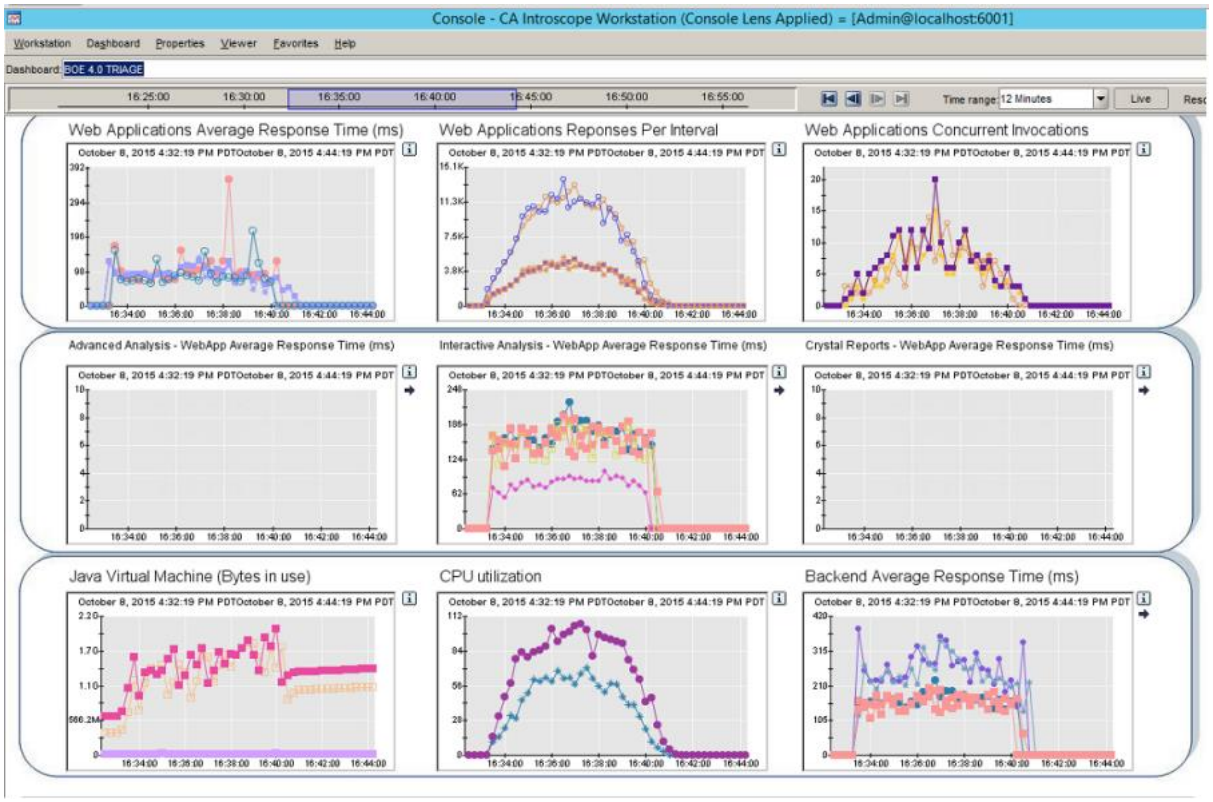
Resources for Test 9

Tomcat CPU Usage – 150 ACU Tweaked

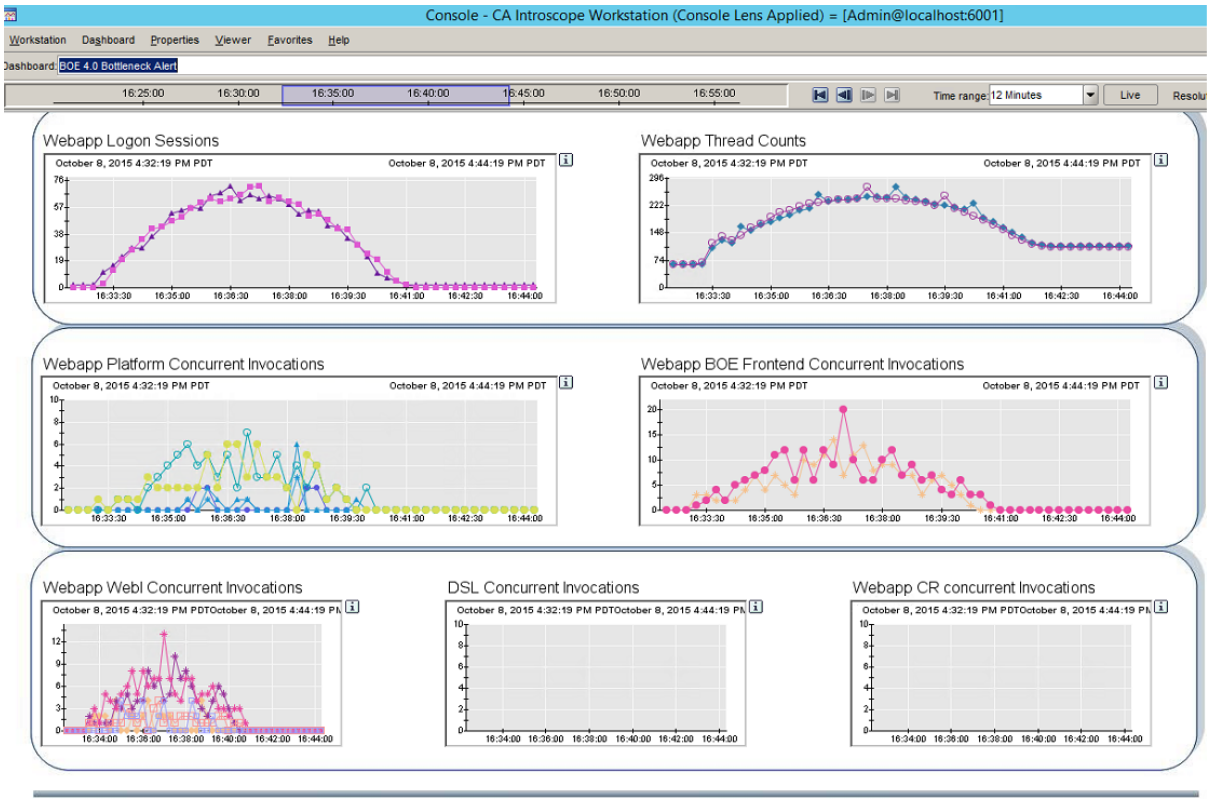


Platform Averages – 150 ACU Tweaked

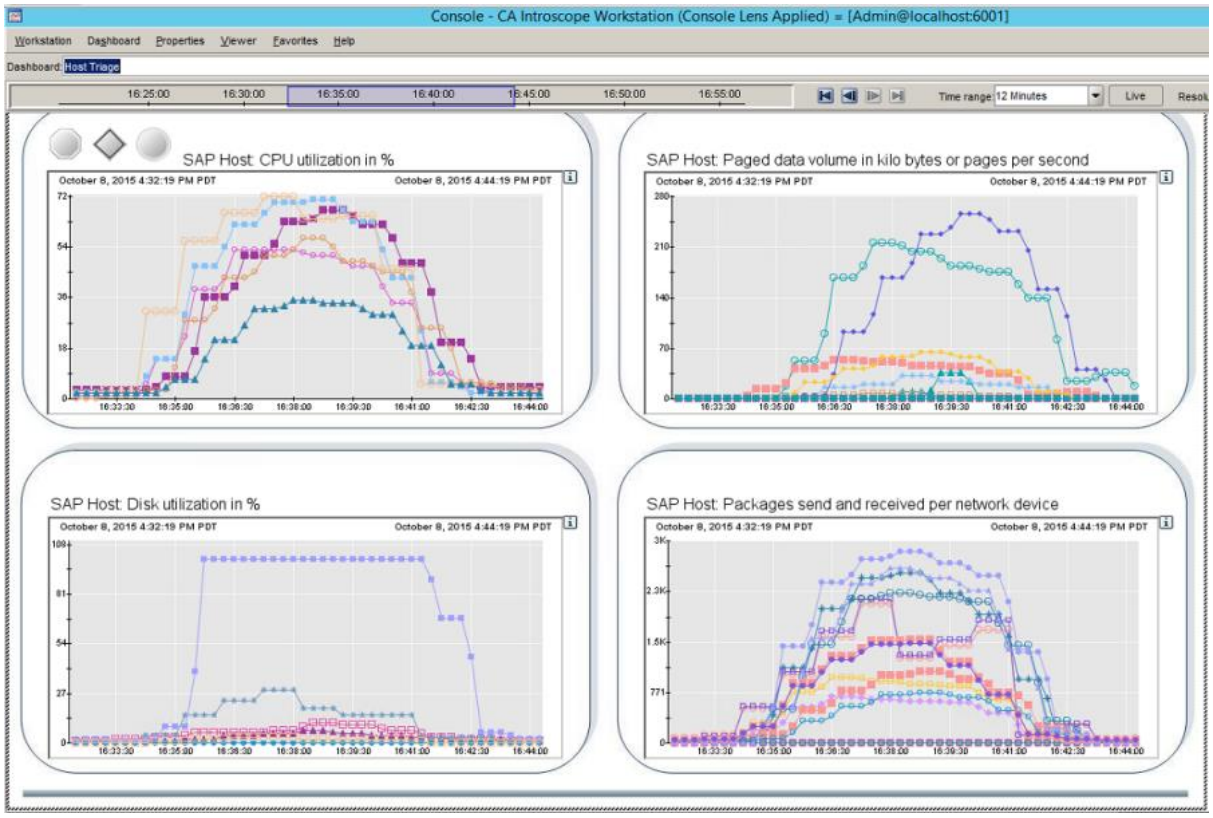
BI Triage – 150 ACU Tweaked-0003



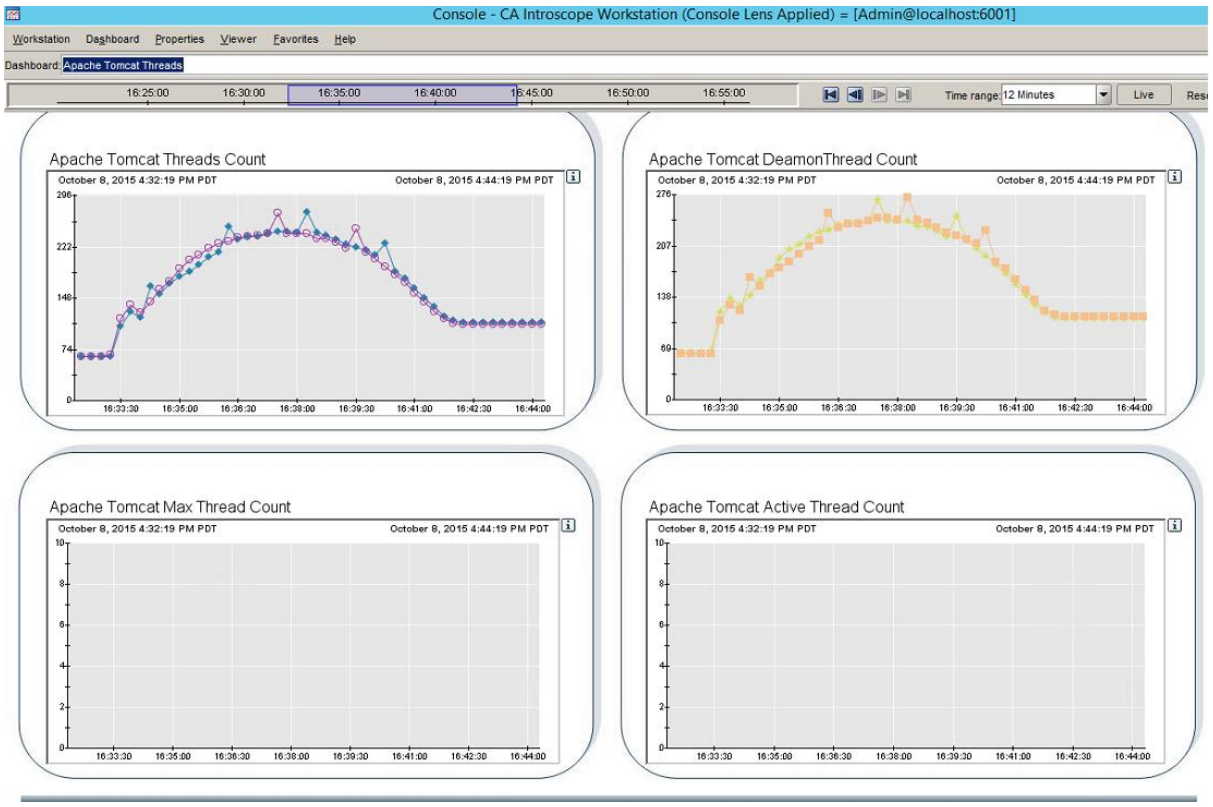
Bottleneck Alert – 150 ACU Tweaked



Host Triage – 150 ACU Tweaked-0002



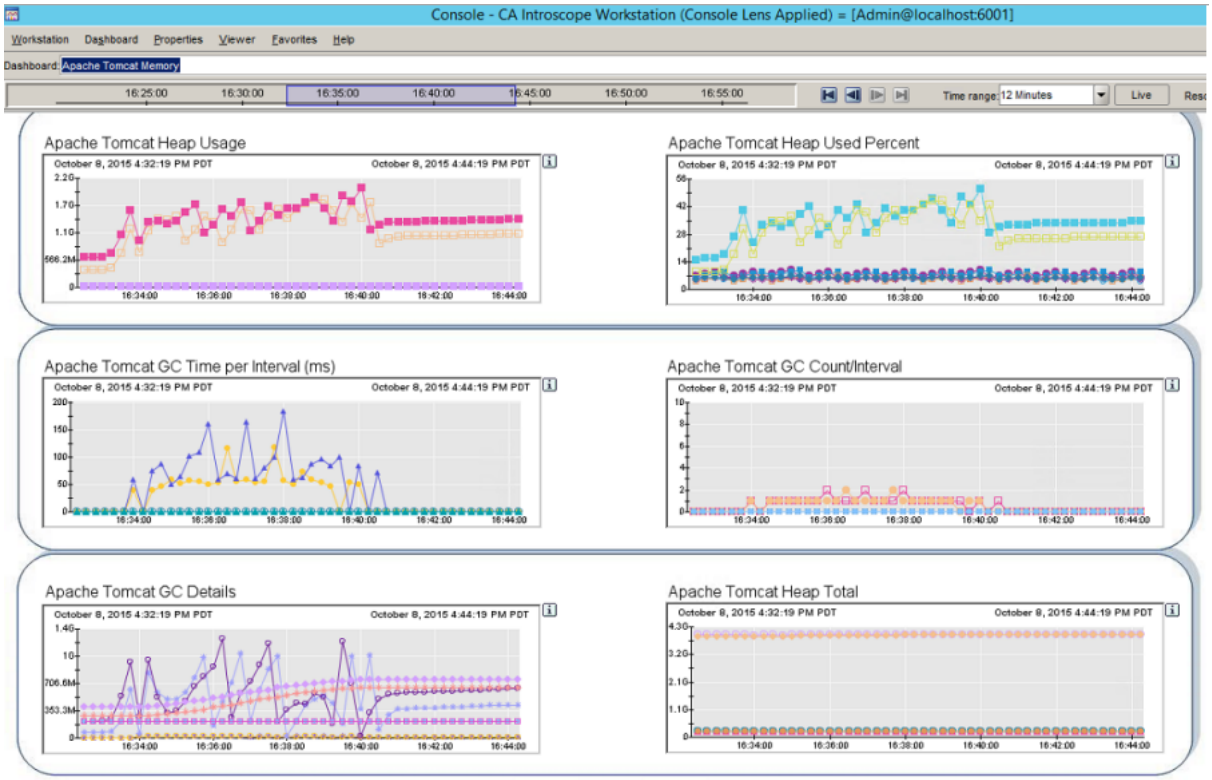
Tomcat Threads – 150 ACU Tweaked



Tomcat Strut – 150 ACU Tweaked

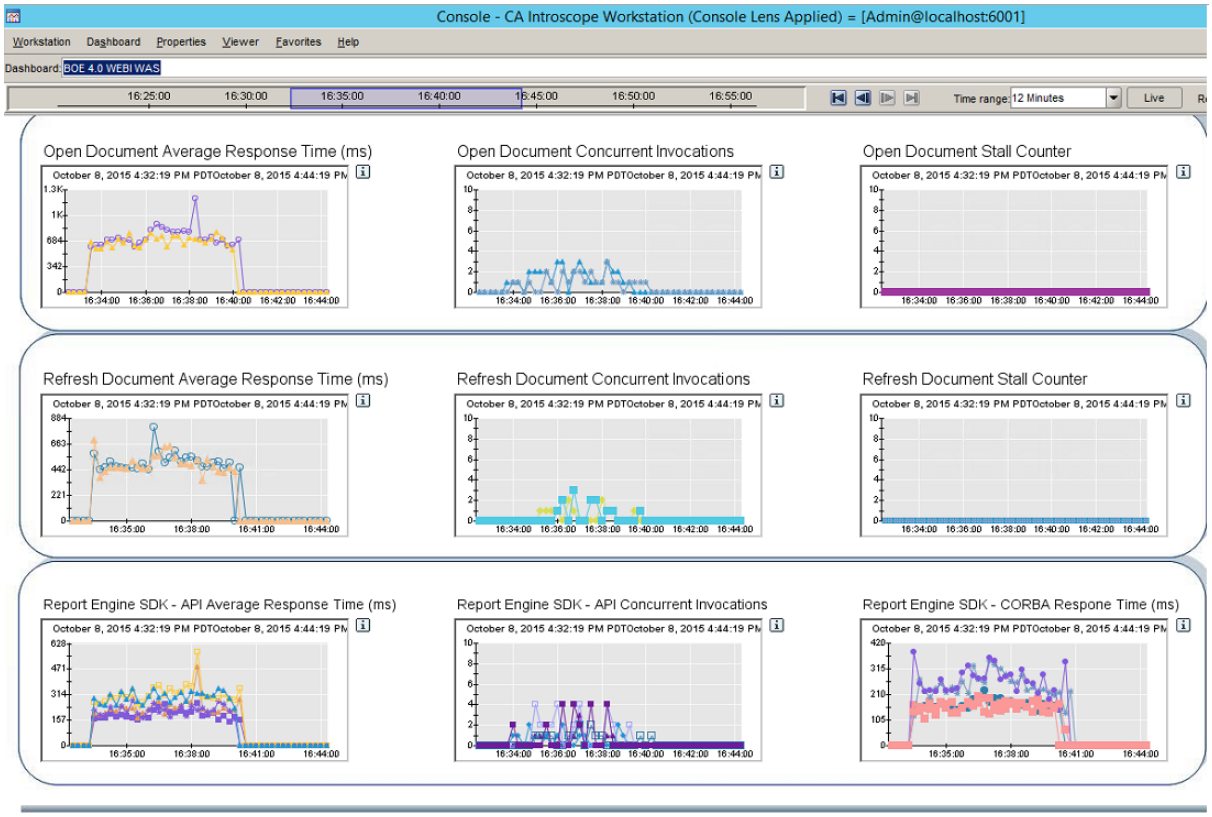
Tomcat GC Details – 150 ACU Tweaked

Tomcat Memory – 150 ACU Tweaked-0000

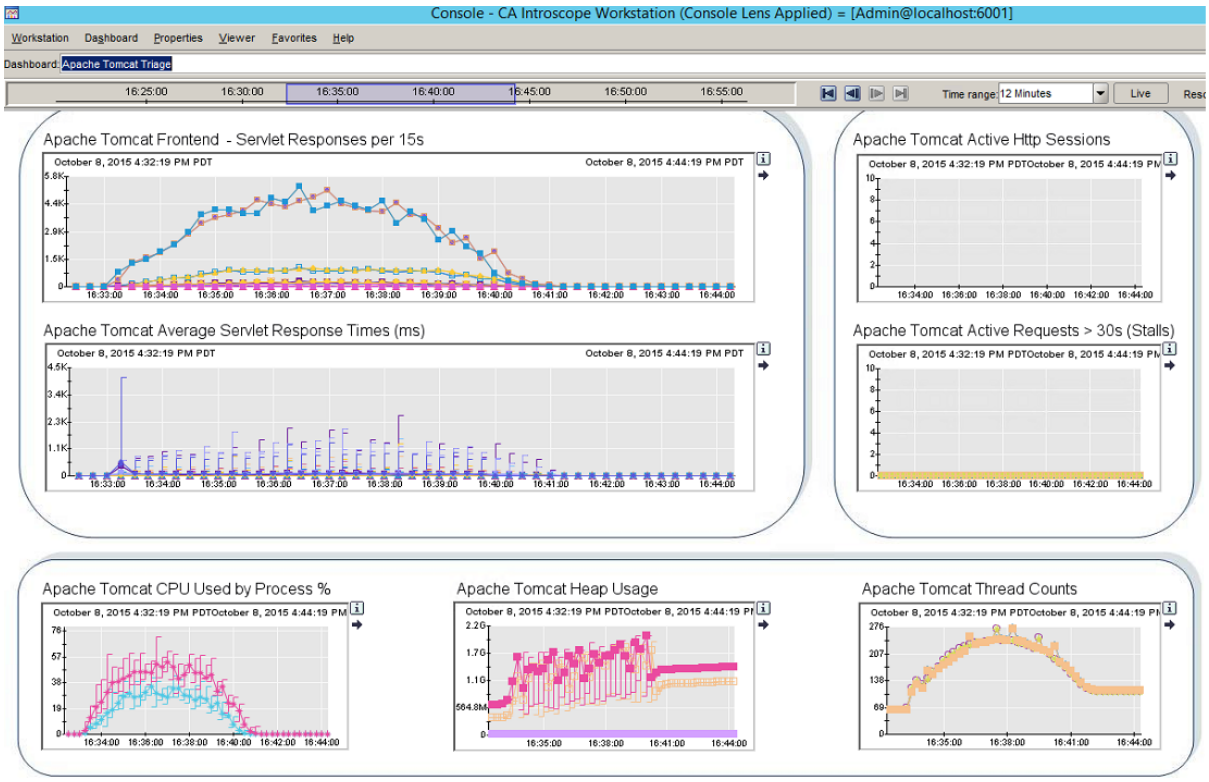


Tomcat Servlets - 150 ACU Tweaked

WebI WAS – 150 ACU Tweaked



Tomcat Triage – 150 ACU Tweaked



WebI Server Details – 150 ACU Tweaked-0001

WebI Server Overview – 150 ACU Tweaked

There are no screenshots and images for Test 10.

- Test 11- 150 Active Concurrent Users - After ESX and Load Balancer Changes
- Test 12- Screenshots and Data
- Test 13- 300 Active Concurrent Users

Test 1- Out of the box baseline and example

The first test we wanted to complete was a basic "Out of the Box" installation. We had broken out the different tiers according to our architecture diagram but did not tweak anything else other than the basic server distribution across multiple machines. This test will serve as our baseline for benchmarking, load testing and optimizing our servers.

The results of this test will serve as an example of the different metrics we can analyze when running this pattern.

Details for Test 1 are shown in the table below.

Test 1	**5 Active Concurrent User (ACU) Edit / 25 Transactions / 60 Second Ramp-up 10 ACU View / 50 Transactions / 120 Second Ramp-up**
Active Concurrent Users	15
Total Transactions	75
Start Time	2:45 PM
End Time	2:52 PM
Duration	~7 Minutes

JMeter Output

Edit Workflow:

Summary Report

Name: Edit Web Intelligence Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	25	746	532	2206	310.08	0.00%	4.7/min	32.78	428958.2
Browse Folders	25	1659	1277	1964	170.29	0.00%	4.6/min	29.48	390355.8
View Report	25	2258	1419	7811	1177.88	0.00%	4.6/min	71.22	952056.0
Edit and Refresh	25	4402	3033	5402	634.45	0.00%	4.5/min	40.05	552430.9
Close and Logout	25	208	100	419	85.73	0.00%	4.8/min	2.40	30909.3
TOTAL	125	1855	100	7811	1584.53	0.00%	19.6/min	150.45	470942.0

For the Edit workflow, we see the following values:

Edit Web Intelligence	Number of Samples	Average	Min	Max	Std Dev
Login	25	746	532	2206	310.08
Browse Folders	25	1659	1277	1964	170.29
View Report	25	2258	1419	7811	1177.88
Edit and Refresh	25	4402	3033	5402	634.45
Close and Logout	25	208	100	419	85.73
Total Transaction	**25**	**9273**	**6361**	**17802**	**2378.43**
TOTAL	125	1855	100	7811	1584.53

We obtain **Total Transaction** values above by adding the 5 action values together for each column (Average, Min, Max and Standard Deviation). The average transaction values for benchmarking signifies the average time the transaction has to run as well as the best & worst-case scenarios we should expect.

Interpretation of the above data

The above information gives a benchmark that we can use as a baseline. The JMeter Summary Report gives the minimum, maximum, standard deviation and average times for each action. It also gives an idea of the throughput. We use these statistics to validate test patterns, give us benchmarks to measure against, and to evaluate the sizing of our deployment.

Total Time (Avg) gives the approximate time taken for the total transaction. It is **9.273 seconds** in the first run, for the Edit transaction. You can see from the JMeter screenshot that this is not calculated by default. We obtain this value by adding together the 5 averages for each action (746 + 1659 + 2258 + 4402 + 208) which is 9273 milliseconds or 9.273 seconds total average time.

We calculate the potential maximum transaction time by adding up the 5 maximum values for each action (2206 + 1964 + 7811 + 5402 + 419) which is **17.802** seconds, as a potential maximum value. This is the worst case scenario value which signifies the longest duration an edit request would take if every action took the maximum amount of time in the test. However, it is unlikely to see the maximum action time occur for every step of a single transaction, thus being a worst-case scenario. This is the same for the minimum (Min), or fastest transaction time. You can obtain this value by adding together the 5 action times in the Min column. This gives us a total of **6.361** seconds. This is theoretically the fastest that this workflow performs. Like with the Maximum value, this is unlikely to occur often, as all actions would need to run at their optimal runtime.

Therefore, the most valuable number is the Average time that each Transaction takes. We use this value as benchmark to evaluate findings on future tests.

View Workflow:

Summary Report

Name: View and Refresh Webi Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	50	682	537	1337	115.33	0.00%	11.2/min	78.38	429037.3
Browse Folders	50	1613	1320	2015	182.06	0.00%	11.1/min	70.80	390404.6
View Report	50	1301	900	6676	825.18	0.00%	11.4/min	44.65	239966.8
Refresh and Prompts	50	2136	1580	5624	608.65	0.00%	11.6/min	48.20	255691.5
Close and Logout	50	225	115	444	70.50	0.00%	11.7/min	6.31	33145.6
TOTAL	250	1191	115	6676	821.83	0.00%	50.9/min	223.47	269649.1

For View workflow, the values are as follows:

View Web Intelligence	Number of Samples	Average	Min	Max	Std Dev
Login	50	682	537	1337	115.33
Browse Folders	50	1613	1320	2015	182.06
View Report	50	1301	900	6676	825.18
Edit and Refresh	50	2136	1580	5624	608.65
Close and Logout	50	225	115	444	70.5
Total Transaction	50	**5957**	**4452**	**16096**	**1801.72**
TOTAL	250	1191	115	6676	821.83

Interpretation of the above data

As with Edit workflow we see, Average, Minimum and Maximum values for the View Workflow. In this test, twice as many transactions as the Edit workflow are run to simulate an average workload on the system. Since most users have only rights to view and refresh a document, 50 View and 25 Edit transactions are present in the test plan.

The Average Total Transaction time is **5.957 Seconds** for the View workflow. The Minimum time being **4.452 Seconds** and the Maximum Time being **16.096 seconds.**

CA Introscope Output:

We install and configure CA Introscope as a part of this deployment. This is done to get additional data on the back-end processes and servers involved in this benchmarking

pattern. We highly recommend you to install and configure CA Introscope if time and resources allow. Below you will see first-hand the value that CA Introscope brings to the testing cycle.

The output of CA Introscope is slightly different compared to JMeter output as CA Introscope captures the entire test run. It does not separate the view vs edit workflows run concurrently. What you see in these results is the overall system performance and metrics during the test run.

For each test run, data for the following 14 dashboards in CA Introscope are collected. To download these screenshots, follow the below link:

Download Resources for Test 1

- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

Below are screenshots from the 14 different dashboards for the first test we did. We can use these dashboards to compare the performance between test plan runs and to help us identify bottlenecks and other issues within our environment. We have shown ALL screenshots that we captured for Test 1. This will be our main reference point and we wanted to show how the system performed without any major modifications. For additional tests, we will only be showing CA Introscope output that differs between test runs. This will help us showcase the value that CA Introscope brings to the table.

Tomcat Servers

Figure: Apache Tomcat CPU Dashboard

Observations for the above figure:

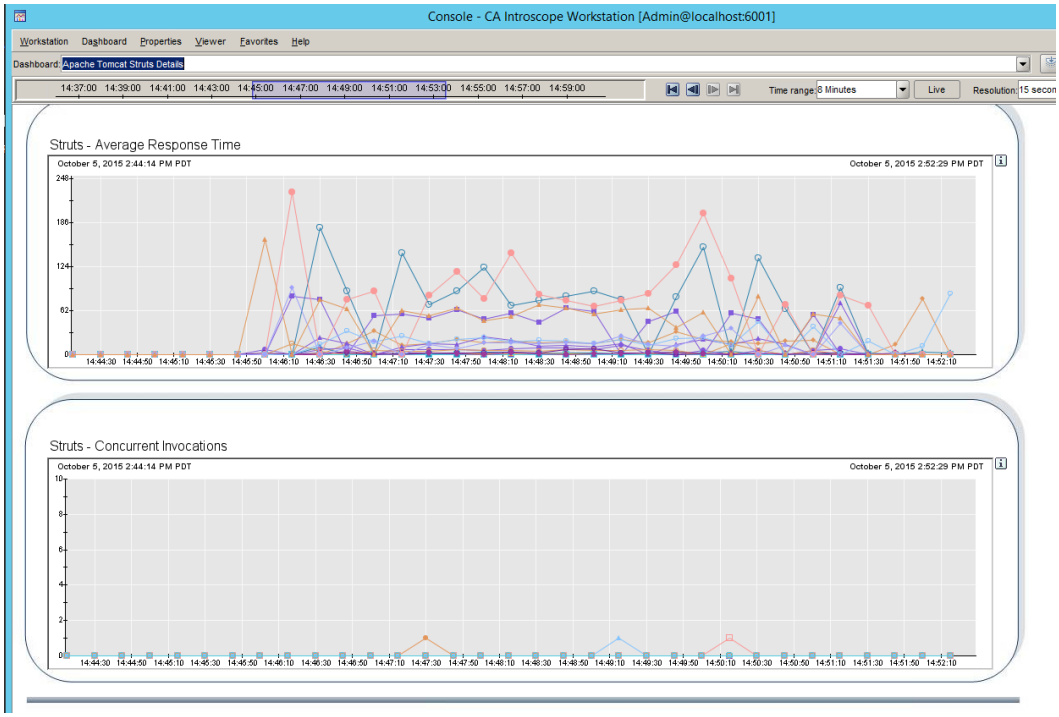
1. CPU utilization is not uniform between our two Tomcat servers. There is about 10% difference between the two servers.
2. Tomcat Thread Counts are very close for the two servers, therefore the load is well distributed.

Figure: Apache Tomcat GC (Garbage Collection) Details Dashboard

Observations for the above figure:

1. Both our Tomcat Servers show similar response for the garbage collector details.

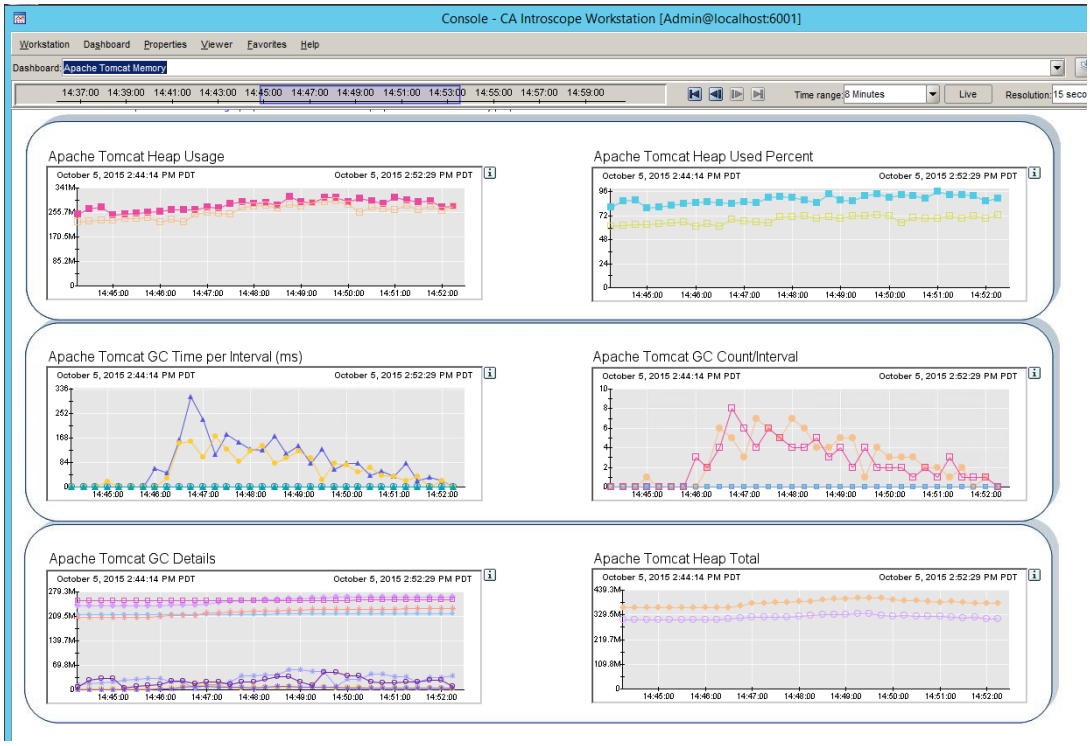
Figure: Apache Tomcat Struts Details Dashboard



Observations for the above figure:

1. We are most interested in the Average Response time of the Struts within this dashboard. We should see this scale up as we increase the load, and can compare response time in future tests to see if it scales accordingly.

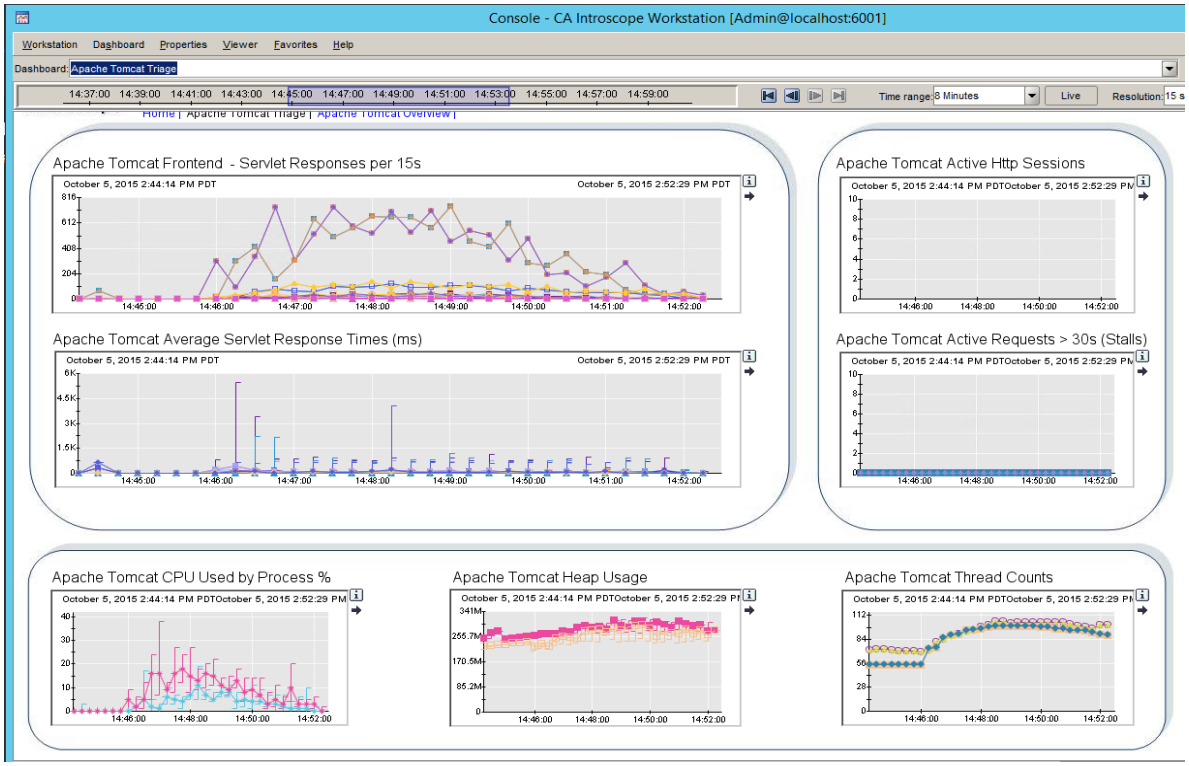
Figure: Apache Tomcat Memory Dashboard



Observations for the above figure:

1. Heap Usage is consistently around 250-300mb and increases gradually throughout the test. This gives us an idea of how much heap is needed to handle a number of concurrent users.

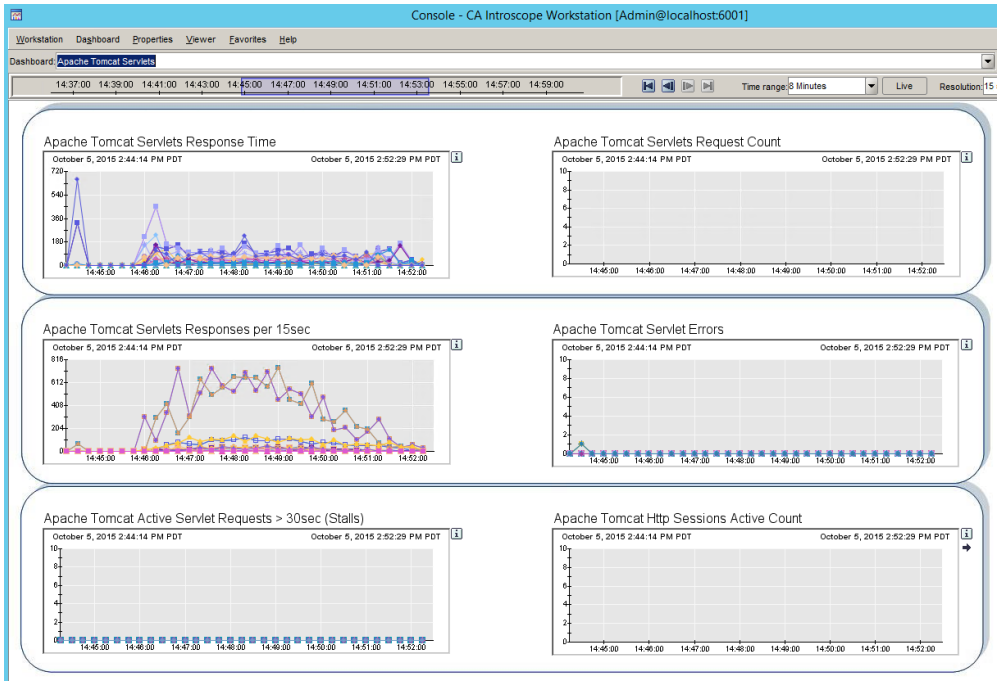
Figure: Apache Tomcat Triage Dashboard



Observations for the above figure:

1. The Servlet Responses per 15s increases with test progress. We expect to see this as we gradually increase the load and concurrent requests during the testing workflow. At its peak, the servers were handling about 500-800 responses each per 15s interval.
2. The Average Servlet Response Time is higher at the beginning of the test but for most time is below the 1000ms (1 second) mark. There is a spike in average response time (~4 seconds) close to the peak of the test but only for a short period before returning to < 1 second averages.

Figure: Apache Tomcat Servlets Dashboard



Observations for the above figure:

We see a small spike in Servlet Response Time at the beginning of the test. This is because of caching and connections that are established. This is expected and we can see that it stabilized after a few seconds and ran consistently faster.

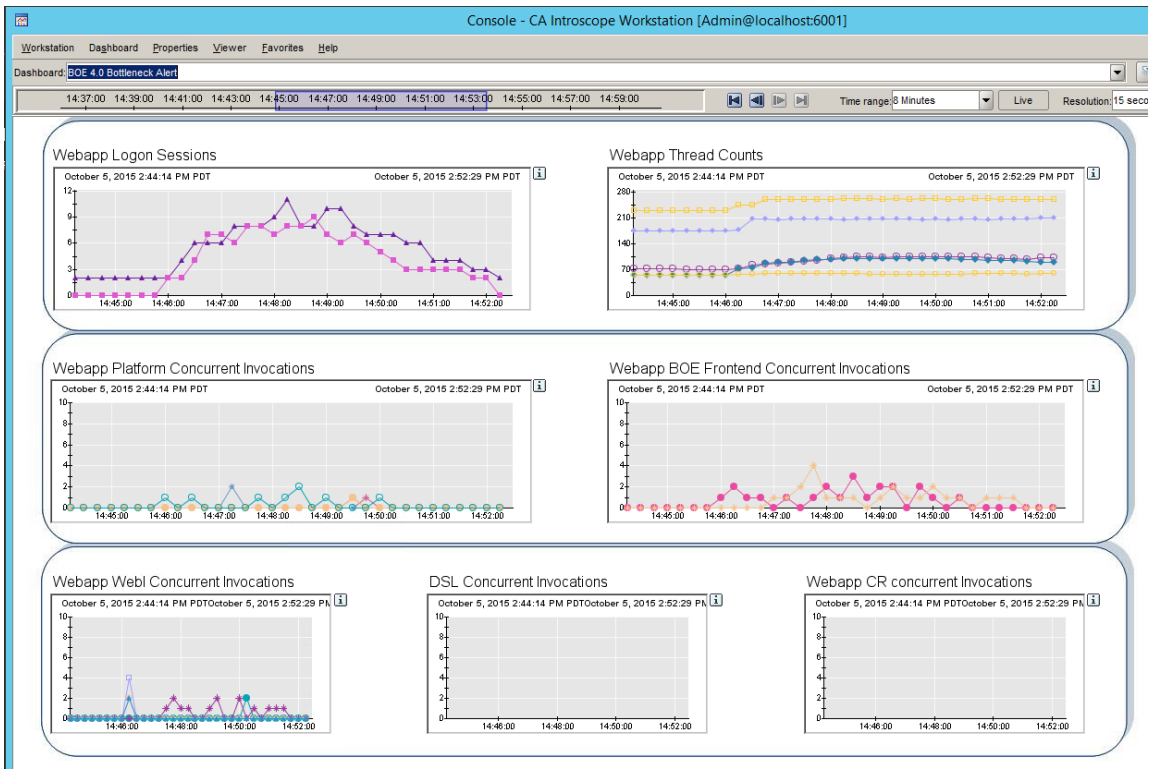
Figure: Apache Tomcat Threads Dashboard

Observations for the above figure:

1. Thread Count is distributed between both servers and peaks at about 110 threads. This gives us an idea of how we are sized and how close to the ceiling we are during our testing. Very helpful for sizing and determining maximum loads.

BI Platform Output

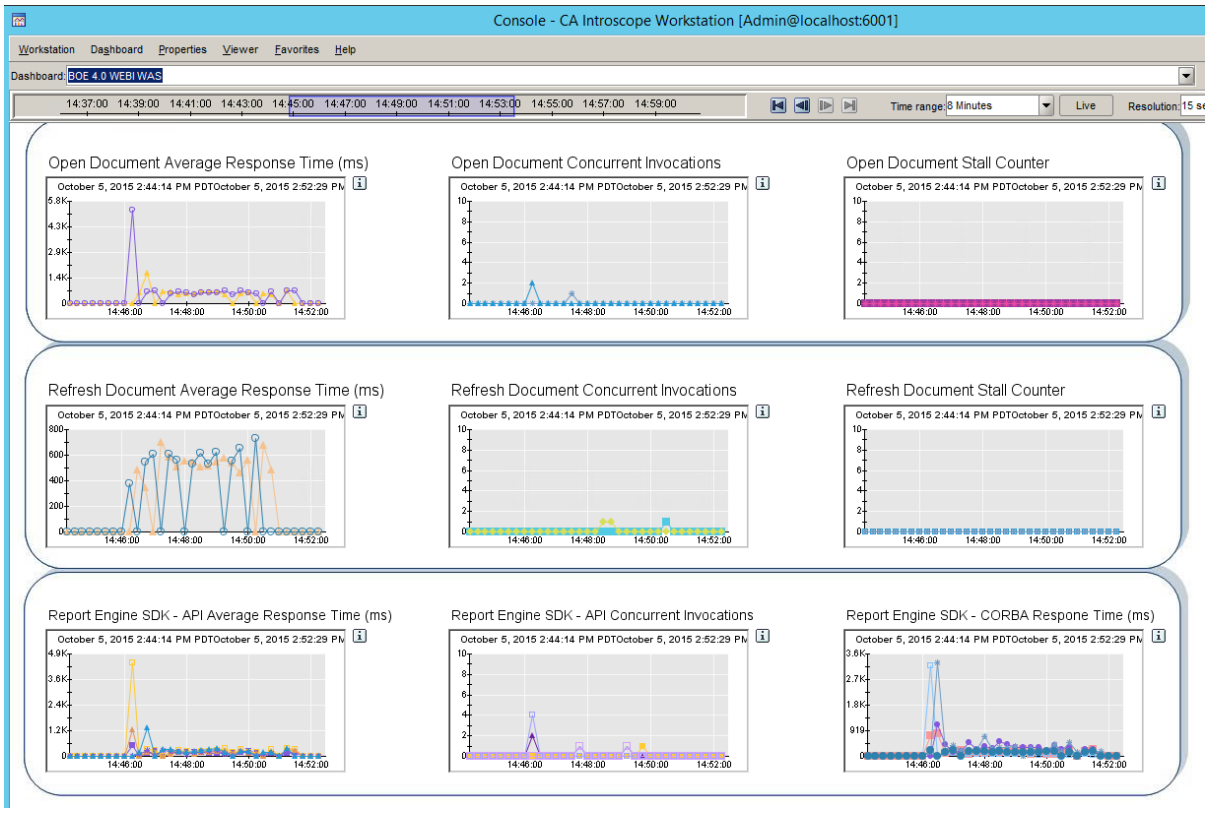
Figure: BusinessObjects Enterprise (BI Platform) 4.x Bottleneck Alert Dashboard



Observations for the above figure:

1. The Web Application Logon Sessions chart gives us an idea of how the sessions are load balanced between our two CMS machines. We can see that it is evenly distributed and peaks at around 10 sessions per server.

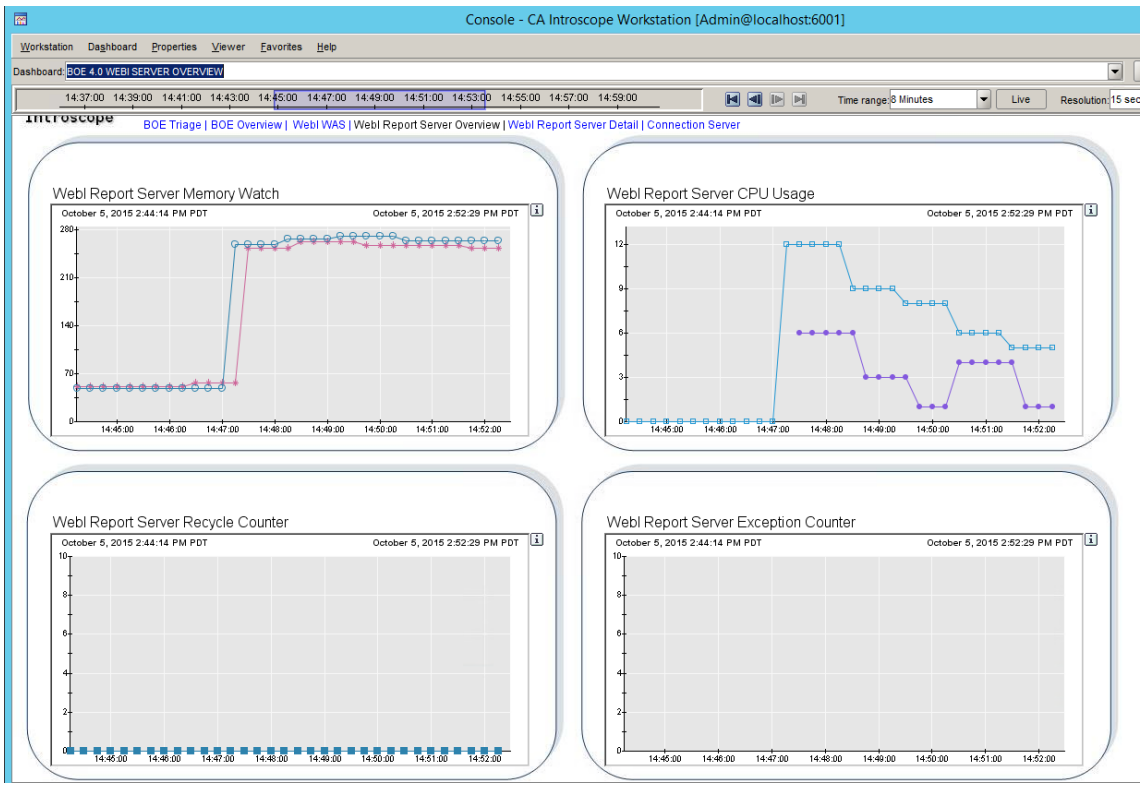
Figure: BusinessObjects Enterprise (BI Platform) 4.x Web Intelligence Web Application Server Dashboard



Observations for the above figure:

1. Open Document Average Response Time (ms) shows a spike at the beginning of the test but a steady average after that. This is due to caching and server connections being established. Initial delays are expected unless you have warmed up the environment before the test.
2. Refresh Document Average Reponses Time is fairly consistent and shows an average of around 600ms in our first test.
3. Report Engine SDK – API Average Response Time is consistent after an initial spike as expected.
4. Report Engine SDK – CORBA Reponses Time also shows a similar spike. This is a delay when servers initially create TCP/IP connections to each other. Once those connections are established, the response time is much better as we can validate in this graph.

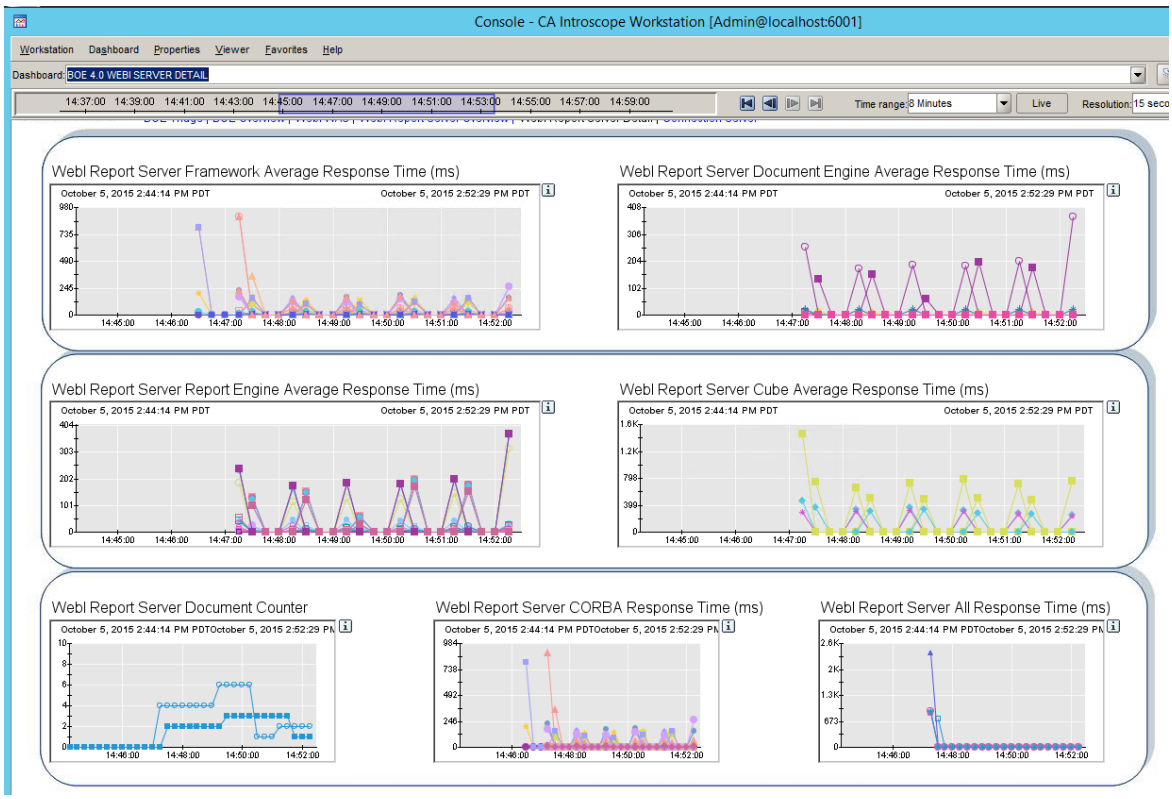
Figure: BusinessObjects Enterprise (BI Platform) 4.x Web intelligence Server Overview Dashboard



Observations for the above figure:

1. Webi Report Server Memory Watch graph shows a sharp spike at the beginning of the test, and steady consumption later. This is because the Web Intelligence Report Server loads libraries on demand, and this increases the memory consumption on first use.
2. Web Intelligence Report Server CPU Usage is below 12% on 1 server and 6% on the other, which results in a large gap. The future tests should be monitored to see if it evens out over time.

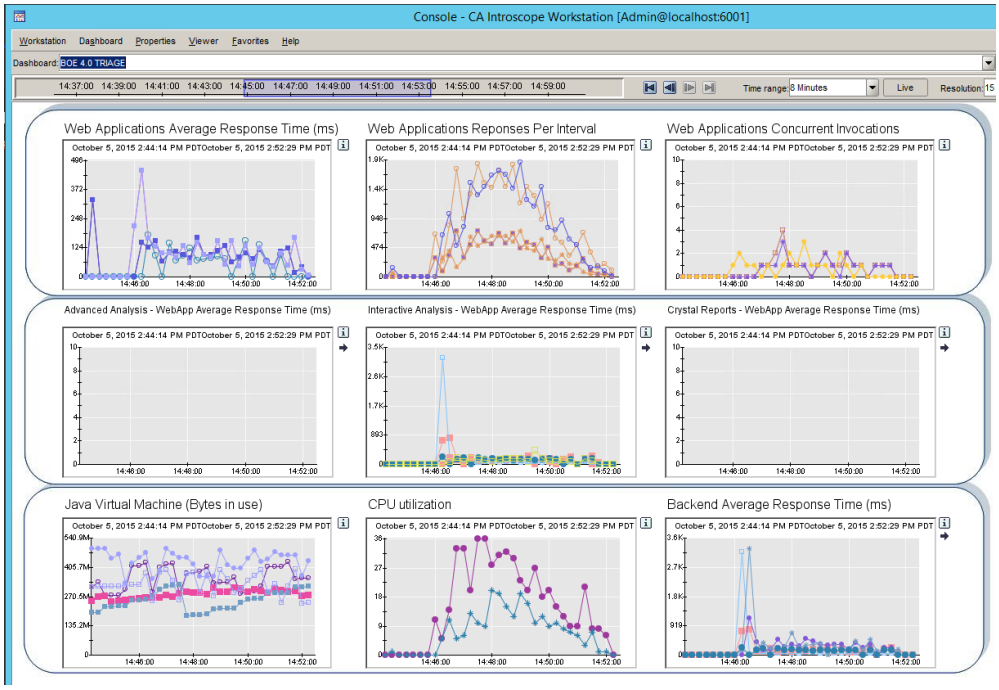
Figure: BusinessObjects Enterprise (BI Platform) 4.x Web Intelligence Server Detail Dashboard



Observations for the above figure:

1. Webi Report Server Framework Average Response Time has the initial spike as in the other graphs.
2. Webi Report Server Document Engine Average Response Time is consistent on the two Webi Report Servers. It was under 200ms on average for both servers.
3. The rest of the graphs look normal, with a few spikes at the beginning.

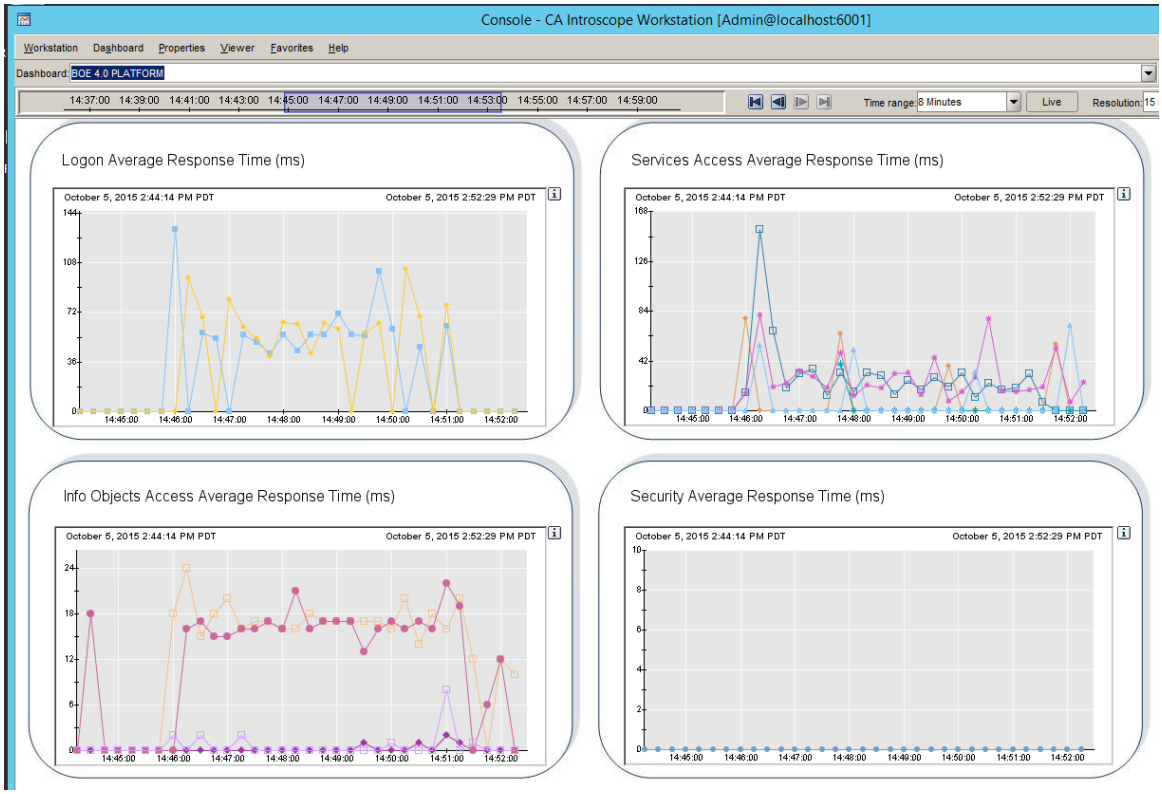
Figure: BusinessObjects Enterprise (BI Platform) 4.x Triage Dashboard



Observations for the above figure:

1. We can see spikes on a few Average Response Time graphs on this dashboard. For future tests, you should consider warming up the servers first to level these spikes to some extent.
2. CPU Utilization has a small spread on the dashboard. This is the measure of CPU on the Tomcat Servers only. We need to investigate whether this is a true CPU metric or not by logging into the box during a test and checking if values in the graph match the true machine readings.

Figure: BusinessObjects Enterprise (BI Platform) 4.x Platform Dashboard

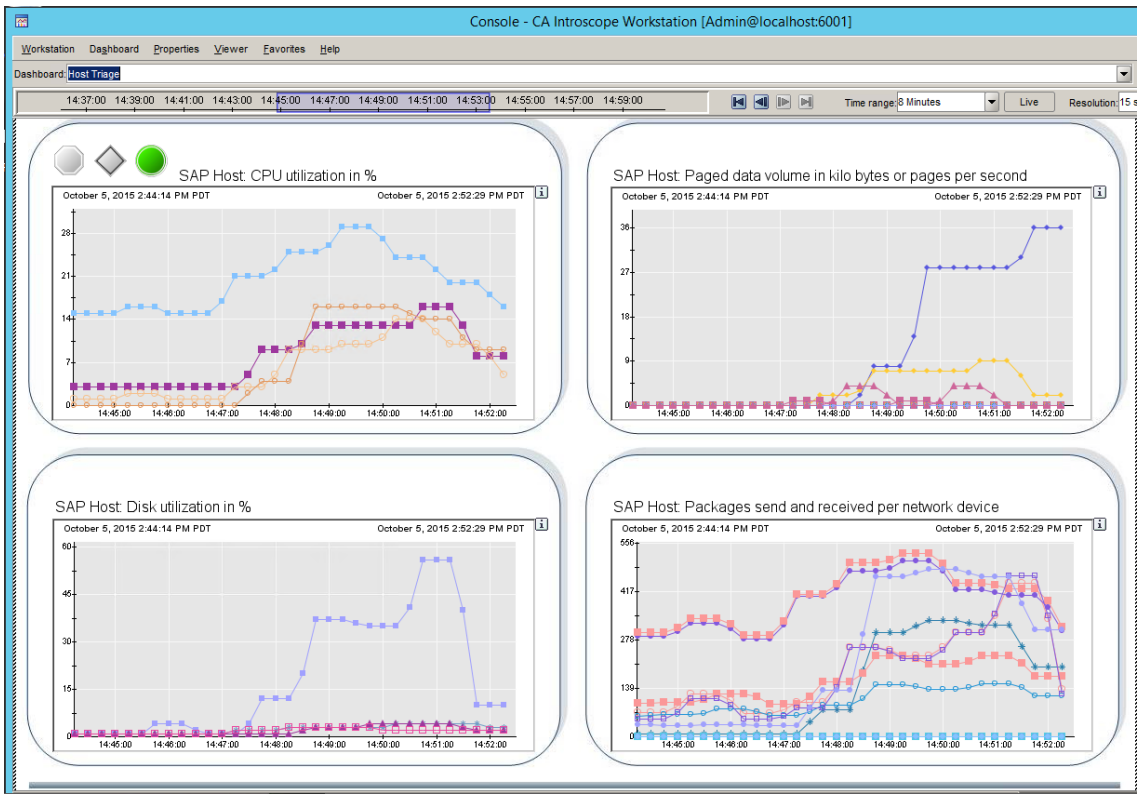


Observations for the above figure:

1. Logon Average Response Time (ART) is around 60ms during our test.
2. Service Access Average Response Time is around 40ms on average with a few higher averages throughout the test.
3. Information Objects Access ART is consistent and fast at about 18ms on average.

Hosts

Figure: Host Triage (BI and Application Server) Dashboard



Observations for the above figure:

1. SAP Host: The percentage of CPU utilization is even for three of the four servers and peaks at about 15%.One of the servers hits around 30% but this could be an issue with CA Introscope data and should be validated if it becomes an issue. It is important to double-check inconsistencies of this type because, CPU utilization does not always account for the correct number of CPUs.
2. SAP Host: Paged data volume in kilobytes or pages per second is consistent but shows one server spiking near the end of the test. This could indicate an issue with memory availability on that machine, so we should watch it in future tests and ensure it's not a sizing issue.
3. SAP Host: The percentage of disk utilization is very low for all machines except the machine with the Page data issue as mentioned above. We should investigate this to see if it has potential to be a bottleneck or could be a server issue.

Test 2- 50 Active Concurrent Users

This test will more than triple the amount of requests that we put through the system to help simulate a 50 Active Concurrent User load. Since our system is sized for more than 50 ACUs, we would expect to see similar results to the Test 1 (15 ACU) test. Before executing this test, we logged in and ran a few reports manually to "warm" the system up. This helps

eliminate huge swings in the Min and Max values that may be caused by initial connections being established.

Details of Test 2 are as follows:

Test 2	• **20 ACU Edit / 100 Transactions / 120 Second Ramp-up** • **30 ACU View / 150 Transactions / 240 Second Ramp-up**
Active Concurrent Users	50
Total Transactions	250
Start Time	3:23 PM
End Time	3:31 PM
Duration	~8 Minutes

JMeter Output

Edit Workflow:

Summary Report

Name: Edit Web Intelligence Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	100	691	487	1676	165.90	0.00%	15.4/min	107.31	428990.0
Browse Folders	100	1682	1351	2596	235.37	0.00%	15.3/min	97.12	390463.5
View Report	100	2108	1368	3139	408.77	0.00%	15.1/min	234.45	952057.1
Edit and Refresh	100	4382	3227	6501	686.28	0.00%	14.4/min	129.83	552432.4
Close and Logout	100	203	100	1139	128.54	0.00%	15.2/min	7.64	30911.3
TOTAL	500	1813	100	6501	1502.95	0.00%	1.1/sec	506.57	470970.9

For the Edit workflow, you can see the following values:

Test 2 - Edit Webi	**Number of Samples**	**Average**	**Min**	**Max**	**Std Dev**
Login	100	691	487	1676	165.9
Browse Folders	100	1682	1351	2596	235.37
View Report	100	2108	1368	3139	408.77

Test 2 - Edit Webi	Number of Samples	Average	Min	Max	Std Dev
Edit and Refresh	100	4382	3227	6501	686.28
Close and Logout	100	203	100	1139	128.54
Total Transaction	**100**	**9066**	**6533**	**15051**	**1624.86**
TOTAL	500	1813	100	6501	1502.95

We obtain the Total Transaction times for this test is by adding the action step values together for Average, Min, Max and Std Dev. This gives us the average, Fastest (Min) and Slowest (Max) transaction times. For this test, we put over three times load on the system. The first test was for 15 ACUs and this test for 50 ACUs.

Interpretation of the above data

Following are observations from the data:

- Average Transaction Time is **9.066** seconds.
- The Fastest potential transaction is around **6.533** seconds.
- The Slowest potential transaction is around **15.051** seconds.
- The Fastest Action is the **Close and Logout** action.
- The Slowest Action is the **Edit and Refresh** action.

This starts to give us some reference points that we can use to analyze our benchmarking results and to gauge how our system is performing overall. The real value is when we compare these results to another test such as Test 1 where we ran the same actions but a lower number of times.

Here, we compare the JMeter results for the Edit workflow between Test 1 and Test 2.

Comparing Test 1 to Test 2

Below is a comparison of the results for Test 1 and Test 2. First in Wiki format and then as an image with more advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**%Diff**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**
Login	25	100	746	691	8%	532	487	9%	2206	1676	32%	310.08	165.9	87%
Browse Folders	25	100	1659	1682	1%	1277	1351	6%	1964	2596	32%	170.29	235.37	38%

View Report	25	100	2258	2108	7%	1419	1368	4%	7811	3139	149%	1177.88	408.77	188%
Edit and Refresh	25	100	4402	4382	0%	3033	3227	6%	5402	6501	20%	634.45	686.28	8%
Close and Logout	25	100	208	203	2%	100	100	0%	419	1139	172%	85.73	128.54	50%
Total Transaction	**25**	**100**	**9273**	**9066**	**2%**	**6361**	**6533**	**3%**	**17802**	**15051**	**18%**	**2378.43**	**1624.86**	**46%**
TOTAL	125	500	1855	1813	2%	100	100	0%	7811	6501	20%	1584.53	1502.95	5%

Below is an image of the above table with more advanced formatting.

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**%Diff**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**
Login	25	100	746	691	8%	532	487	9%	2206	1676	32%	310.08	165.9	87%
Browse Folders	25	100	1659	1682	1%	1277	1351	6%	1964	2596	32%	170.29	235.37	38%
View Report	25	100	2258	2108	7%	1419	1368	4%	7811	3139	149%	1177.88	408.77	188%
Edit and Refresh	25	100	4402	4382	0%	3033	3227	6%	5402	6501	20%	634.45	686.28	8%
Close and Logout	25	100	208	203	2%	100	100	0%	419	1139	172%	85.73	128.54	50%
Total Transaction	**25**	**100**	**9273**	**9066**	**2%**	**6361**	**6533**	**3%**	**17802**	**15051**	**18%**	**2378.43**	**1624.86**	**46%**
Total	125	500	1855	1813	2%	100	100	0%	7811	6501	20%	1584.53	1502.95	5%

We can see that this Edit Workflow performance is very close to Test 1 results.

Test 1 **9.273** seconds
Test 2 **9.066** seconds
Difference **0.207** seconds (207 ms)

We actually see a faster average response time with the 2nd test. We have a larger sampling though so this helps dilute the first couple of transactions, which may have run slower due to caching and server connection initializations.

But what do we see when we compare more than just the Average Transaction time?

At first glance, the numbers appear to be all over the board with no real consistency. For most of these numbers, that's perfectly fine. There will always be slight inconsistencies between your tests and you will never get the same results (even on an identical test) so don't worry about it too much. What you do want to focus on is what PERCENTAGE difference (% Diff) these values represent. In most cases, a 10-15% range is acceptable and shouldn't raise any concerns. If you are seeing a larger percentage difference, it might be worth investigating.

To calculate the % Difference for the two numbers, you want to use formula: ***(N1 – N2) / N2*** eg. ***(746 – 691) / 691 = 0.08*** or 8% difference as we can see above for the Average Login difference between Test 1 and Test 2.

We can see that the Average times above are all within an acceptable range. The Min (fastest) values are also within the acceptable range. Where we do see a potential problem is with the Max and Standard Deviation values. We would want to look into these as they do have much larger differences between the two tests.

For our results, the Max values are higher on both Test 1 and Test 2 for different actions so this doesn't really give us a pattern to work with. This easily could have just been a small blip in the testing cycle that skewed our results.

Warning: The Min and Max values are really just the lowest and highest single values that were recorded during the test so they don't necessarily weigh heavy on the results. This is why we focus mainly on averages.

For Standard Deviation, we can also see that there were some large percentage differences. These numbers are not consistently higher or lower for one test vs the other so we would conclude that the difference is likely caused by the number of samples that it is working with. Test 1 only had 25 samples whereas Test 2 had 100. Standard Deviation is more interesting when comparing identical tests run multiple times for this reason.

Next, we will see how the View workflow compared to Test 1.

View Workflow:

Summary Report

Name: View and Refresh Webi Report

Comments:

Write results to file / Read from file

Filename Browse... **Log/Display Only:** Errors Successes **Configure**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	150	719	489	1806	181.81	0.00%	23.0/min	160.97	429058.5
Browse Folders	150	1673	1228	2535	231.14	0.00%	22.9/min	145.76	390461.1
View Report	150	1194	782	2357	237.19	0.00%	23.4/min	91.32	239967.9
Refresh and Prompts	150	2001	1444	4157	371.52	0.00%	23.3/min	96.81	255691.2
Close and Logout	150	234	109	977	127.55	0.00%	23.3/min	12.56	33146.1
TOTAL	750	1164	109	4157	681.29	0.00%	1.8/sec	467.55	269664.9

For the View workflow, we see the following values:

View Web Intelligence	Number of Samples	Average	Min	Max	Std Dev
Login	150	719	489	1806	181.81
Browse Folders	150	1673	1228	2535	231.14
View Report	150	1194	782	2357	237.19

View Web Intelligence	Number of Samples	Average	Min	Max	Std Dev
Refresh and Prompts	150	2001	1444	4157	371.52
Close and Logout	150	234	109	977	127.55
Total Transaction	**150**	**5821**	**4052**	**11832**	**1149.21**
TOTAL	750	1164	109	4157	681.29

Interpretation of the above data

From the above data, we make the following observations:

- The Average total Transaction time is **5.821** seconds
- The Fastest potential Transaction time can be **4.052** seconds
- The Slowest potential Transaction time could be **11.832** seconds
- The Slowest action is the **Edit and Refresh**

Comparing Test 1 to Test 2

The following table gives a comparison of the results. The first table is in Wiki format and the second is a screenshot with more formatting:

	Number of Samples		Average			Min			Max			Std Dev		
View Webi	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**
Login	50	150	682	719	5%	537	489	10%	1337	1806	35%	115.33	181.81	58%
Browse Folders	50	150	1613	1673	4%	1320	1228	7%	2015	2535	26%	182.06	231.14	27%

View Report	50	150	1301	1194	9%	900	782	15%	6676	2357	183%	825.18	237.19	248%
Refresh & Prompt	50	150	2136	2001	7%	1580	1444	9%	5624	4157	35%	608.65	371.52	64%
Close and Logout	50	150	225	234	4%	115	109	6%	444	977	120%	70.5	127.55	81%
Total Transaction	50	150	**5957**	**5821**	**2%**	**4452**	**4052**	**10%**	**16096**	**11832**	**36%**	**1801.72**	**1149.21**	**57%**
Total	250	750	1191	1164	2%	115	109	6%	6676	4157	61%	821.83	681.29	21%

Below is an image of the above table with advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
View Webi	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**%Diff**	**Test 1**	**Test 2**	**% Diff**	**Test 1**	**Test 2**	**% Diff**
Login	50	150	682	719	5%	537	489	10%	1337	1806	35%	115.33	181.81	58%
Browse Folders	50	150	1613	1673	4%	1320	1228	7%	2015	2535	26%	182.06	231.14	27%
View Report	50	150	1301	1194	9%	900	782	15%	6676	2357	183%	825.18	237.19	248%
Refresh & Prompt	50	150	2136	2001	7%	1580	1444	9%	5624	4157	35%	608.65	371.52	64%
Close and Logout	50	150	225	234	4%	115	109	6%	444	977	120%	70.5	127.55	81%
Total Transaction	**50**	**150**	**5957**	**5821**	**2%**	**4452**	**4052**	**10%**	**16096**	**11832**	**36%**	**1801.72**	**1149.21**	**57%**
Total	250	750	1191	1164	2%	115	109	6%	6676	4157	61%	821.83	681.29	21%

Comparing the Average Total Transaction time:

Test 1 **5.957** seconds
Test 2 **5.821** seconds
Difference **0.136** seconds (~2% better)

We see that Test 2 performs better than Test 1 as there are more samples. This close timing affirms that our benchmark value should be around the 5.8 second mark for a View Web Intelligence request.

As for the Min or Max or Standard Deviation values, we see some fluctuations. However, Test 2 runs quicker on most comparisons. These values are used to compare multiple runs of identical tests as mentioned previously.

Test 2 shows significantly lower values of Max for the “View Report” and “Refresh & Prompt” actions. This is because we ran a few test reports ahead of the test to warm up the cache and create the system connections between the components.

CA Introscope Output:

The screenshots for Test 1 and Test 2 help us compare graphs to give an insight into potential bottlenecks or system issues. We use these screenshots to validate the JMeter findings.

We have the following dashboards captured for Test 2.

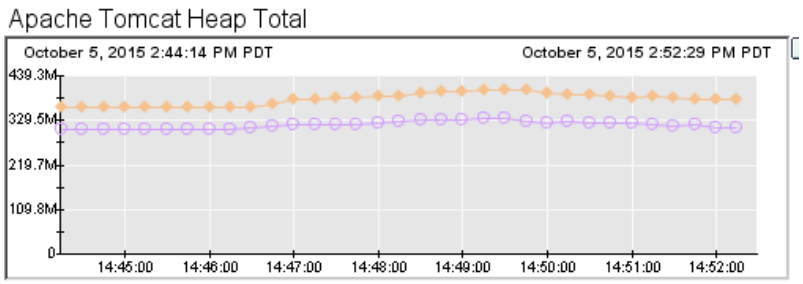
- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

We will compare a few of these graphs from Test 1 and Test 2 below to demonstrate how a comparison can be carried out.

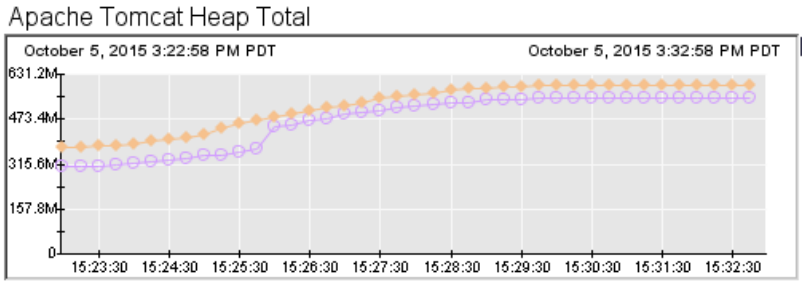
Apache Tomcat Memory Dashboard

Figure: Apache Tomcat Heap Total

Test 1 – 15 ACU



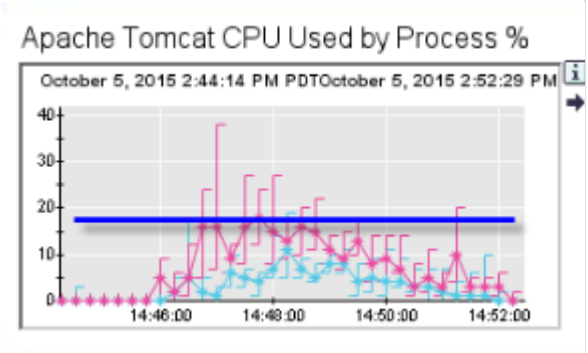
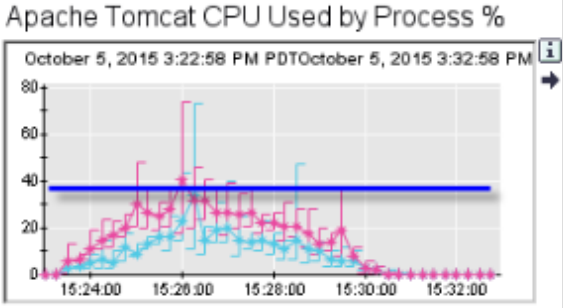
Test 2 – 50 ACU

Test 1 – 15 ACU


Comparison Notes: For Test 1, the maximum heap reaches 350px about 430 MB during the test. For the Test 2, it rises up to around 630mb. This gives us an idea of how much more heap is required to run 50 ACU across these two Tomcat Servers. You can use this to validate your sizing and ensure that you are not hitting a bottleneck in your testing. We have a Maximum Heap Size (-Xmx) on our Tomcat servers set to 4GB in our pattern which is well under the maximum heap for this test. We do not run large reports in this test, so we do not expect to exceed our heap.

Apache Tomcat Triage

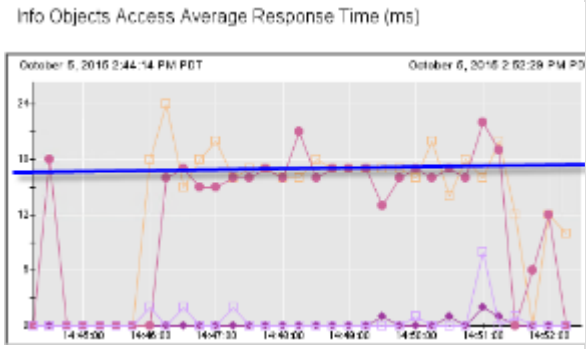
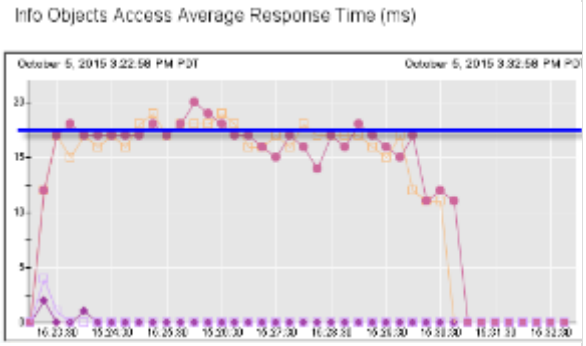
Apache Tomcat CPU Used by Process %

Test 1 – 15 ACU	Test 2 – 50 ACU
	

Comparison notes: Test 1 reaches an average ceiling of around 15-20% CPU Utilization. Test 2 reaches about 35-40% CPU Utilization indicating that there is some room to grow and that CPU Utilization does not triple when we triple the load. We see a few spikes in both tests which is likely while the tomcat process is doing garbage collection. We can make some tweaks to reduce this impact.

Apache Tomcat Servlets

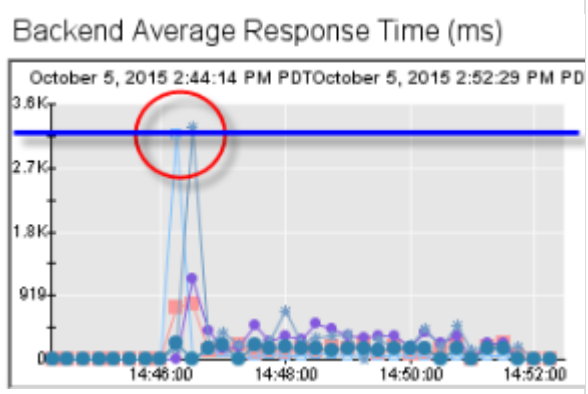
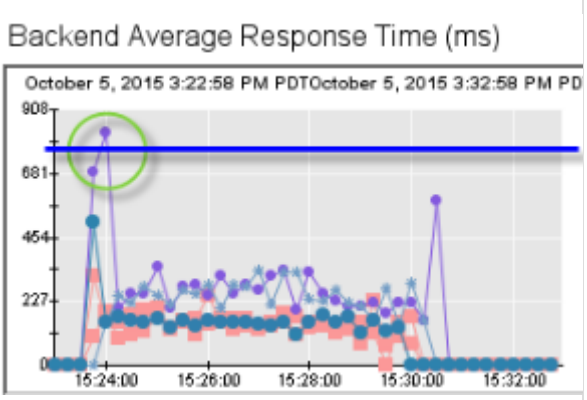
Apache Tomcat Servlet Response Time

Test 1 – 15 ACU	Test 2 – 50 ACU
	

Comparison notes: Here the scale is different, but you can see the average InfoObjects Access time is almost the same. 18ms is the average. This represents the time CMS takes to find a requested InfoObject. If CMS has the Object loaded in memory, this will be a quick request. If the CMS has to query the System Database for objects, this access time slows down generally. Spikes here indicate a bottleneck on the Central Management System (CMS) side, which is eliminated by increasing the number of records the CMS caches by default.

BOE 4.0 Triage

Backend Average Response Time (ms)

Test 1 – 15 ACU	Test 2 – 50 ACU
	

Comparison Notes: Test 1 shows average back-end response times that are higher than Test 2. For example, we see two data points that are above 3000ms (3s) for Test 1 and in Test 2, and a peak average of between 700-850ms which is less. This could account for longer Max times that in Test 1 JMeter output. These occur at the beginning of the test due to the "warm-up" time.

Warning: We did "warm-up" the system before executing Test 2. We did this by manually running a few reports to initiate connections and fill the cache.

Summary of CA Introscope Output Analysis for Test 1 vs Test 2

The JMeter results indicated that the tests performed pretty consistently based on the average transaction times of both the view and edit workflows. We saw nearly the same average times between Test 1 and 2.

In the CA Introscope output, we could see similar results. The CA Introscope data didn't uncover any major bottlenecks or delays that we would have to worry about. This is fantastic because we didn't run a huge load and we didn't expect to see many differences given the server configuration and sizing that we are using.

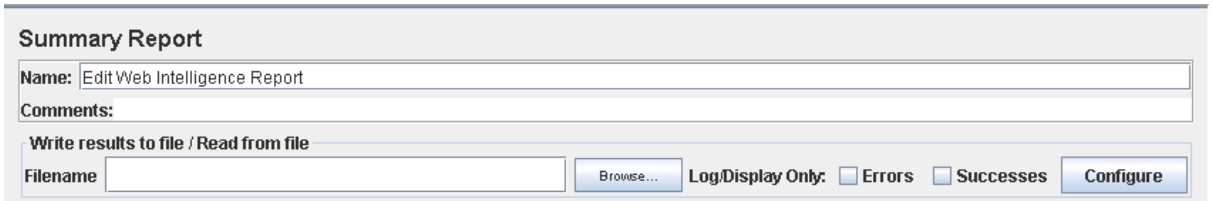
Introscope did show us that CPU and Memory resources went up with the heavier load and they also showed us that average response time stayed pretty well consistent for the metrics that we reviewed. We found a few outliers that we can explain and that show up in our JMeter results as well.

All in all, this comparison went well and there were no major surprises. It's time to crank it up a little and see what we can uncover in Test 3.

Resources for Test 2

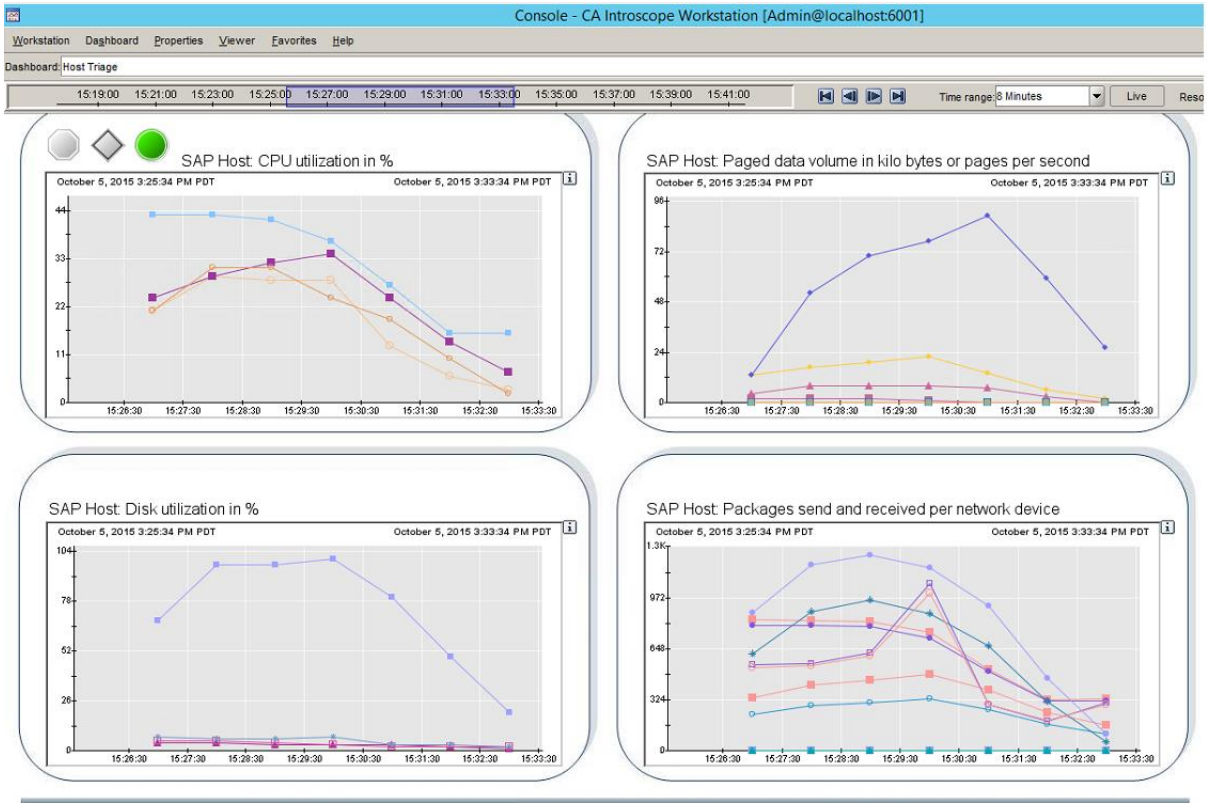
These are the CA Introscope dashboard screenshots captured during this test.

JMeter – Edit – Summary Report

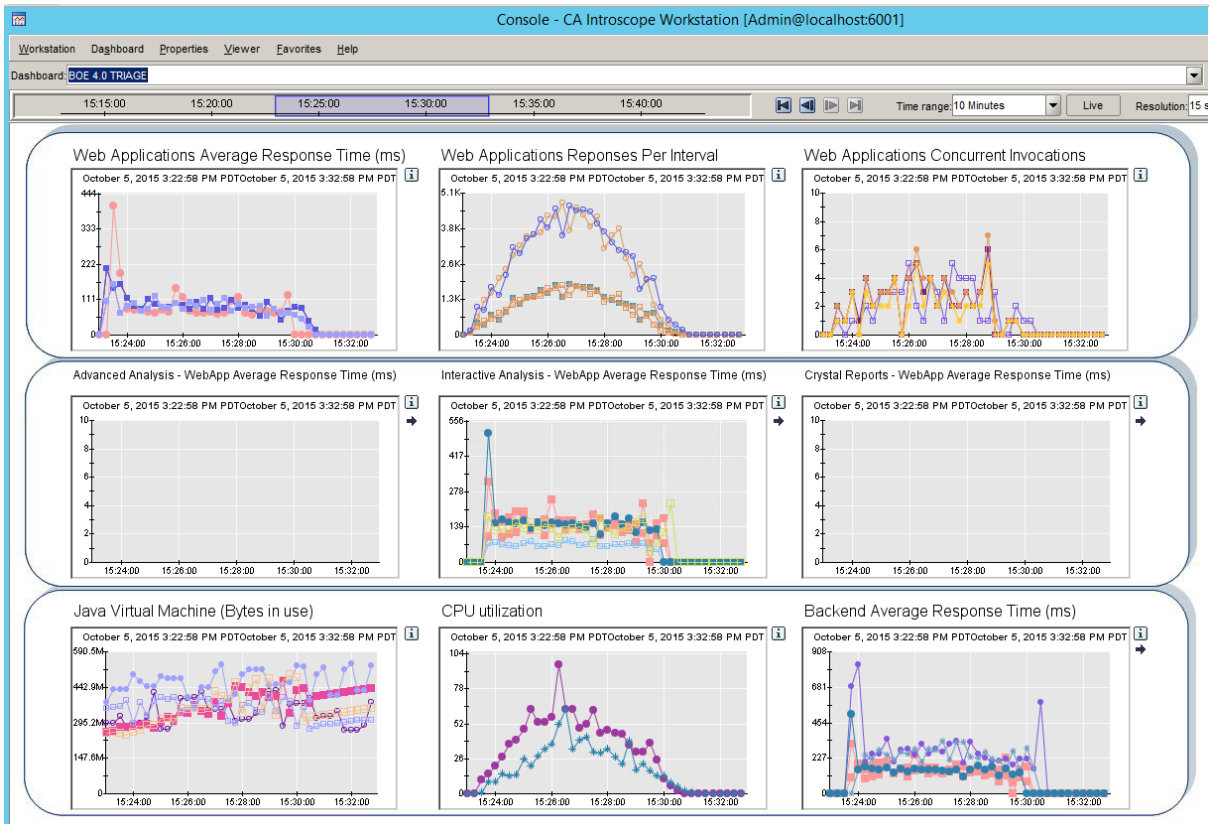


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	100	691	487	1676	165.90	0.00%	15.4/min	107.31	428990.0
Browse Folders	100	1682	1351	2596	235.37	0.00%	15.3/min	97.12	390463.5
View Report	100	2108	1368	3139	408.77	0.00%	15.1/min	234.45	952057.1
Edit and Refresh	100	4382	3227	6501	686.28	0.00%	14.4/min	129.83	552432.4
Close and Logout	100	203	100	1139	128.54	0.00%	15.2/min	7.64	30911.3
TOTAL	500	1813	100	6501	1502.95	0.00%	1.1/sec	506.57	470970.9

Host 2 Triage – Test 2

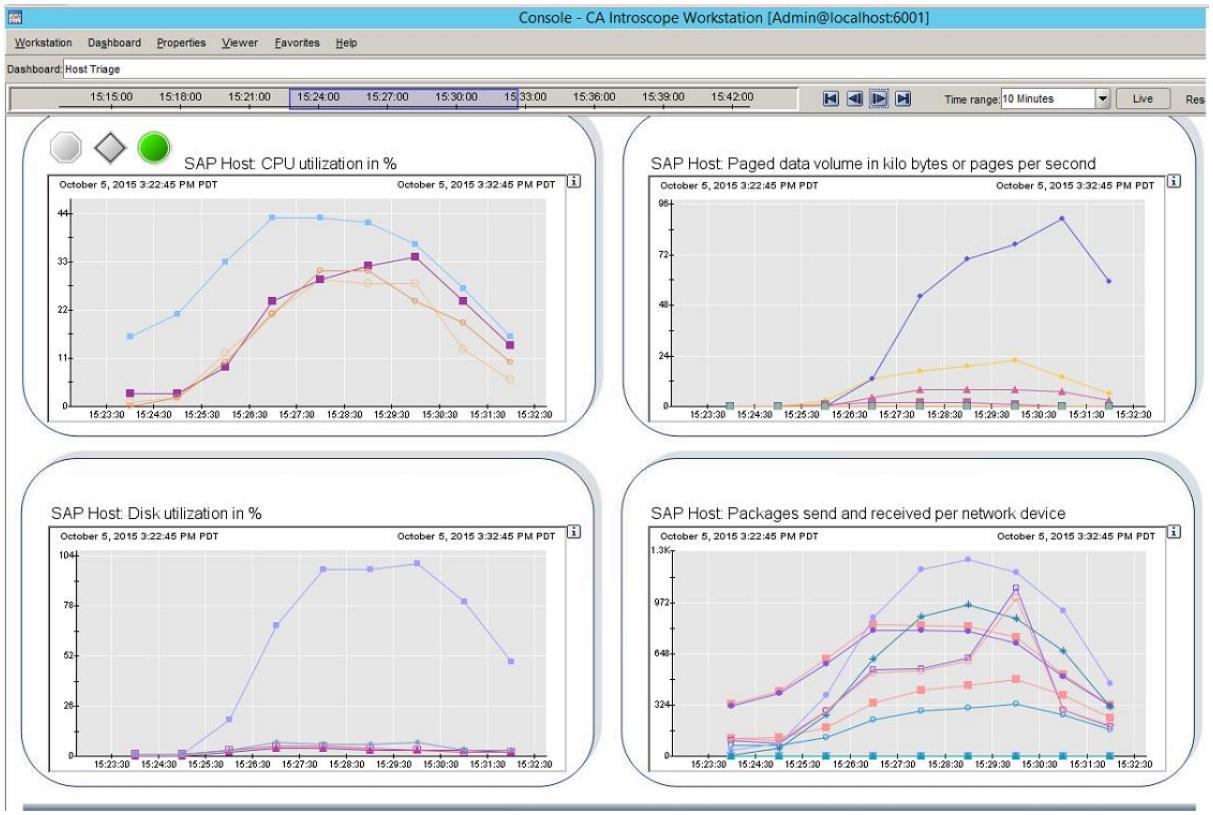


BI Triage – Test 2

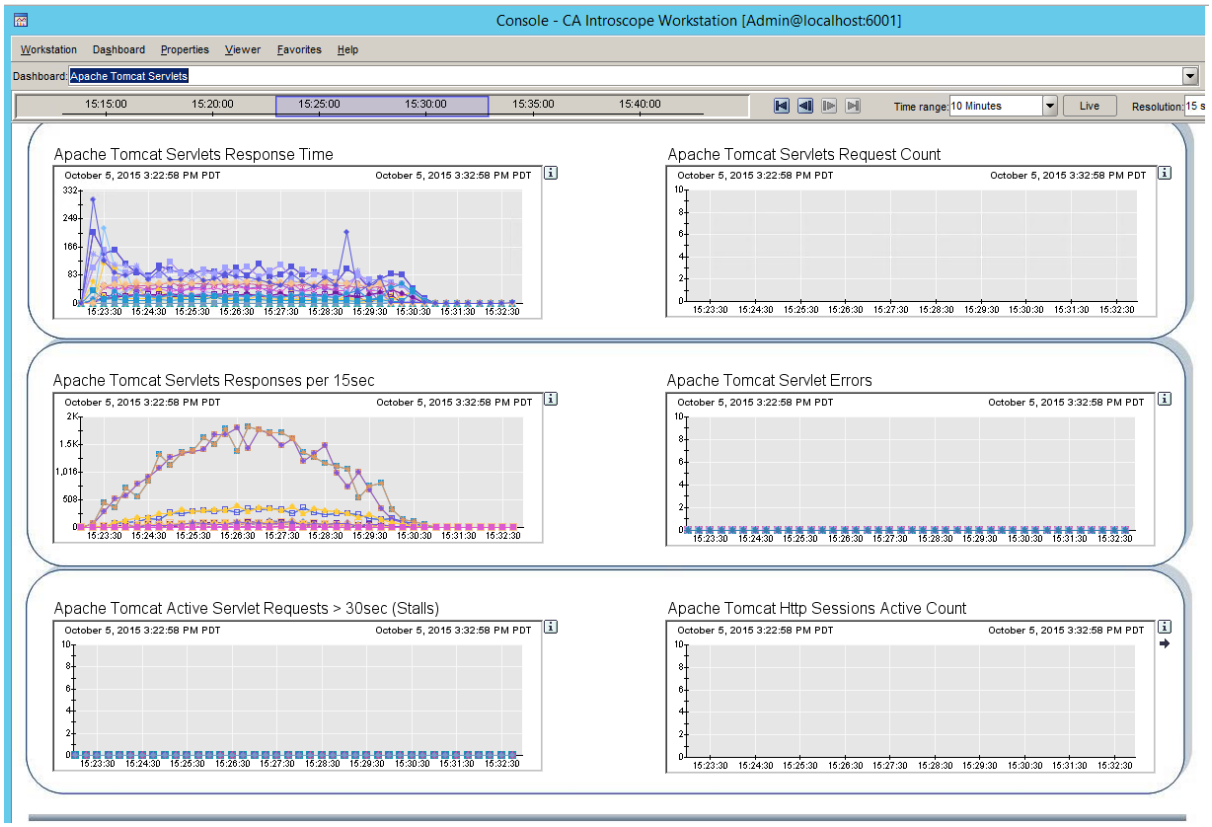


Bottleneck Alert – Test 2

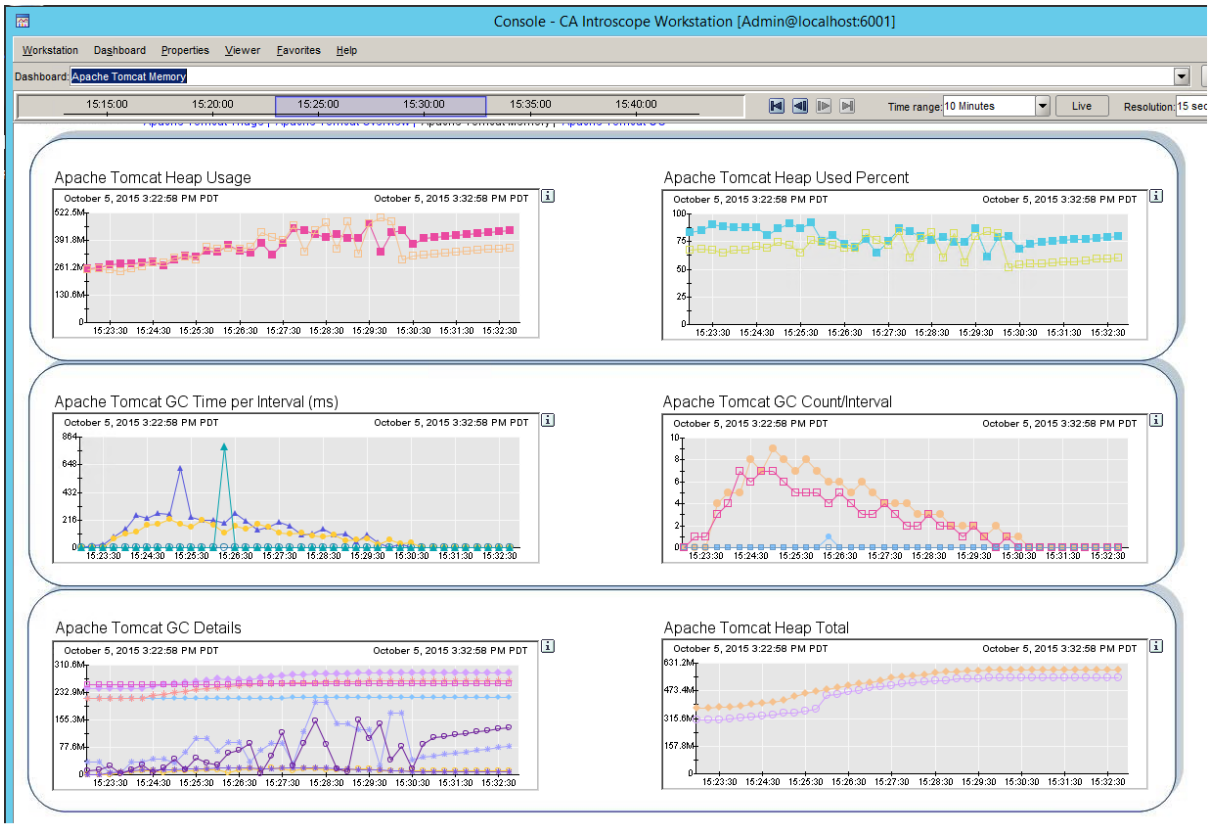
Host Triage – Test 2-2



Tomcat Servlets – Test 2



Tomcat Memory – Test 2

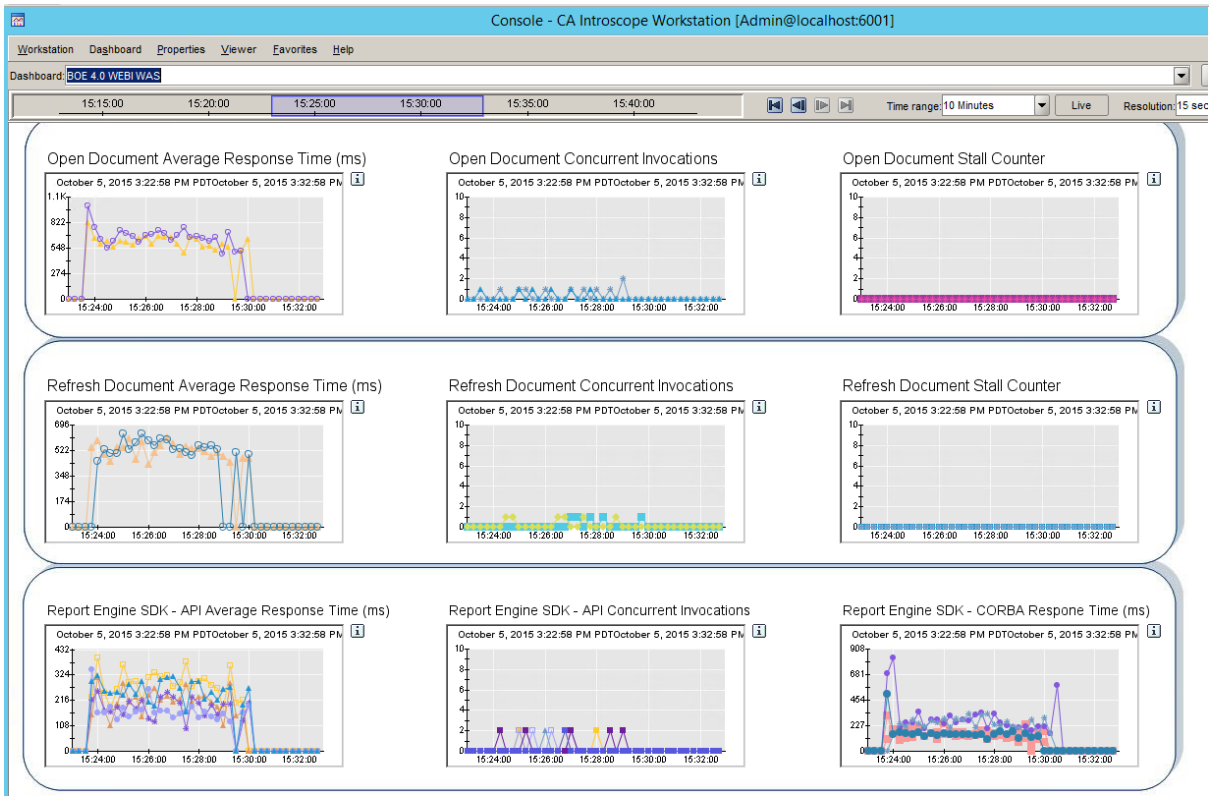


Platform Averages – Test 2

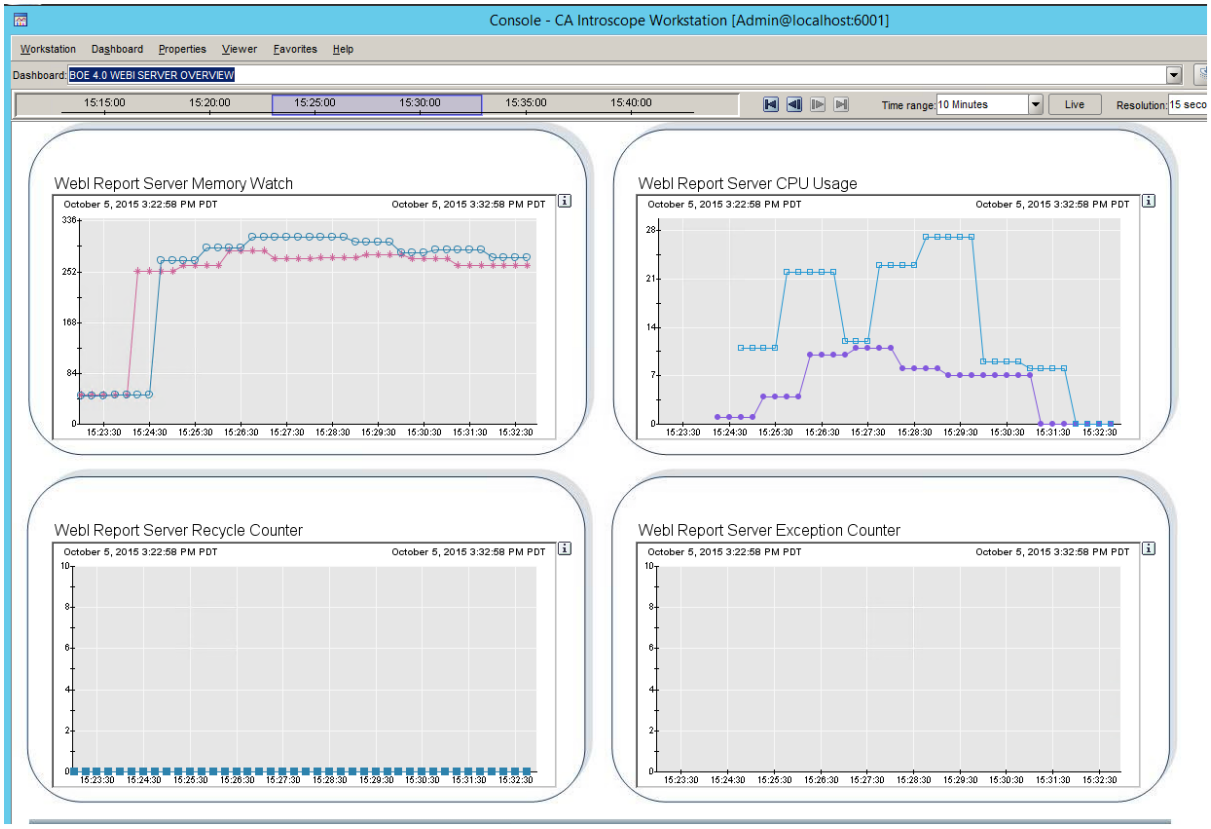
Tomcat CPU Usage – Test 2

Tomcat GC Details – Test 2

WebI WAS – Test 2

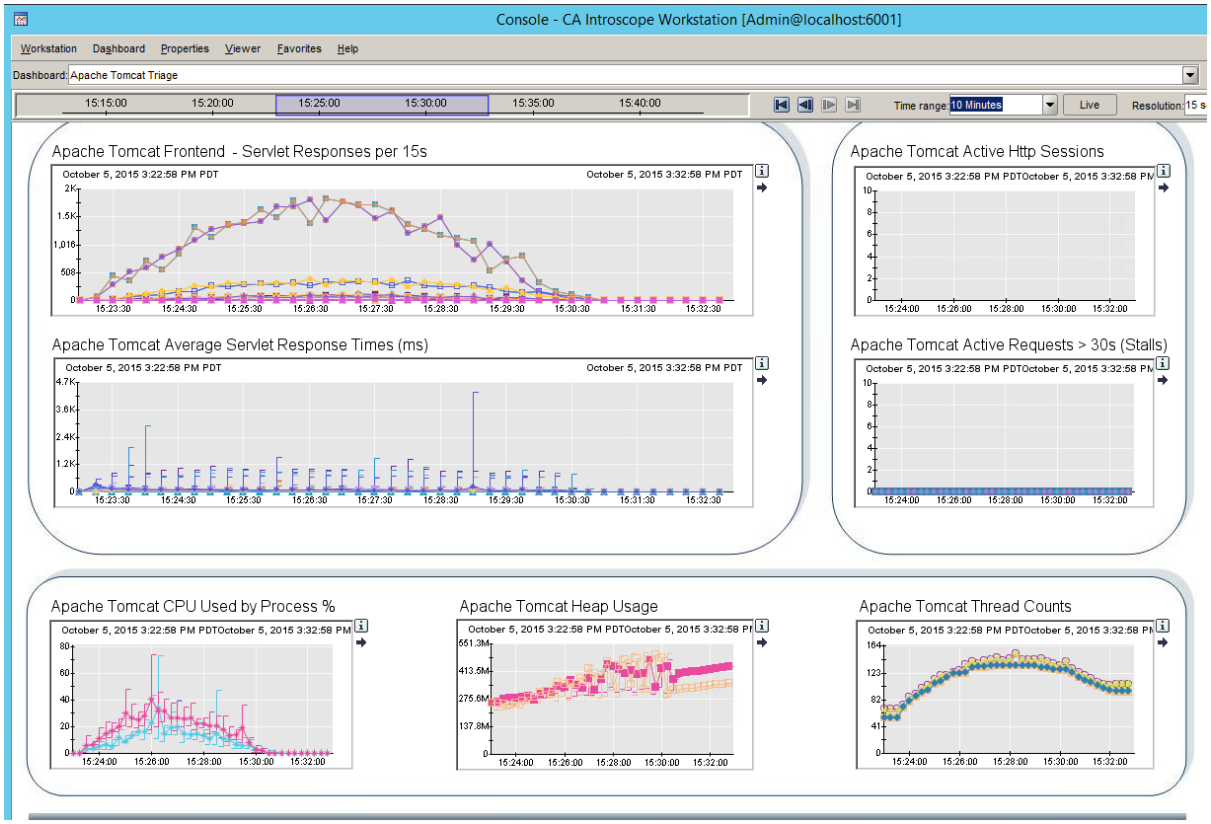


WebI Server Overview – Test 2

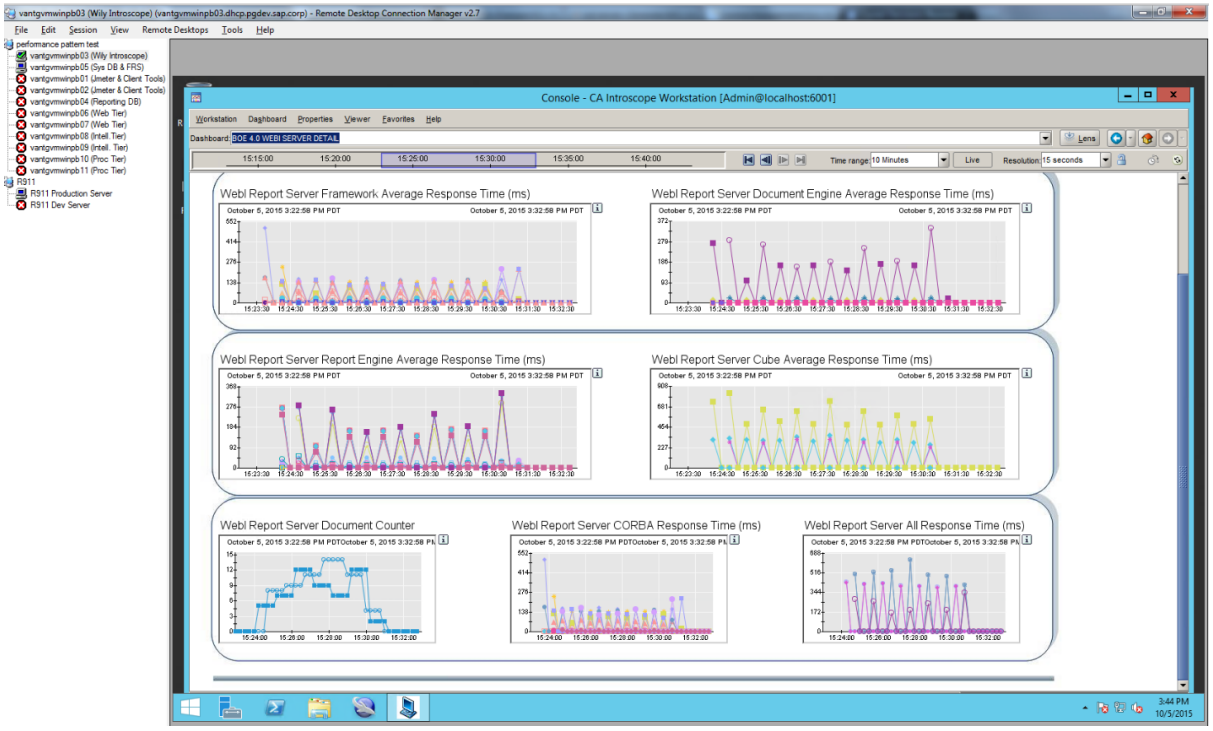


Tomcat Threads – Test 2

Tomcat Triage – Test 2



WebI Server Details – Test 2



Test 3- 150 Active Concurrent Users

Now that we know our system can handle 15-50 ACUs with little to no difference in performance, we wanted to see how it handled 150! This is about 3 times the load that we used in Test 2 and should start skewing results.

Details of Test 3 are as follows:

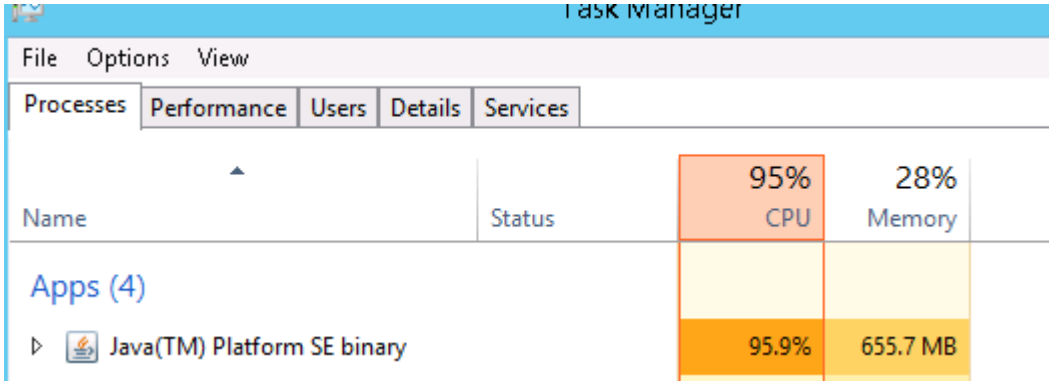
Test 3	• **60 ACU Edit / 300 Transactions / 120 Second Ramp-up** • **90 ACU View / 450 Transactions / 120 Second Ramp-up**
Duration	~8 Minutes
Active Concurrent Users	150
Start Time	3:47 PM
End Time	3:55 PM
Total Transactions	750

As you can see, we tripled the ACUs and the Transactions that we would try to push through our system. We also lowered the ramp-up time on the View requests to increase the load on the server.

FAIL!

This test started off fine but a few minutes in we started to notice an issue. The JMeter box was pegged at close to 100% CPU.

Figure: Test Fail



On further investigation, we discovered that the initial sizing of the JMeter client was insufficient for the amount of transactions we were trying to run. Most notably, the maximum heap space was set to only 500mb and clearly we needed more memory to complete this test. JMeter went into a hung state where it continually attempted to clean up memory through garbage collection.

We killed the process and aborted our test. It's a good thing the box had room to grow and we had a 2nd JMeter server to use in a cluster. This was one of the roadblocks we anticipated

in our sizing but weren't sure where and when we might hit it. We also didn't really know how it would surface!

A quick google search landed us on the JMeter Wiki hosted here: http://wiki.apache.org/jmeter/JMeterFAQ#JMeter_keeps_getting_.22Out_of_Memory.22_errors._What_can_I_do.3F

This wiki recommends updating the –Xms and –Xms (minimum and maximum heap space) settings in the jmeter.bat file.

Here is a screenshot of our jmeter.bat

Figure: Screenshot of jmeter.bat

```
if .%JM_LAUNCH% == . set JM_LAUNCH=java.exe

if exist jmeter.bat goto winNT1
if .%JMETER_BIN% == . set JMETER_BIN=%~dp0

:winNT1
rem On NT/2K grab all arguments at once
set JMETER_CMD_LINE_ARGS=%*

rem The following link describes the -XX options:
rem http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html
rem http://java.sun.com/developer/TechTips/2000/tt1222.html has some more descriptions
rem Unfortunately TechTips no longer seem to be available

rem See the unix startup file for the rationale of the following parameters,
rem including some tuning recommendations
set HEAP=-Xms4g -Xmx4g
set NEW=-XX:NewSize=128m -XX:MaxNewSize=128m
set SURVIVOR=-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50%
set TENURING=-XX:MaxTenuringThreshold=2
```

Since our machine was dedicated to be our jmeter server, we set our Xms and Xmx to 4g (4 GB) which will reserve 4GB of RAM when the process is started and will limit the number of garbage collections that are needed later on. This should also speed up some memory access calls when it is running too as memory addresses will be readily available for the process.

Ensure that you tune your benchmarking software to ensure it can handle the load you plan on using against it.

Note: Ensure that you tune your benchmarking software to ensure it can handle the load you plan on using against it.

We should not run into these JMeter memory issues in the next test.

Test 4- 150 Active Concurrent Users

Test 3 failed due to an insufficient heap size on the JMeter server. We toyed with either introducing our 2nd JMeter server into the mix, or just increasing the Xmx (Max Heap) for

our single JMeter server and ended up going with the latter. We now have a heap size of 4GB which is 8x more than the previous test. This should be plenty for what we are doing.

Details of Test 4 is as follows:

Test 4 (same as Test 3)	• **60 ACU Edit / 300 Transactions / 120 Second Ramp-up** • **90 ACU View / 450 Transactions / 120 Second Ramp-up**
Active Concurrent Users	150
Total Transactions	750
Start Time	4:17 PM
End Time	4:26 PM
Duration	~9 Minutes (but it was aborted)

We did keep the ramp-up time lowered for View Requests on this run. We wanted to see if that had an effect now that we sorted the JMeter issue.

FAIL #2

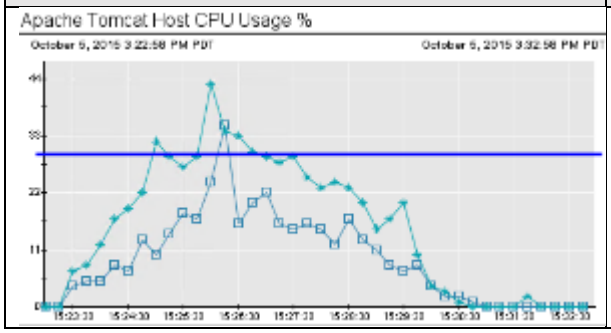
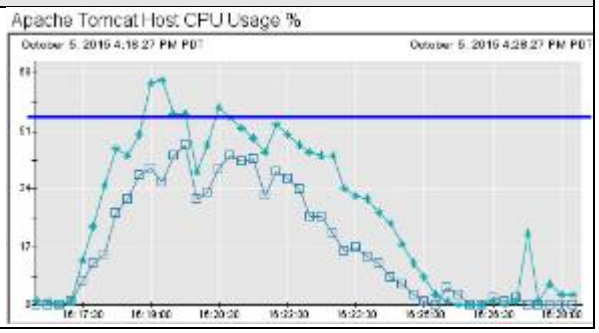
This time the heap size proved to be sufficient and we did not see issues on the JMeter side. JMeter was just the first bottleneck in the mix apparently and this time we were experiencing others as well. This is great actually as this is the whole point of using load testing and benchmarking software.

We aborted this test part way through after seeing much higher averages across the board. We suspected something else might be going on. The delays were significant enough that we were sure we'd hit a bottleneck of some sort.

We unfortunately didn't capture the JMeter output of this test but we did manage to grab some CA Introscope screenshots. Here are a few of the interesting highlights for these, when compared to Test 2.

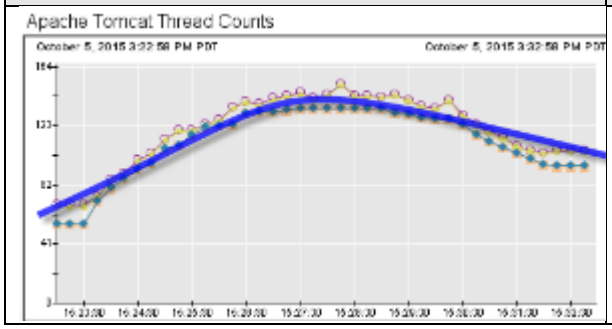
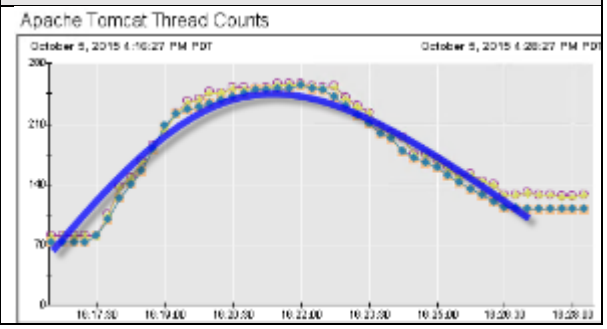
Apache Tomcat CPU

Apache Tomcat Host CPU Usage

Test 2 – 50 ACU	Test 4 – 150 ACU
	

Comparison Notes: We can see a little bit more CPU being utilized by the Tomcat Servers but we are not hitting a bottleneck as far as CPU power goes on either of the tomcat hosts. This means the bottleneck is likely found at a different tier. We can also see that the curve of the CPU usage is more aggressive for Test 4. This is likely due to the decreased ramp-up time that we used.

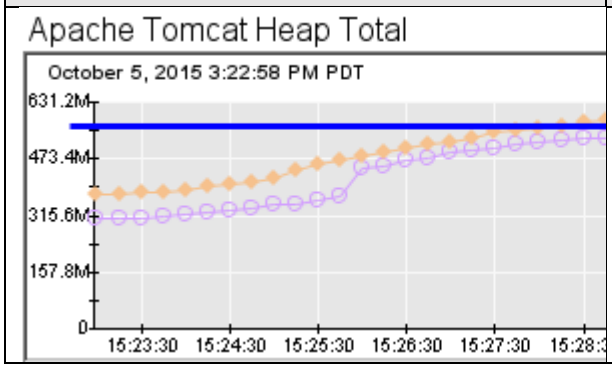
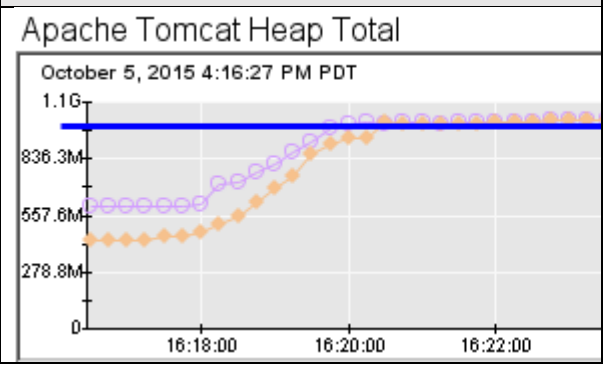
Apache Tomcat Thread Counts

Test 2 – 50 ACU	Test 4 – 150 ACU
	

Comparison Notes: You can see by the steeper curve in the Test 4 graph that our ramp-up time was likely too aggressive. For Test 2, we can see a nice gradual ramp-up to a peak of around 140 threads per Tomcat server. Test 4 shows a steep jump from around 70 threads to 250 in about 3 minutes. This made us reevaluate the ramp up changes we made.

Apache Tomcat Memory

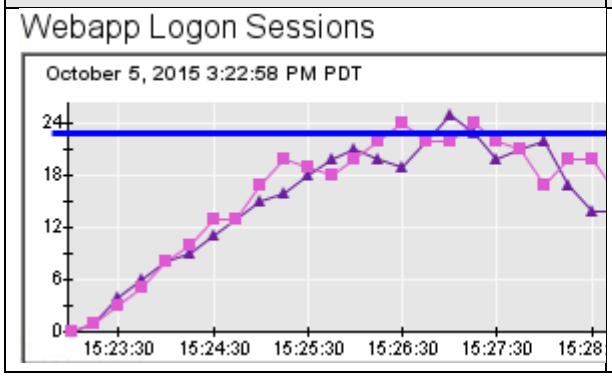
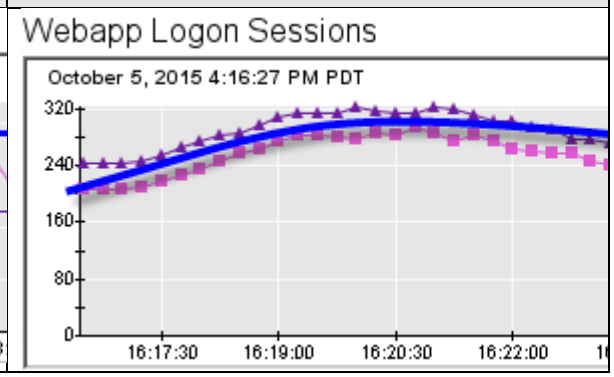
Apache Tomcat Heap Total

Test 2 – 50 ACU	Test 4 – 150 ACU
	

Comparison Notes: Test 2 used a maximum heap of around 600MB. Test 4 reach about 1GB. The Tomcat machines both use an Xmx of 4G which means they are well below the maximum heap space limits. This does not appear to be a bottle neck with Memory on the Tomcat machines.

BOE 4.0 Bottleneck Alert

Webapp Logon Sessions

Test 2 – 50 ACU	Test 4 – 150 ACU
	

Comparison Notes: For Test 2, there was a clear ramp-up and ramp-down period where we went from 0 – 25 Logon Sessions per server. This is expected as we ramp up to 50 ACUs in Test 2 so 25 per server would give us our 50 ACUs. For Test 4 however, the # of logon sessions starts at around 200 and ramps up to over 300. This is not expected and is likely the bottleneck we were looking for.

At this stage, we believe the previous failed test left a number of sessions open as JMeter was killed due to it hanging. This test will need to be burned as well due to dirty results.

Information:

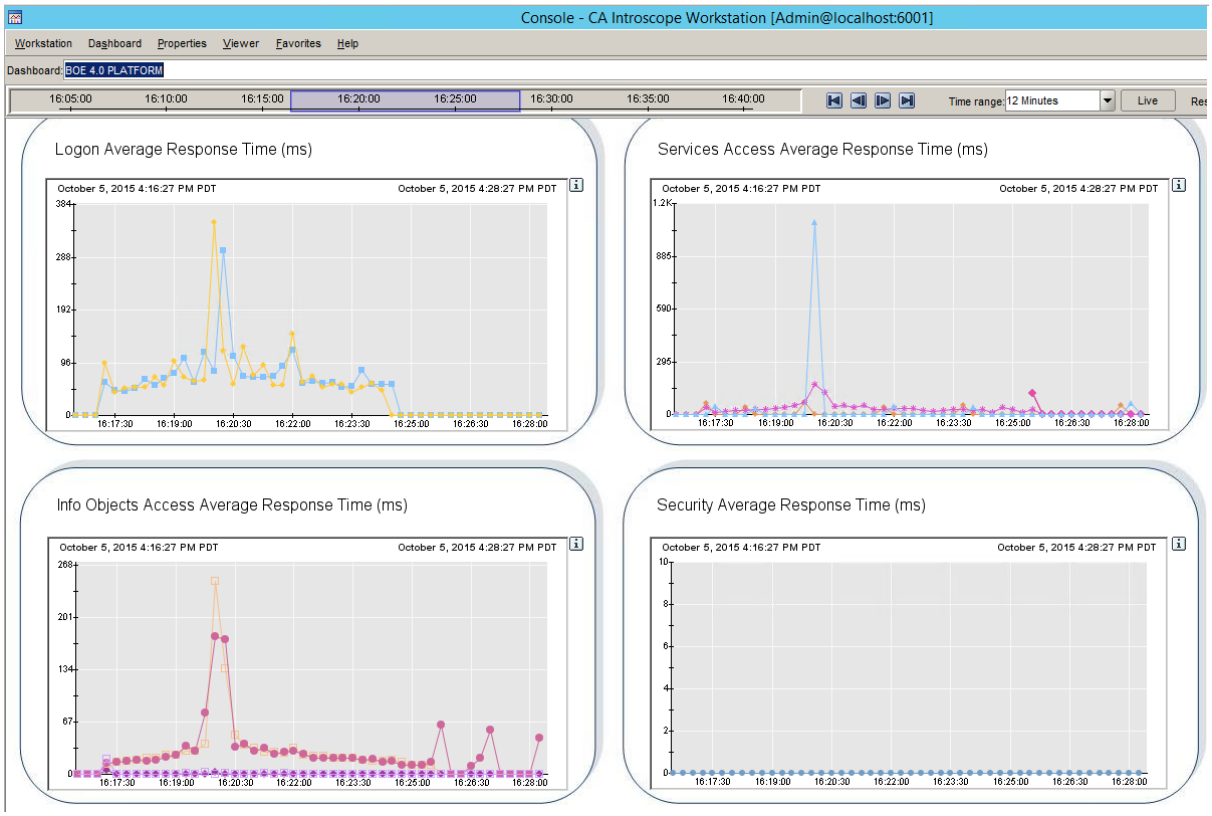
It is a good idea to reset your testing environment by either restarting the processes or the entire boxes after a few testing cycles. This ensures that there is nothing left over from previous runs that affect your results.

Resources for Test 4 - 150 ACU

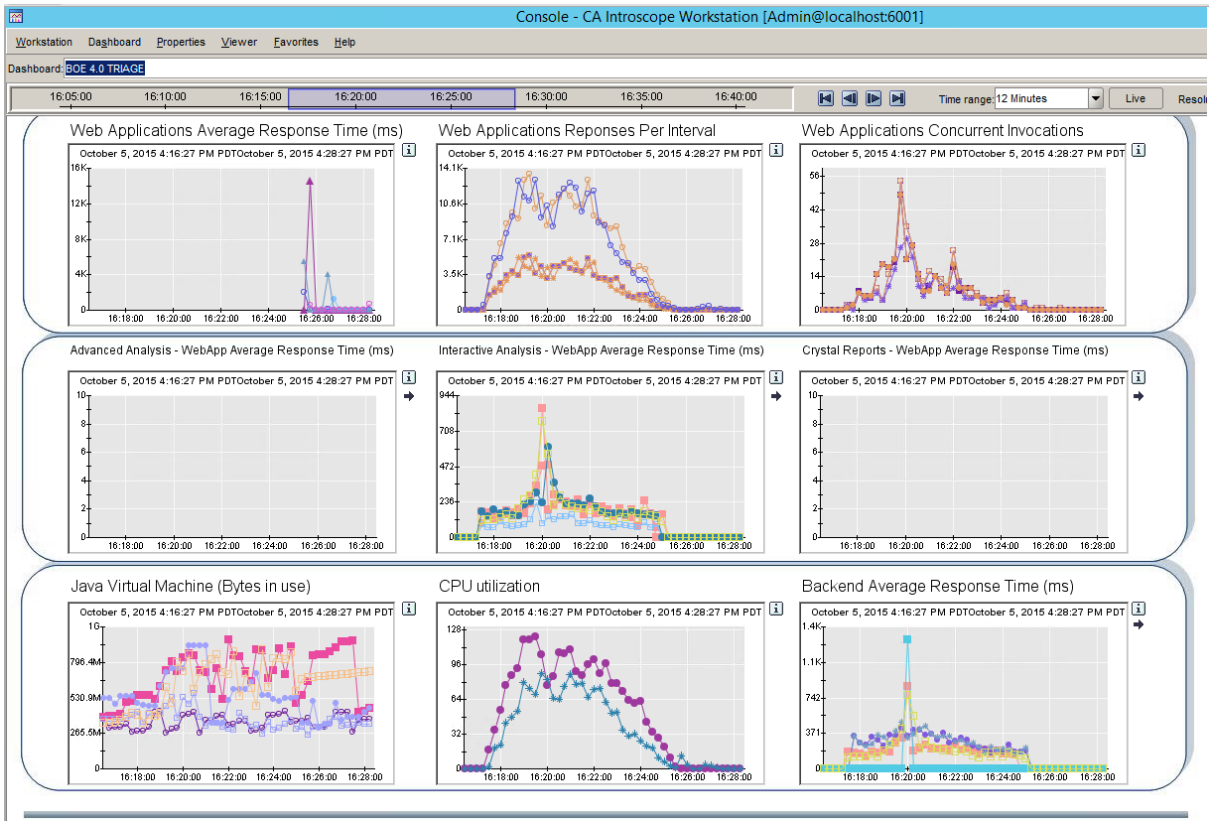
These are the CA Introscope dashboard screenshots that we collected from our Test 4 run.

Tomcat CPU Usage – Test 4

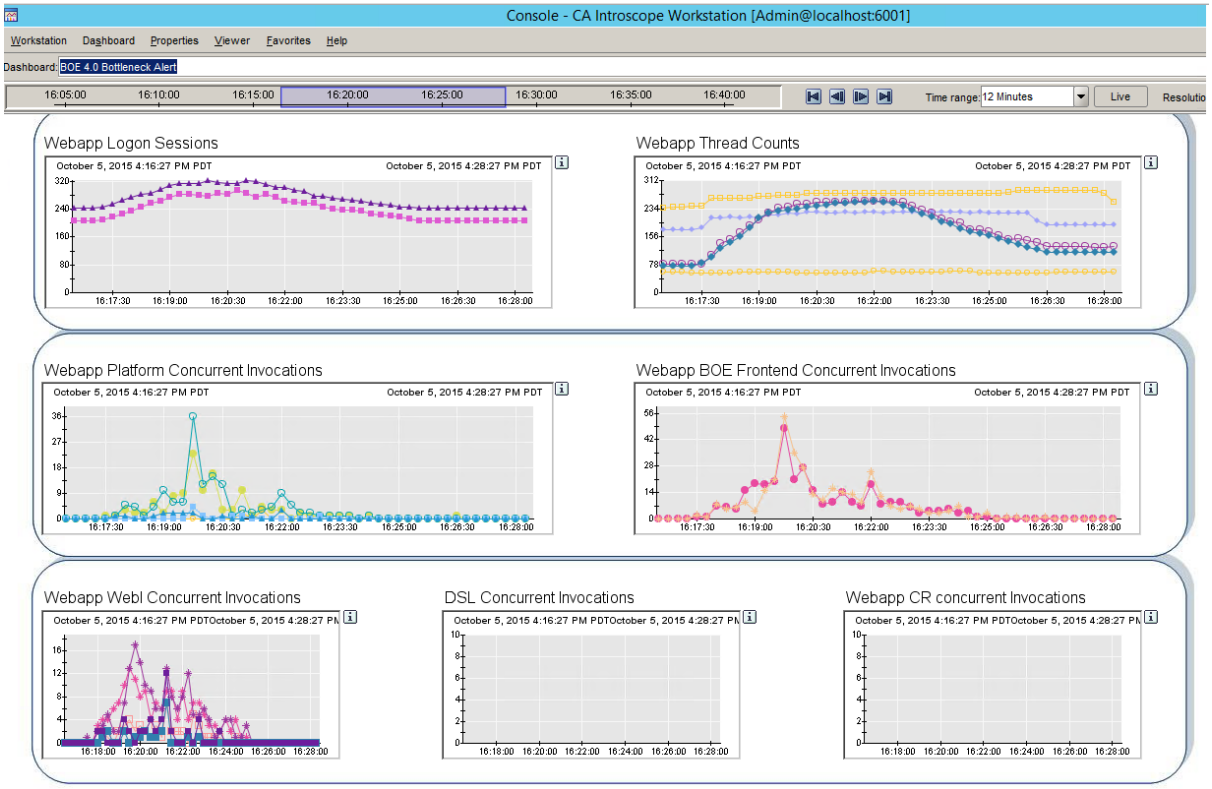
Platform Averages – Test 4



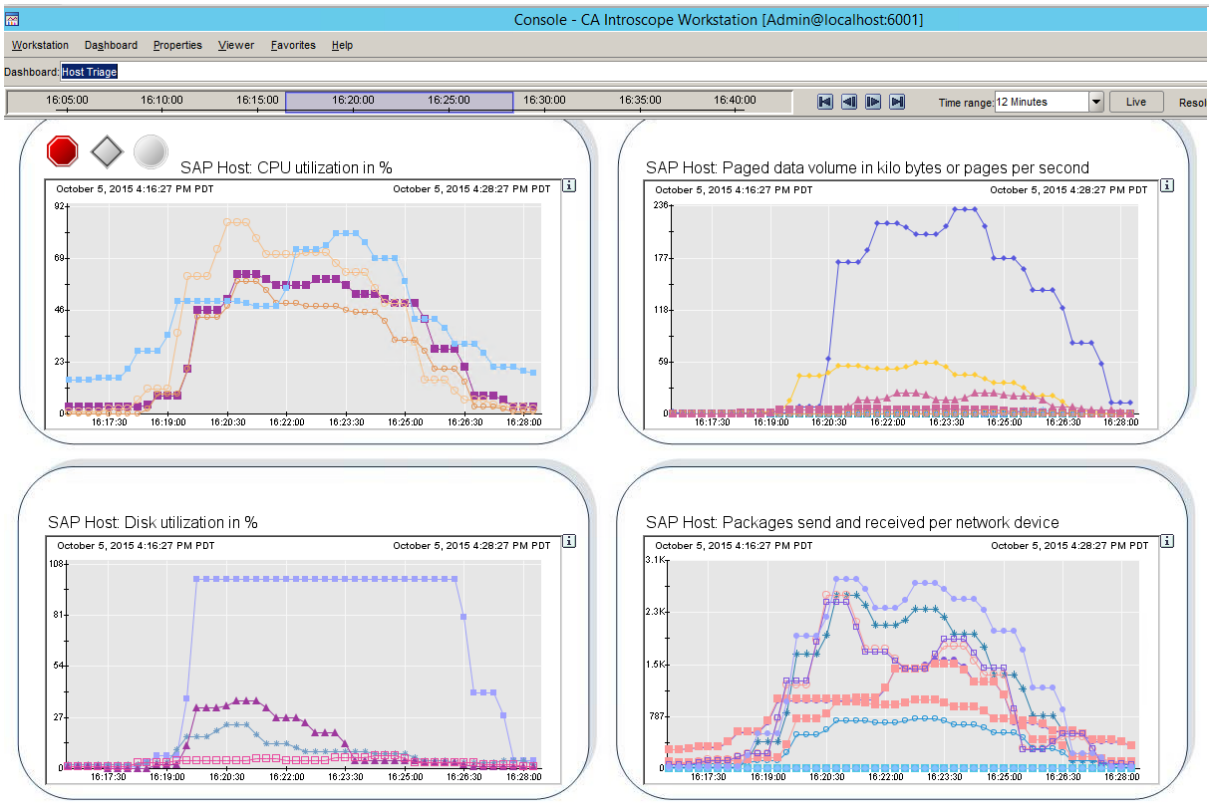
BI Triage – Test 4



Bottleneck Alert – Test 4

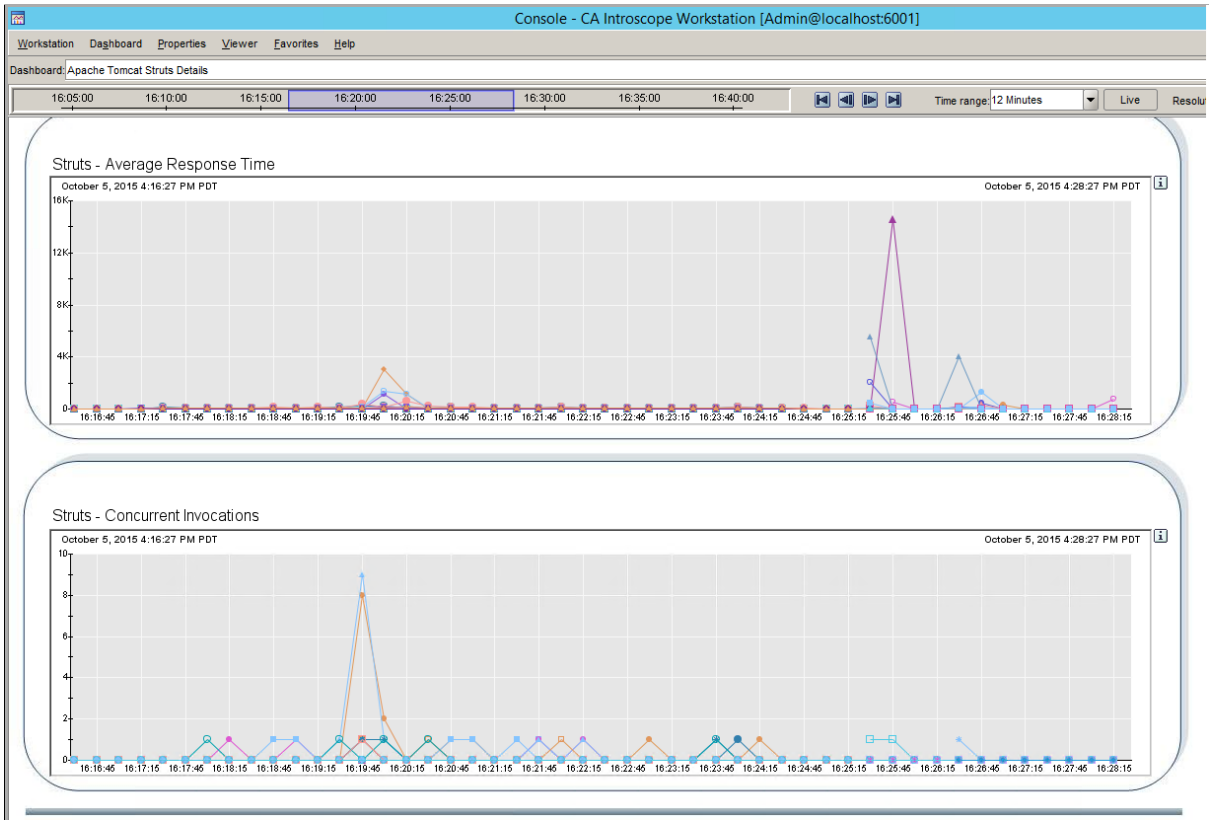


Host Triage – Test 4



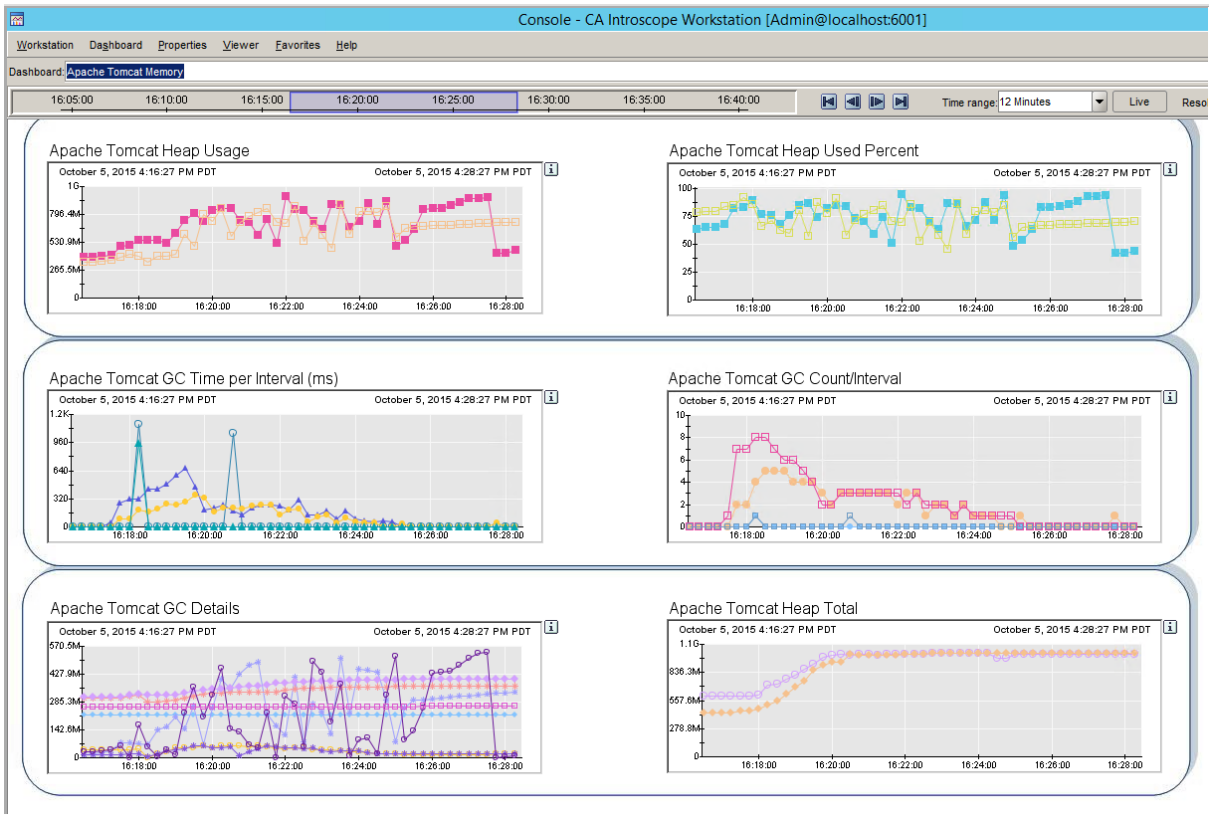
Tomcat Threads – Test 4

Tomcat Strut – Test 4

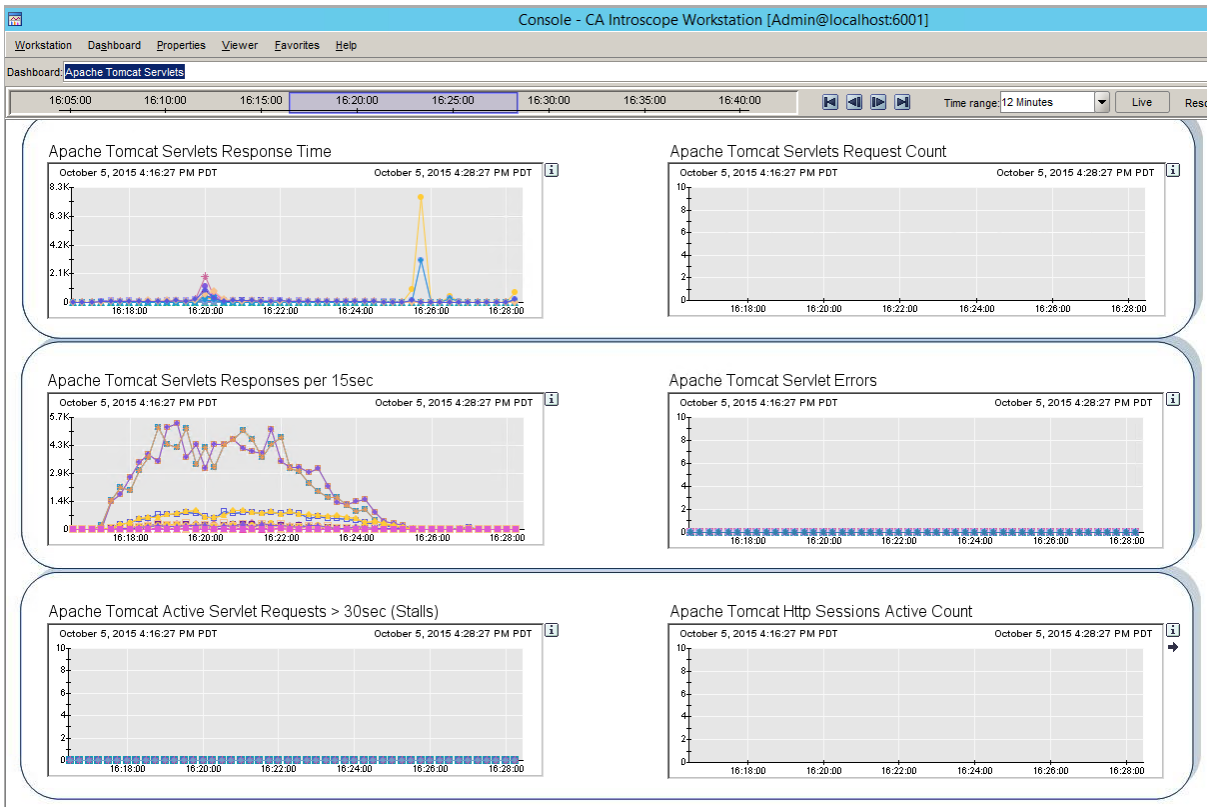


Tomcat GC Details –Test 4

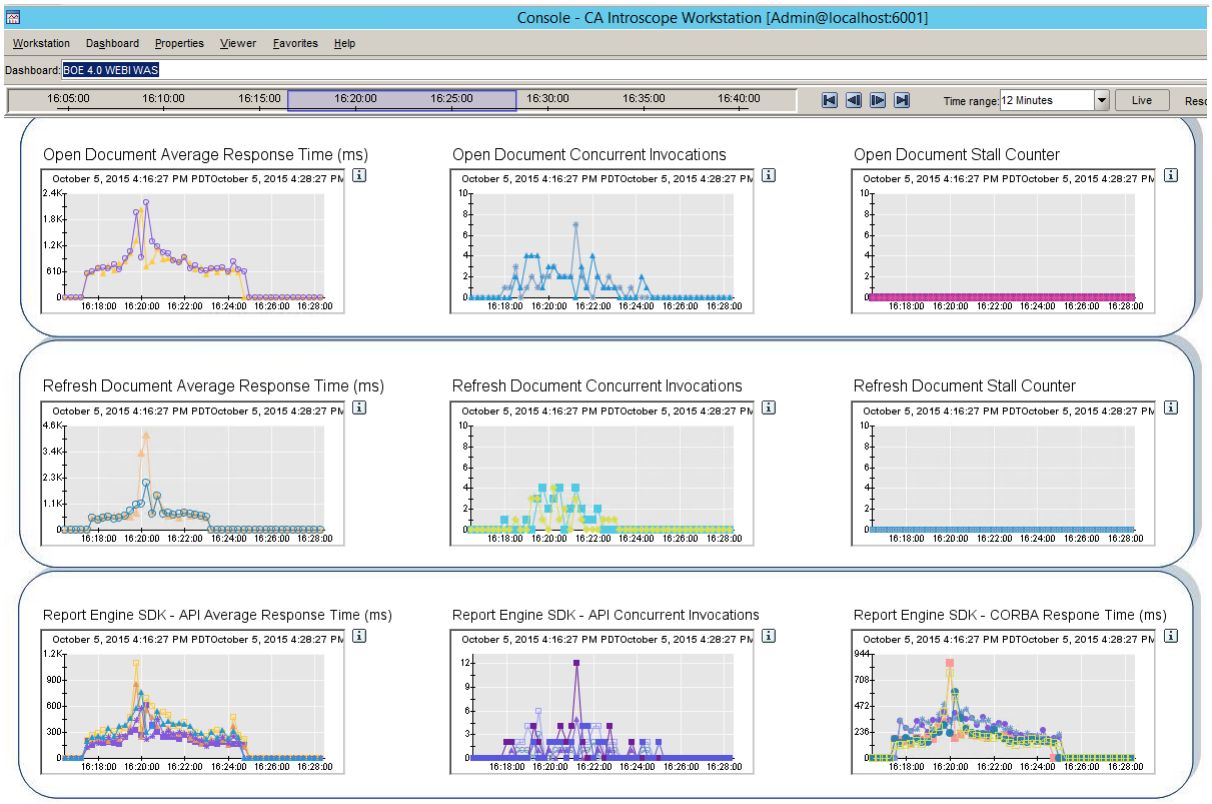
Tomcat Memory – Test 4



Tomcat Servlets – Test 4

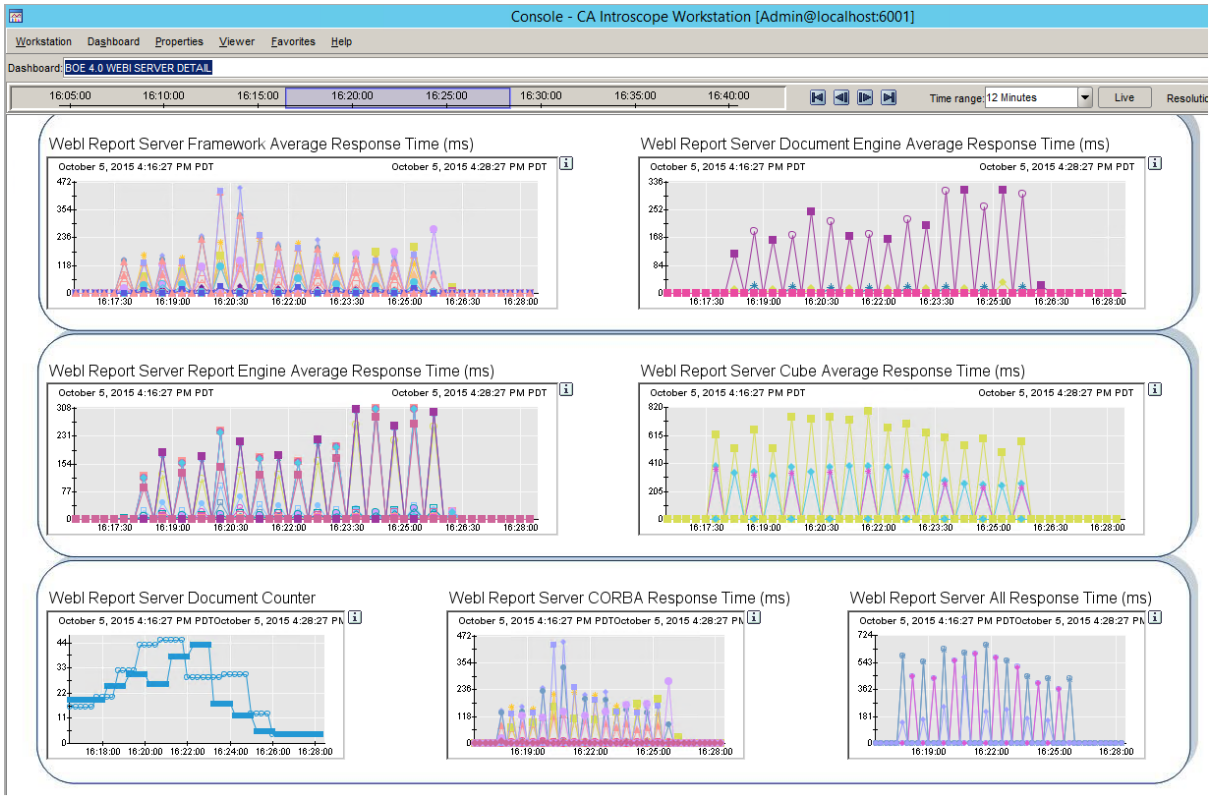


WebI WAS - Test 4



Tomcat Triage – Test 4

WebI Server Details – Test 4



WebI Server Overview – Test 4

Test 5- 150 Active Concurrent Users

After the failures in Test 3 and 4, we decided to take the time to review our environment to ensure it could handle 150 ACUs. We reviewed the system and found a few areas where we could make some changes that would help us hit our 150 ACU target. Below are the changes we made to the environment before running Test 5:

1. We had BI Monitoring enabled and setup to monitor quite aggressively. We turned this off to ensure it was not a performance factor.
2. We stopped the Web Application Container Server (WACS) APS and Lumira Server APS to free up resources on the BI Processing Server nodes
3. We found one Adaptive Processing Server that had logging set to High. We changed this back to Unspecified

We had also recycled the processes and did a quick warm up for the system before running this test. We manually ran a few reports to establish connections and pre-populate some of the cache files.

The details of Test 5 are as follows:

Test 5	• 60 ACU Edit / 300 Transactions / 120 Second Ramp-up • 90 ACU View / 450 Transactions / 120 Second Ramp-up
End Time	8:58 AM
Start Time	8:50 AM
Total Transactions	750
Active Concurrent Users	150
Duration	~8 Minutes

JMeter Output

Edit Workflow:

Summary Report

Name: Edit Web Intelligence Report

Comments:

Write results to file / Read from file

Filename Browse... **Log/Display Only:** Errors Successes **Configure**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	300	2086	529	5971	1360.39	0.00%	40.7/min	284.12	428989.7
Browse Folders	300	3279	1387	7057	1530.49	0.00%	40.3/min	256.36	390463.6
View Report	300	3871	1662	9983	1679.99	0.00%	39.8/min	616.03	952055.6
Edit and Refresh	300	5307	2962	10372	1577.33	0.00%	38.8/min	349.29	552429.1
Close and Logout	300	953	281	4931	700.75	0.00%	41.0/min	20.65	30911.2
TOTAL	1500	3099	281	10372	2055.81	0.00%	3.0/sec	1361.14	470969.8

For the Edit workflow, we see the following values:

Edit Webi	Number of Samples	Average	Min	Max	Std Dev
View Report	300	3871	1662	9983	1679.99
Total Transaction	**300**	**15499**	**6821**	**38314**	**6848.95**
Login	300	2089	529	5971	1360.39
Edit and Refresh	300	5307	2962	10372	1577.33
Close and Logout	300	953	281	4931	700.75
Browse Folders	300	3279	1387	7057	1530.49
Average	1500	3099	281	10372	2055.81

Unlike Test 3 and 4, this test was successful in the fact that it completed without any catastrophic failures.

Interpretation of the above data

We can make the following observations from the data:

- The Average Total Transaction time is **15.499** seconds from the Edit workflow.
- The Fastest (Min) potential workflow is **6.821** seconds
- The Slowest (Max) potential workflow is **38.314** seconds
- The Standard Deviation is high during this test at over **1 second deviation** per action.

At a glance, it appears that this test was much slower than our previous successful run of 50 ACU. We may have hit a bottleneck of some sort and we'll want to investigate this deeper. Let's get started by doing some simple comparison between Test 2 and Test 5.

Comparison of Test 2 to Test 5

Below are the results for Test 2 and Test 5 shown side by side with some limited analysis run against them.

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 2**	**Test 5**	**Test 2**	**Test 5**	**% Diff**	**Test 2**	**Test 5**	**%Diff**	**Test 2**	**Test 5**	**% Diff**	**Test 2**	**Test 5**	**% Diff**
Login	100	300	691	2086	202%	487	529	9%	1676	5971	256%	165.9	1360.39	720%
Browse Folders	100	300	1682	3279	95%	1351	1387	3%	2596	7057	172%	235.37	1530.49	550%
View Report	100	300	2108	3871	84%	1368	1662	21%	3139	9983	218%	408.77	1679.99	311%

Edit and Refresh	100	300	4382	5307	21%	3227	2962	9%	6501	10372	60%	686.28	1577.33	130%
Close and Logout	100	300	203	953	369%	100	281	181%	1139	4931	333%	128.54	700.75	445%
Total Transaction	**100**	**300**	**9066**	**15496**	**71%**	**6533**	**6821**	4%	**15051**	**38314**	**155%**	**1624.86**	**6848.95**	**322%**
Total	500	1500	1813	3099	71%	100	281	181%	6501	10372	60%	1502.95	2055.81	37%

We can see that almost everything ran slower in Test 5 when compared to Test 2. If we compare the Average Total Transaction time:

Test 2	**9.066** seconds
Test 5	**15.496** seconds
Difference	**6.43** seconds (71% worse performance)

It is pretty clear that we hit some sort of bottleneck that we cannot ignore. The percentage difference is much higher than the 10-15% that we would expect as a deviance between like tests so hopefully CA Introscope dashboards can show us where things went wrong.

First, we will compare the View workflow as well.

View Workflow:

Summary Report

Name: View and Refresh Webi Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	450	2853	639	6112	1403.54	0.00%	1.4/sec	578.56	429060.8
Browse Folders	450	4010	1439	8065	1449.35	0.00%	1.4/sec	520.79	390463.4
View Report	450	2900	893	7174	1145.74	0.00%	1.4/sec	323.91	239967.6
Refresh and Prompts	450	3646	1477	7798	1249.15	0.00%	1.4/sec	345.18	255694.2
Close and Logout	450	1358	334	4845	801.30	0.00%	1.4/sec	44.55	33147.1
TOTAL	2250	2953	334	8065	1532.29	0.00%	6.3/sec	1665.67	269666.6

For the View workflow, we could see the following values:

View Webi	# Samples	Average	Min	Max	Std Dev
Login	450	2853	639	6112	1403.54
Browse Folders	450	4010	1439	8065	1449.35
View Report	450	2900	893	7174	1145.74
Refresh & Prompts	450	3646	1477	7798	1249.15
Close and Logout	450	1358	334	4845	801.3
Total Transaction	450	**14767**	**4782**	**33994**	**6049.08**
Average	2250	2953	334	8065	1532.29

Explanation of the above data

From the above data, we can make the following observations:

- The Average Total Transaction time was 14.767 seconds
- The Fastest potential Transaction time could be 4.782 seconds
- The Slowest potential Transaction time could be 33.994 seconds
- The slowest action on average is the Browse Folders action at 4.010 seconds

Comparing Test 2 to Test 5

Below is a side by side comparison of the results from our Test 2 (50 ACU) and our Test 5 (150 ACU) JMeter output

	# Samples		Average			Min			Max			Std Dev		
View Webi	**Test 2**	**Test 5**	**Test 2**	**Test 5**	**% Diff**	**Test 2**	**Test 5**	**% Diff**	**Test 2**	**Test 5**	**% Diff**	**Test 2**	**Test 5**	**% Diff**
Login	150	450	719	2853	297%	489	639	31%	1806	6112	238%	181.81	1403.54	672%
Browse Folders	150	450	1673	4010	140%	1228	1439	17%	2535	8065	218%	231.14	1449.35	527%
View Report	150	450	1194	2900	143%	782	893	14%	2357	7174	204%	237.19	1145.74	383%
Refresh & Prompt	150	450	2001	3646	82%	1444	1477	2%	4157	7798	88%	371.52	1249.15	236%
Close and Logout	150	450	234	1358	480%	109	334	206%	977	4845	396%	127.55	801.3	528%
Total Transaction	**150**	**450**	**5821**	**14767**	**154%**	**4052**	**4782**	**18%**	**11832**	**33994**	**187%**	**1149.21**	**6049.08**	**426%**
Total	750	2250	1164	2953	154%	109	334	206%	4157	8065	94%	681.29	1532.29	125%

Comparing the Average Total Transaction time:

Test 2	**5.821** seconds
Test 5	**14.767** seconds
Difference	**8.946** seconds (154% worse performance)

For the View workflow, we see even worse performance than the Edit workflow, percentage wise anyways. Both sets of results are different enough that we will want to dig into the other data to try and find root cause.

CA Introscope Output:

We grabbed the same screenshots for Test 5 as we did for Test 2. This will allow us to compare the results to try and find the bottleneck that is causing Test 5 to run significantly slower on average than Test 2.

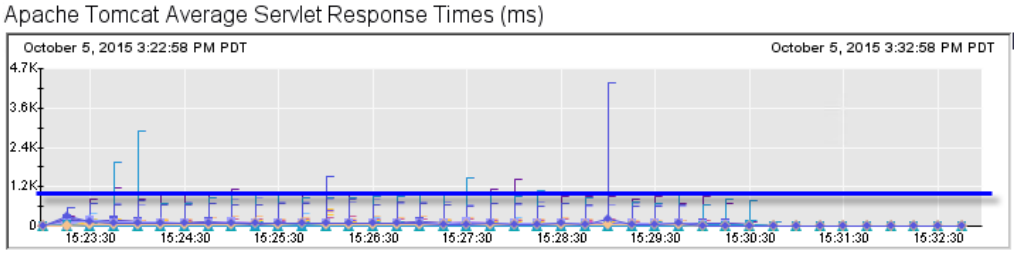
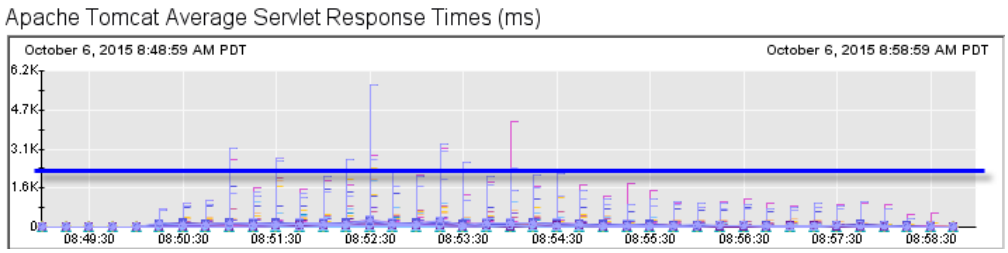
Following are the dashboards captured for Test 5.

- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

We now compare some of the key dashboard metrics for Test 2 and 5 to identify potential bottlenecks.

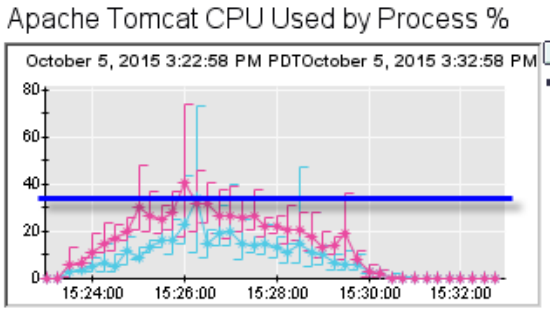
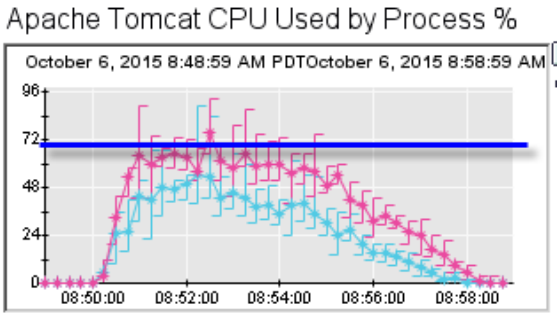
Apache Tomcat Triage

Apache Tomcat Average Servlet Response Times (ms)

Test 2 – 50 ACU	
Test 5 – 150 ACU	

Comparison Notes: Test 2 had an average response time of around 1000ms (1 sec) whereas Test 5 had a much higher response time of closer to 2000ms (2 secs) – This represents double the response time of the earlier test. This could be a result of a bottleneck found on the backend servers or delays in front of the Apache server such as the Web Server or Load Balancer. We'll have to dig into those servers as well to see what they tell us.

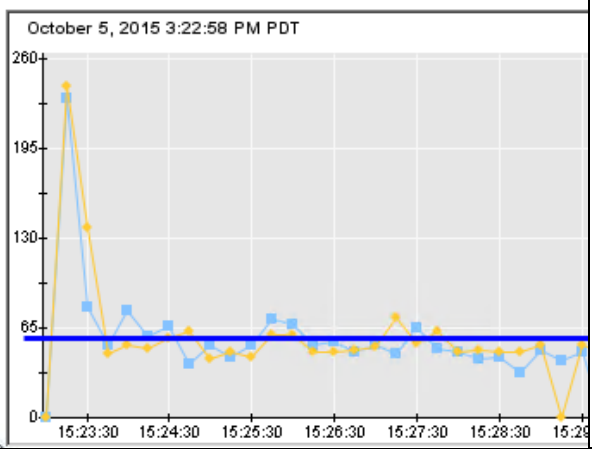
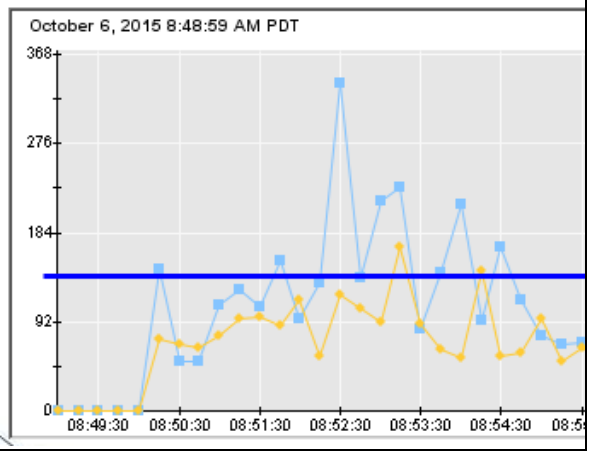
Apache Tomcat CPU Used by Process %

Test 2 – 50 ACU	Test 5 – 150 ACU
	

Comparison Notes: We can definitely see a spike in CPU usage for the Tomcat Processes between the two tests. For Test 2, the average peak was around 35% whereas Test 5, it was around 70%. This is about double the CPU for our 150 ACU test. This still isn't likely our bottleneck though as it is below the danger zone of 80%+ CPU and we did triple the load on the system so higher CPU usage is expected.

BOE 4.0 Platform Dashboard

Logon Average Response Time (ms)

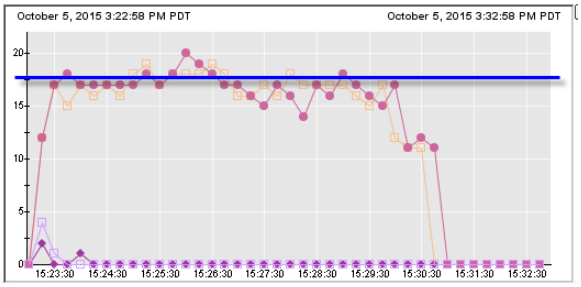
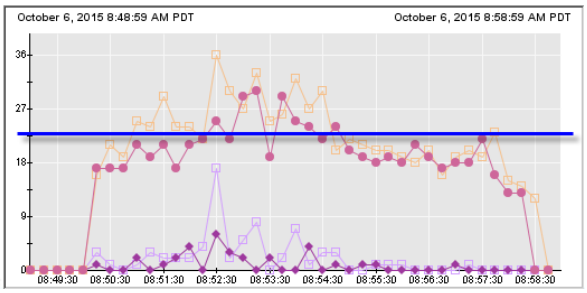
Test 2 – 50 ACU	Test 5 – 150 ACU
	

Comparison Notes: The Logon Average Response times between Test 2 and 5 showed some interesting differences. The average overall for Test 2 was around 60ms and for Test 5 it was around 150ms. This is about 2-3x slower on average. We also see a couple of spikes in average response time in the middle of the test. This would be explained by some sort of bottleneck rather than caching or connection delays like we have seen at the beginning of some tests. This data matches what we were seeing in the JMeter results for the "Login" action:

	# Samples		Average		
View Webi	**Test 2**	**Test 5**	**Test 2**	**Test 5**	**% Diff**
Login	150	450	719	2853	297%

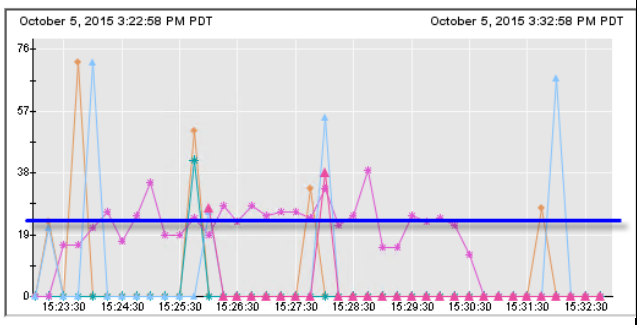
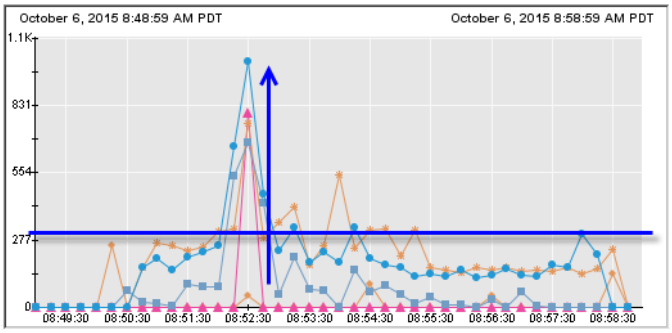
Specifically, the 297% difference between Test 2 and Test 5 for the Login action is almost 300% worse with Test 5, which is similar to what we saw in the above graphs.

Info Object Access Average Response Time (ms)

Test 2 – 50 ACU	Test 5 – 150 ACU
Info Objects Access Average Response Time (ms) 	Info Objects Access Average Response Time (ms) 

Comparison Notes: The Info Object Access Average goes up slightly in Test 5. It is around 17ms in Test 2 and about 24ms in Test 5. This is about a 40% decrease in performance so it may also be a factor.

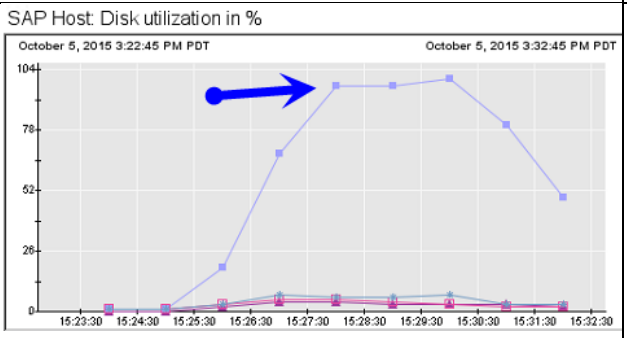
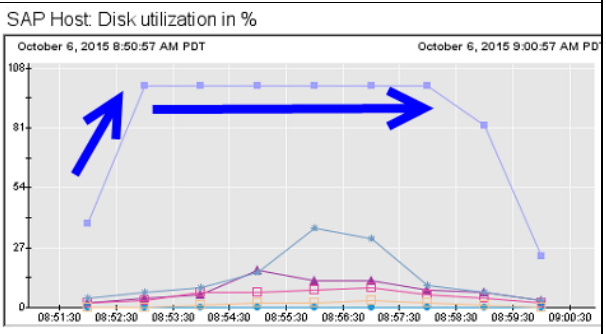
Service Access Average Response Time (ms)

Test 2 – 50 ACU	Test 5 – 150 ACU
Services Access Average Response Time (ms) 	Services Access Average Response Time (ms) 

Comparison Notes: The Service Access ART is much higher for Test 5 as well. This is an average of the general round-trip time that a server experiences when accessing a service on another server. For Test 2, most of the time it was pretty close to 0 on average with a average peak of around 21ms. Test 5 shows an average peak of around 300ms with a big spike up to 900+ms around 8:52:30 AM. If we look back at other Test 5 dashboards, we see similar spikes for other metrics around this time as well.

Host Triage

SAP Host Disk Utilization %

Test 2 – 50 ACU	Test 5 – 150 ACU
	

Comparison Notes: We noticed a bit of an outlier on our Host Triage dashboard. For Test 2, we could see one of our processing servers was showing close to 100% Disk Utilization. It grew to this slowly and then reduced back down near the end of the test. This indicates we may have been close to a bottleneck but it never created a noticeable dip in performance. It may have just been a few higher than average (Max) actions in the JMeter output. For Test 5 however, we can see it grew very quickly to 100% disk utilization and it remained there through the entire test. This could definitely be the bottleneck that raised our averages and Max values up quite a bit more and if we note the time that this happened, it was 8:52:30 AM, which is exactly when we saw spikes in most of our other graphs. This is very likely our key bottleneck and investigating this further, it was 1 of our Processing Servers that was consistently showing this spike in Disk Utilization. The other one was way lower in both tests.

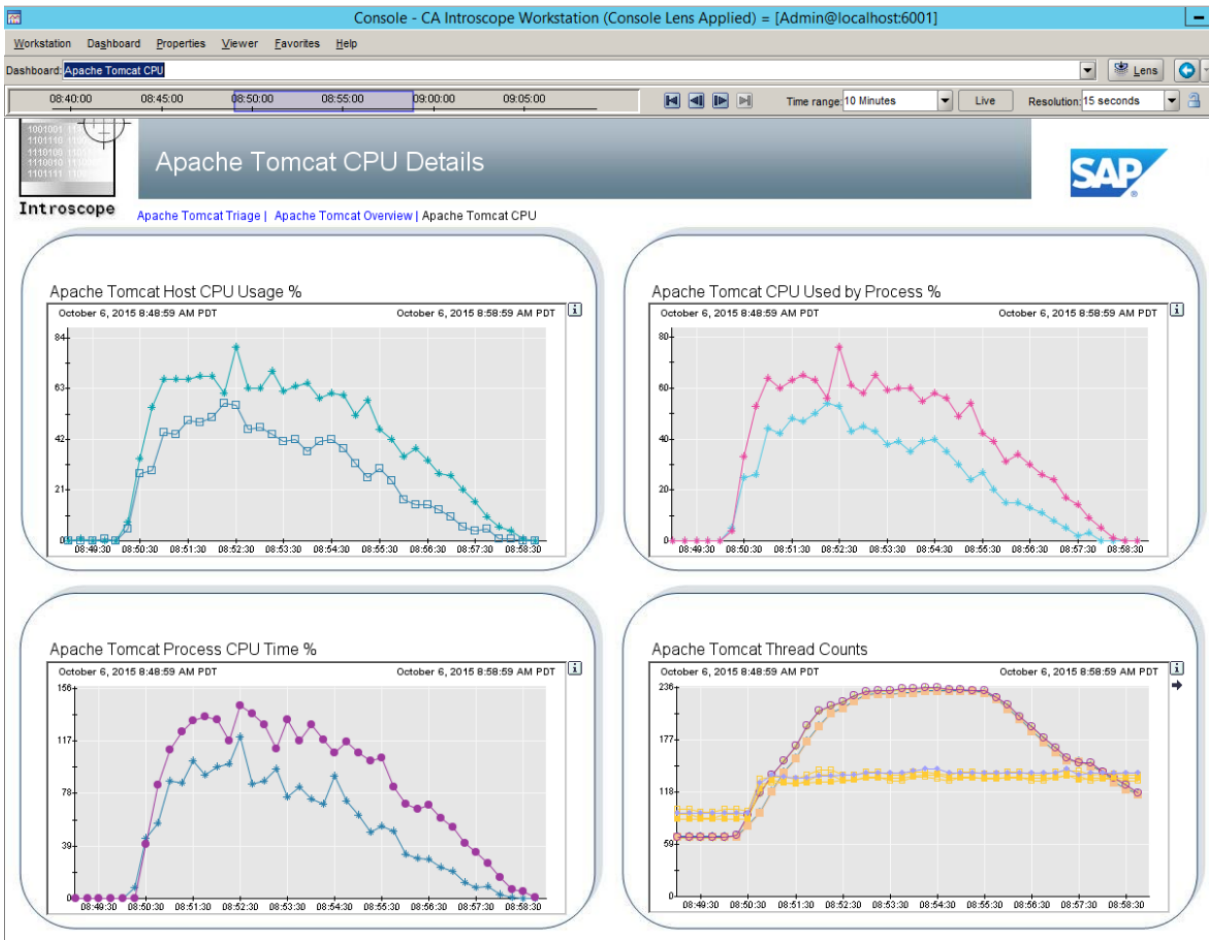
Summary of CA Introscope Output Analysis for Test 2 vs Test 5

The Introscope output definitely reflected what we saw in the JMeter output. We could see that averages across the board were much higher and percentage wise, these were in line with our JMeter percentages. It is safe to say that there was a bottleneck on one of our Processing Servers that was causing delays across the board for our tests.

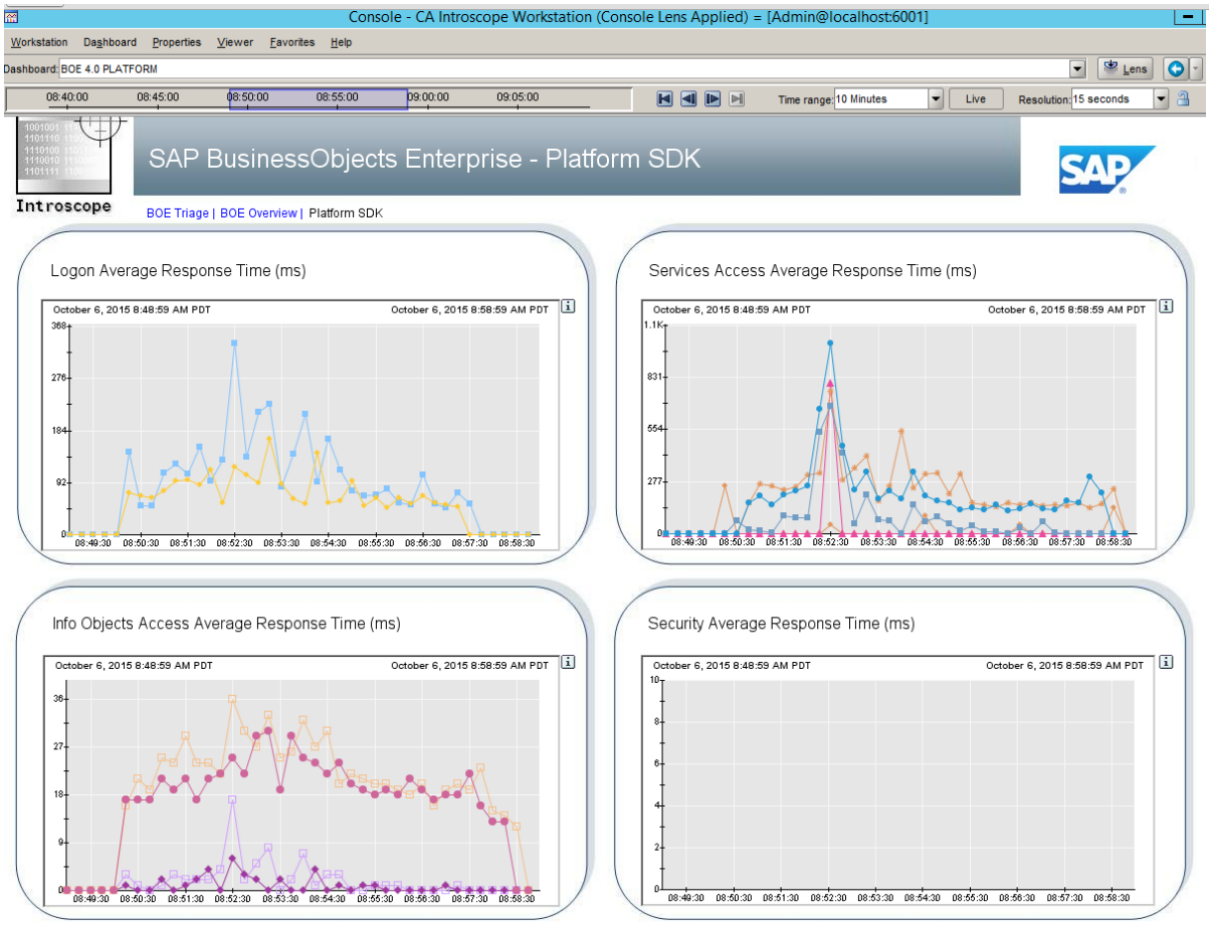
The Disk Utilization % was the most obvious bottleneck that we could find. The fact that both Processing Server were not giving similar results told us that this could be a server configuration issue that was isolated to our 1 server. The next steps were for us to investigate what could have been causing this type of delay.

The first step we took was to lower the ACU down to 100. This should give us an idea of if this was purely load related, as we went from 50 to 150 ACU pretty quickly.

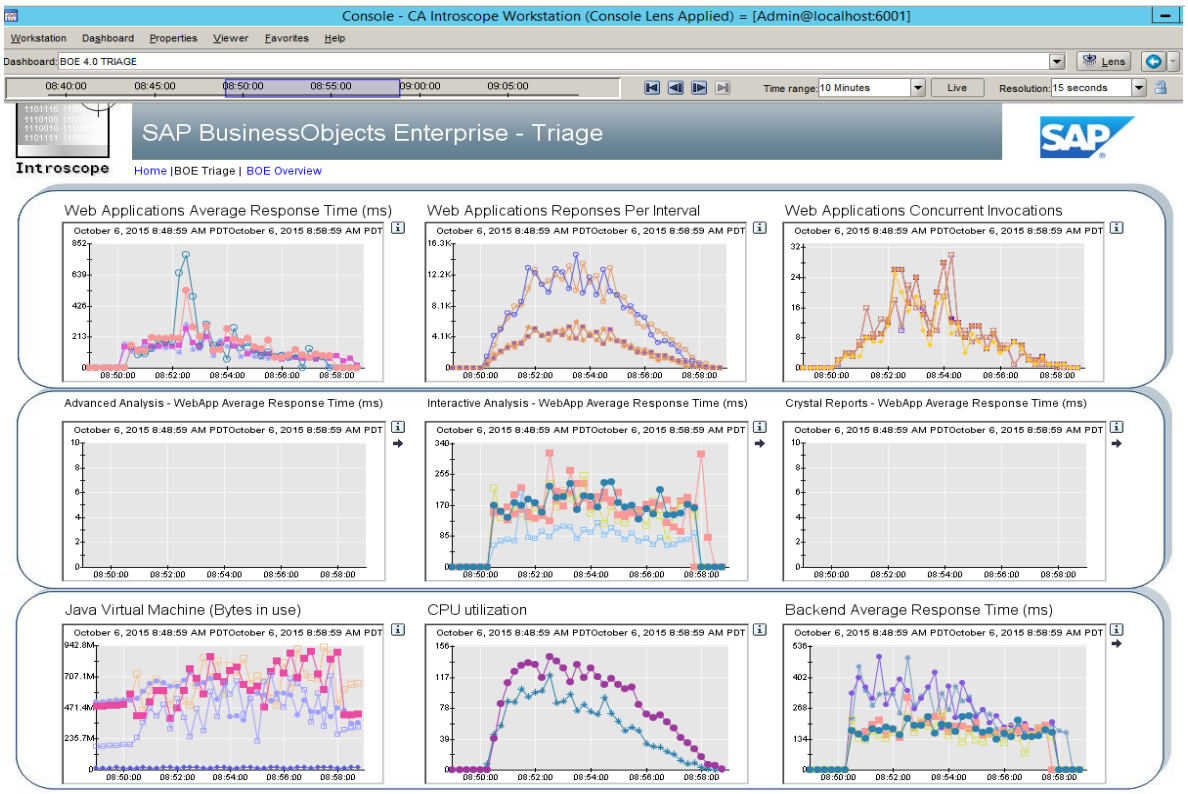
Tomcat CPU Usage – Test 5



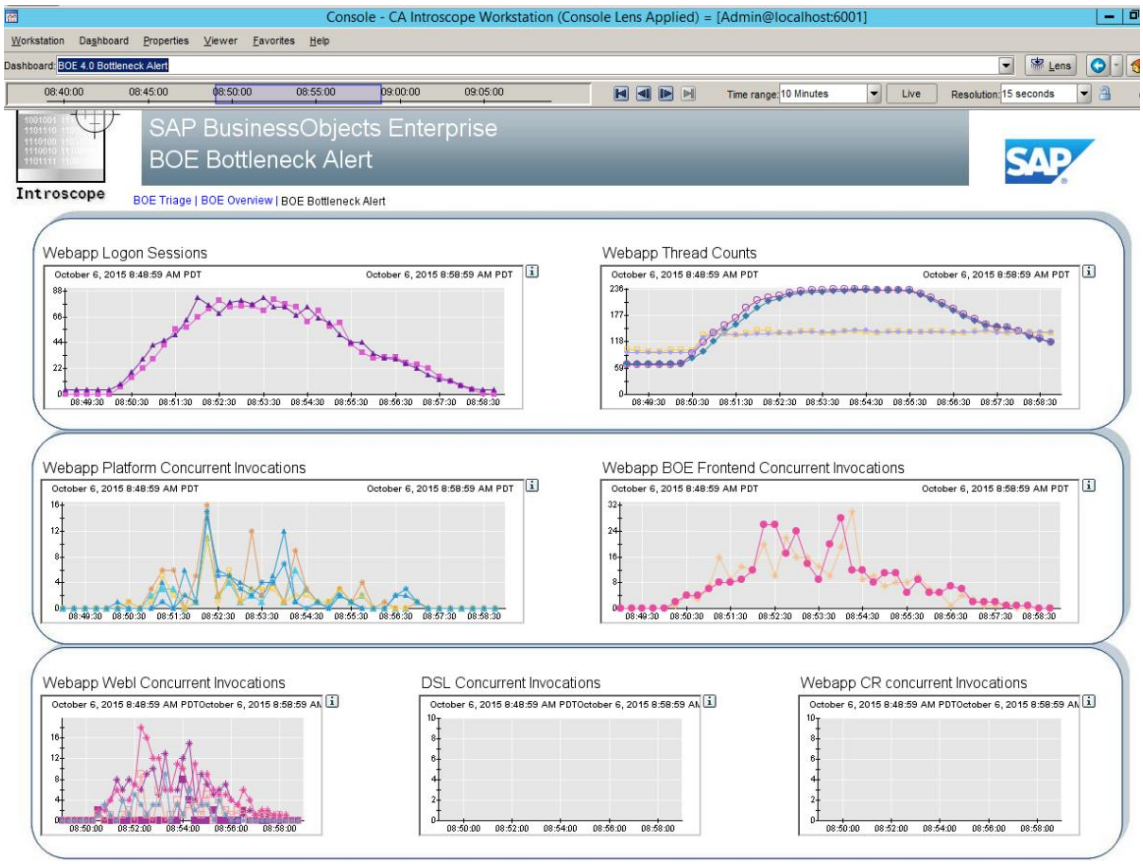
Platform Averages – Test 5



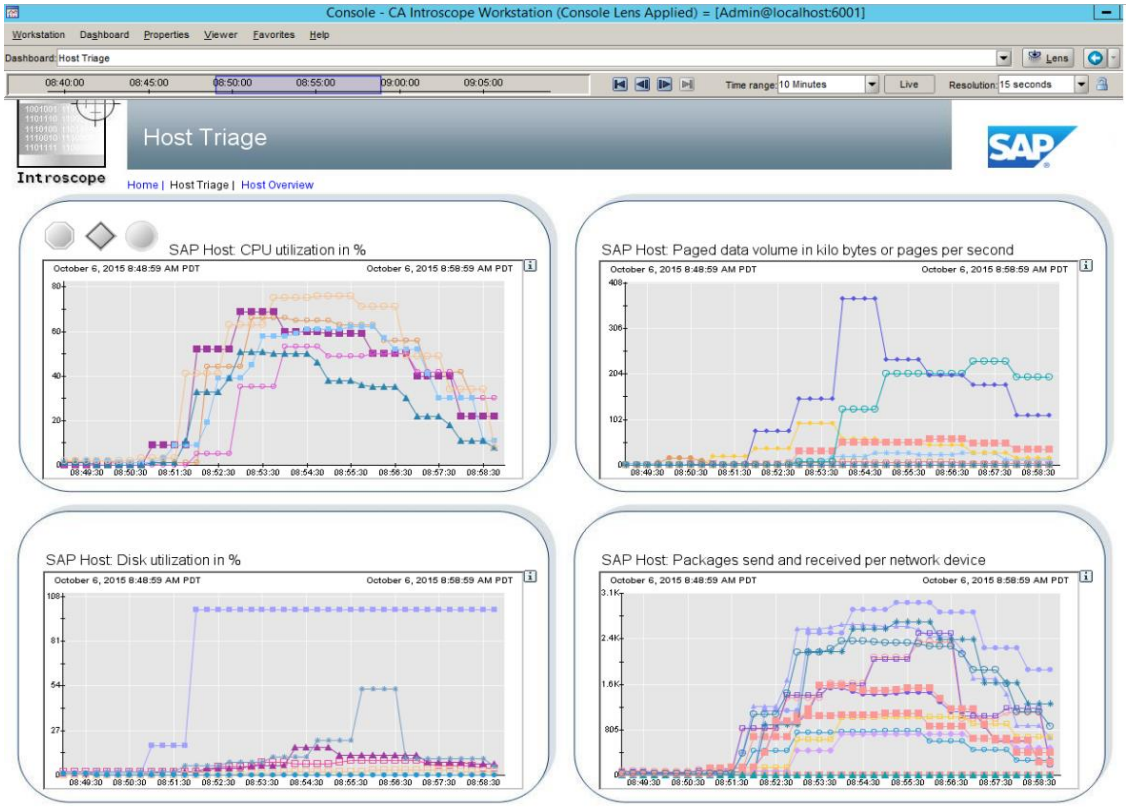
BI Triage – Test 5



Bottleneck Alert – Test 5

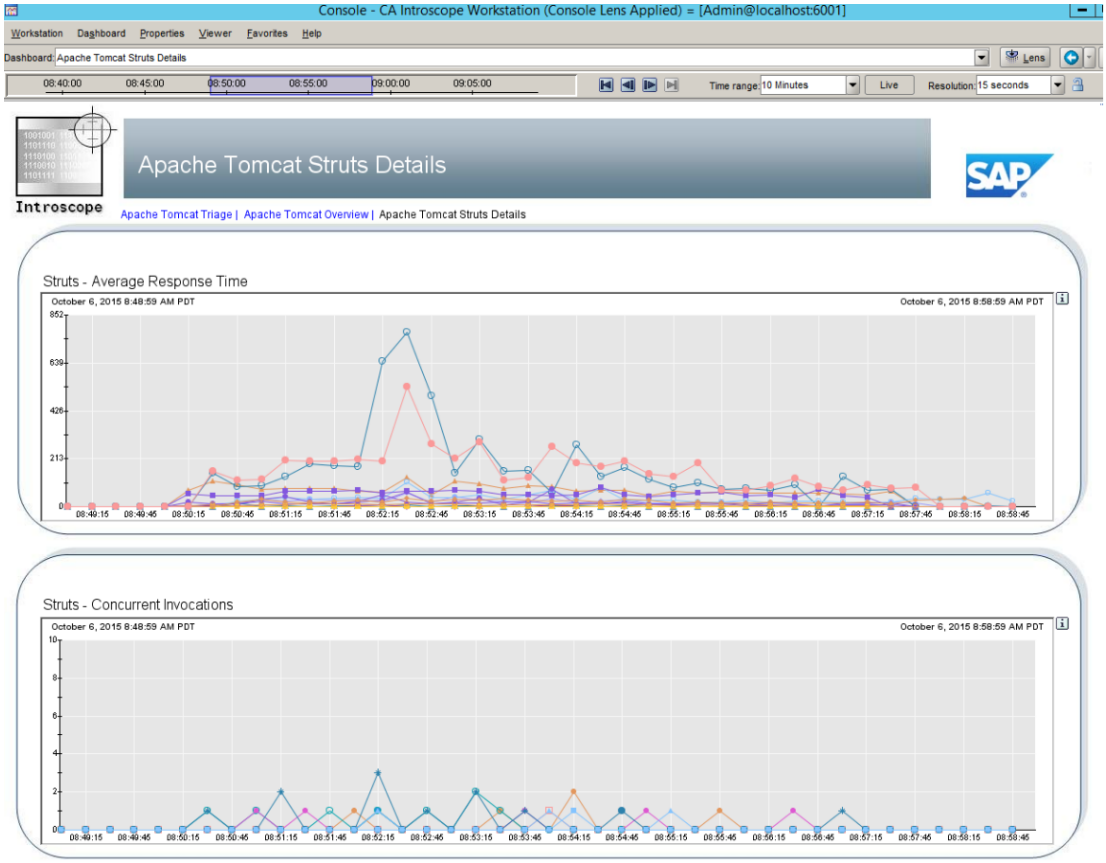


Host Triage – Test 5

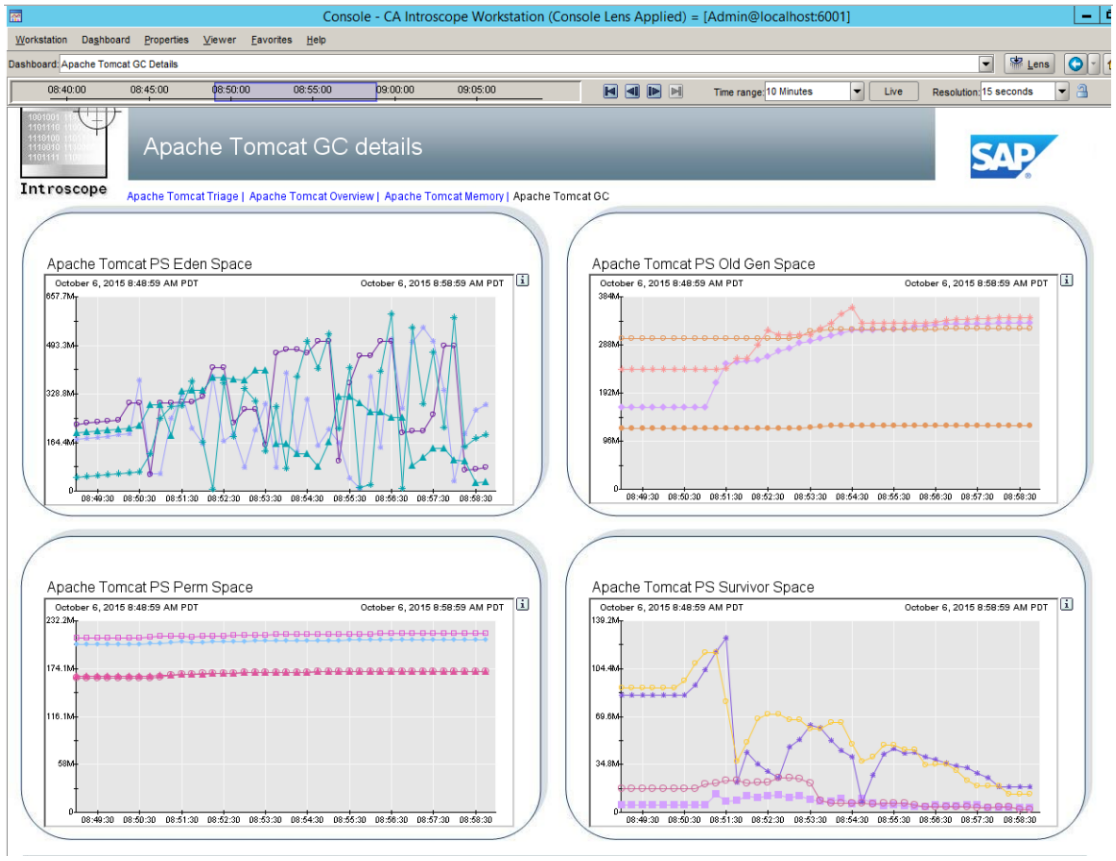


Tomcat Threads – Test 5

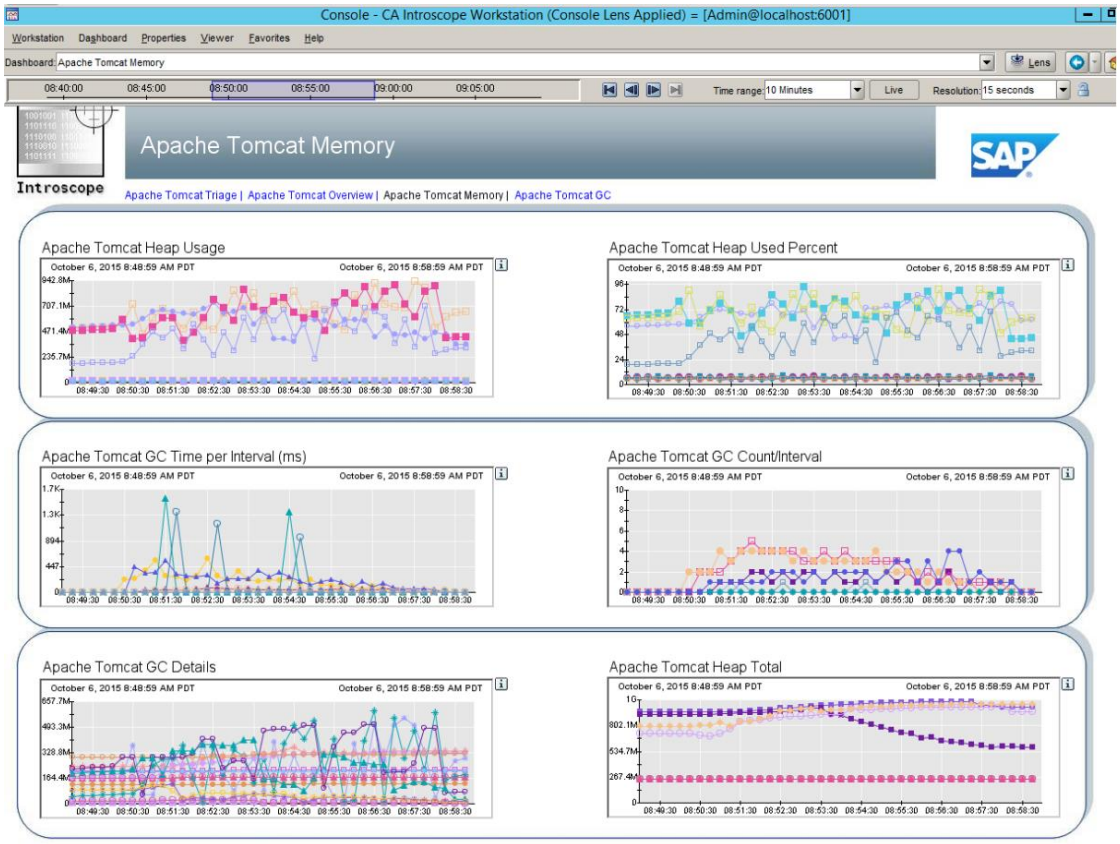
Tomcat Strut – Test 5



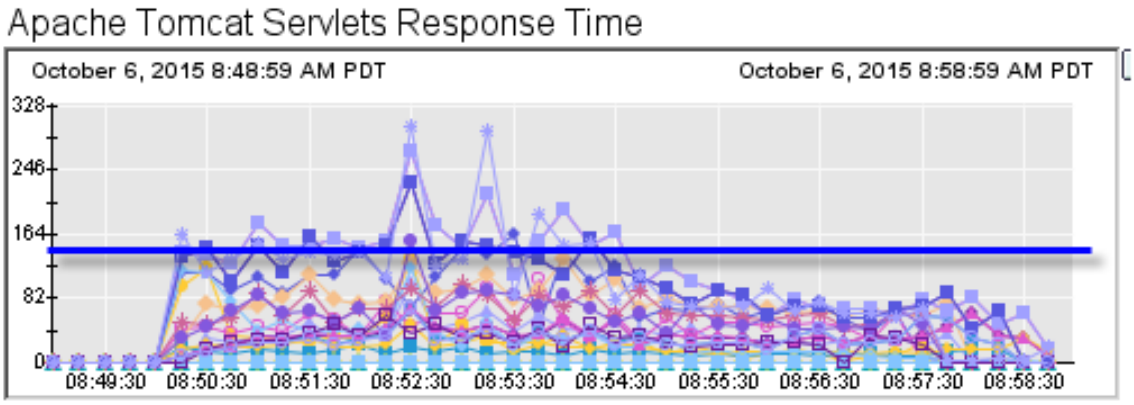
Tomcat GC Details – Test 5



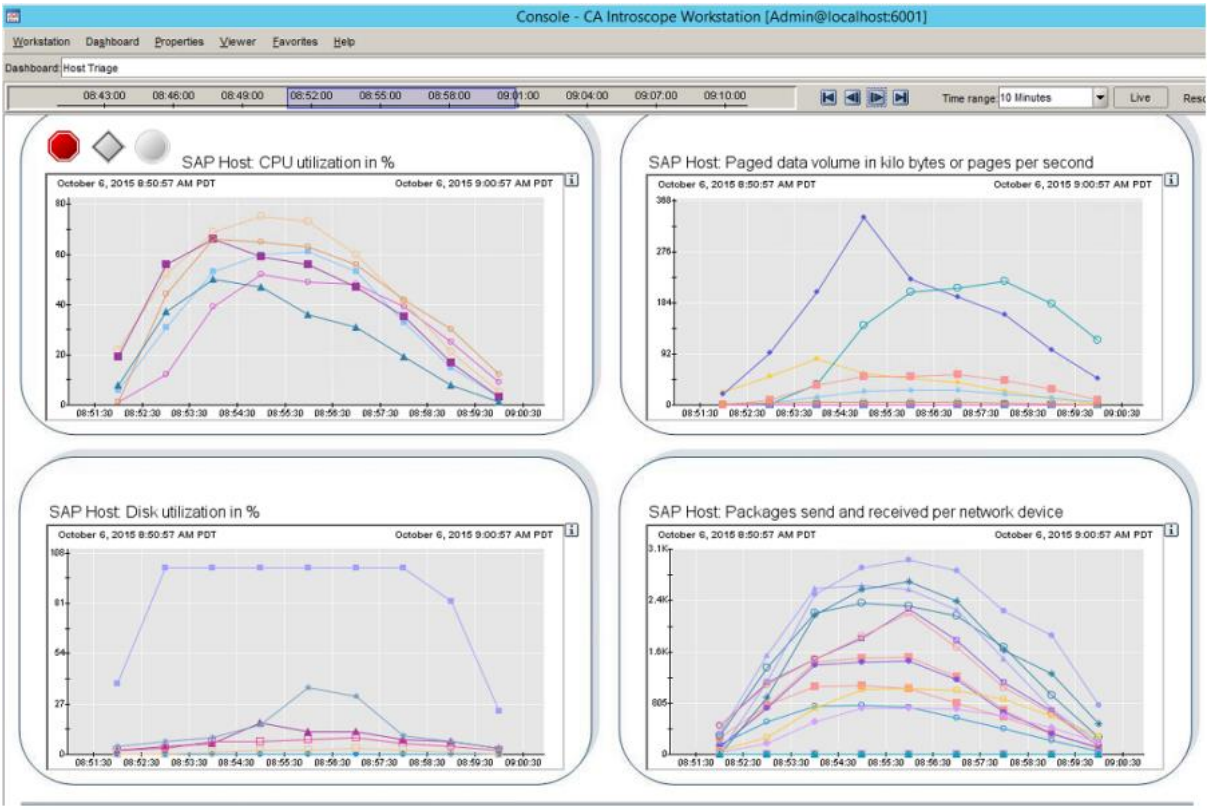
Tomcat Memory – Test 5



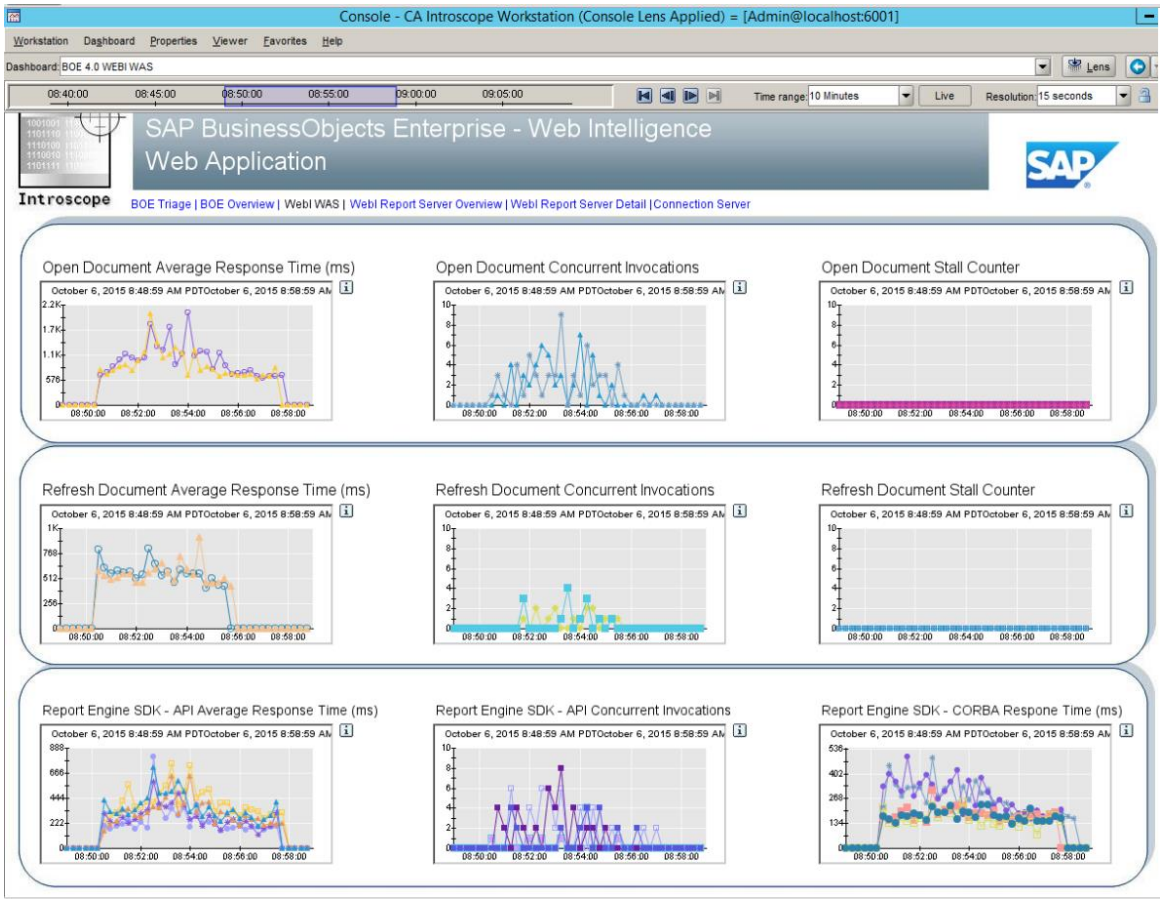
Tomcat Servlets – Test 5



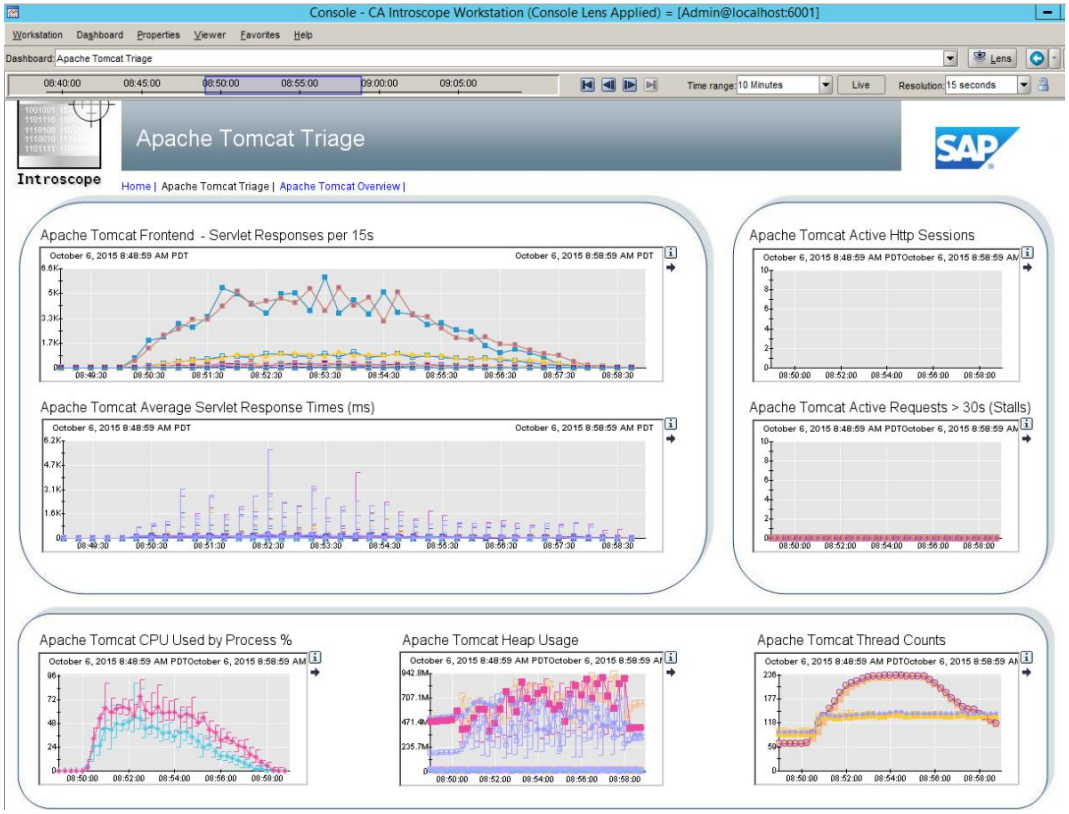
Host Triage – Test 5 2nd



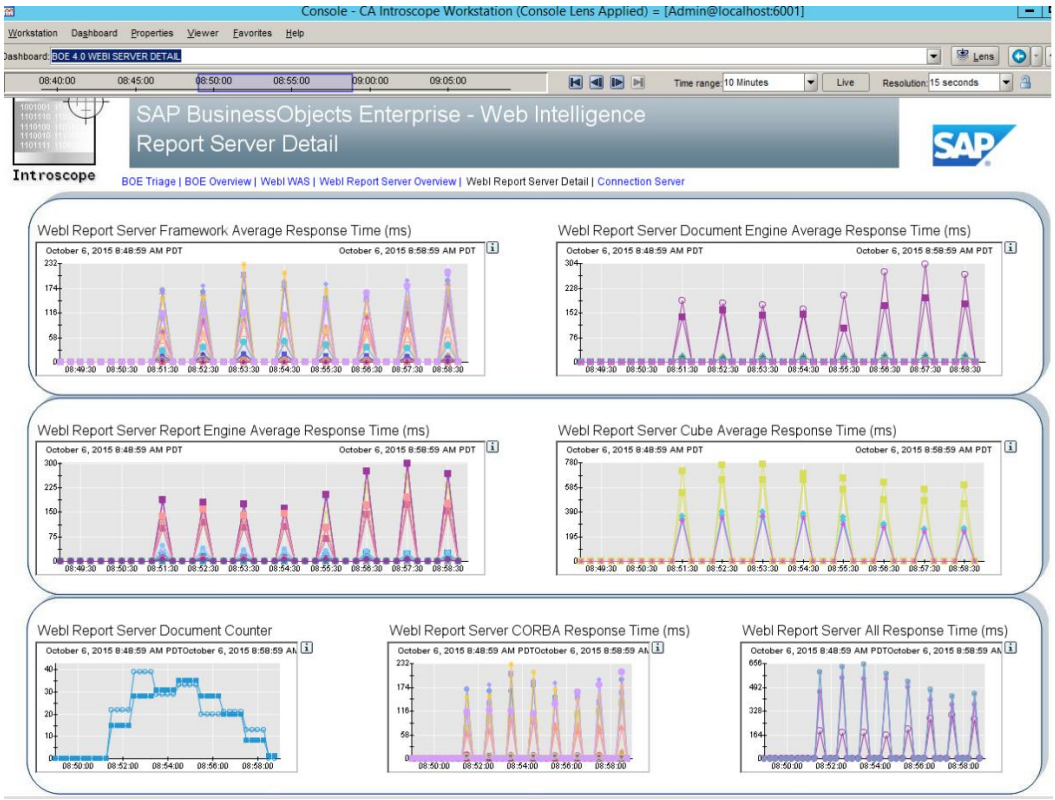
WebI WAS – Test 5



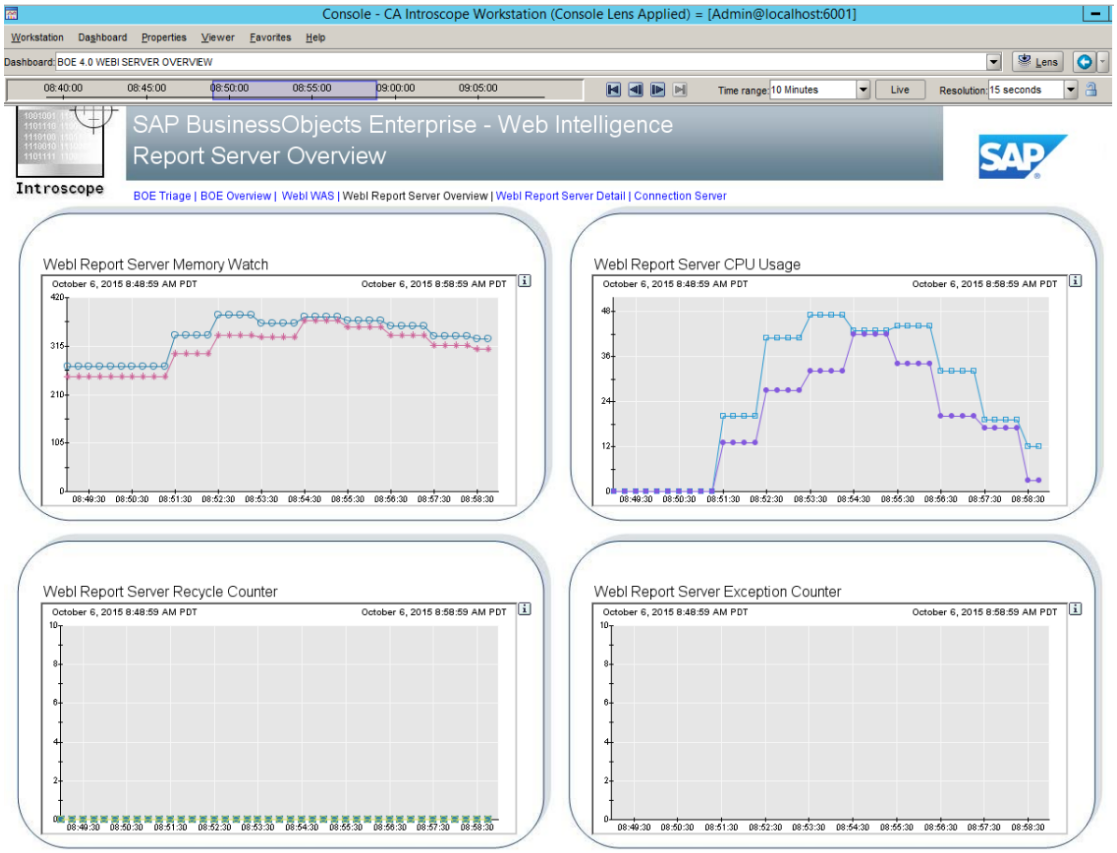
Tomcat Triage – Test 5



WebI Server Details – Test 5



WebI Server Overview - Test 5



Test 6- 100 Active Concurrent Users

For Test 6, we have reduced the number of Active Concurrent Users (ACUs) to 100 and increased the ramp-up time to reduce heavy load on the system.

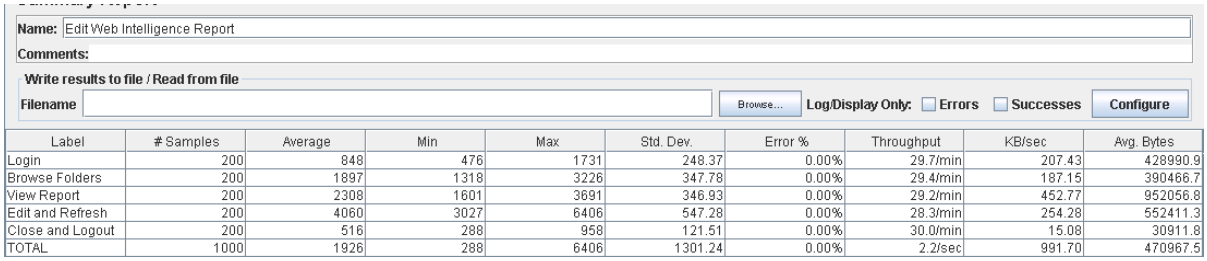
The details for Test 6 are as follows:

Test 6	**40 ACU Edit / 200 Transactions / 120 Second Ramp-up** **60 ACU View / 300 Transactions / 240 Second Ramp-up**
Active Concurrent Users	100
Total Transactions	500
Start Time	8:40 AM
End Time	8:47 AM
Duration	~7 Minutes

Note: for this test, we also increased the Ramp-up time for the View actions to 240 seconds. This should give us a more gradual ramp-up period.

JMeter Output

Edit Workflow:



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	200	848	476	1731	248.37	0.00%	29.7/min	207.43	428990.9
Browse Folders	200	1897	1318	3226	347.78	0.00%	29.4/min	187.15	390466.7
View Report	200	2308	1601	3691	346.93	0.00%	29.2/min	452.77	952056.8
Edit and Refresh	200	4060	3027	6406	547.28	0.00%	28.3/min	254.28	552411.3
Close and Logout	200	516	288	958	121.51	0.00%	30.0/min	15.08	30911.8
TOTAL	1000	1926	288	6406	1301.24	0.00%	2.2/sec	991.70	470967.5

For the Edit workflow, the values are as follows:

Edit Webi	**Number of Samples**	**Average**	**Min**	**Max**	**Std Dev**
Login	200	848	476	1731	248.37
Browse Folders	200	1897	1318	3226	347.78
View Report	200	2308	1601	3691	346.93
Edit and Refresh	200	4060	3027	6406	547.28
Close and Logout	200	516	288	958	121.51
Total Transaction	**200**	**9629**	**6710**	**16012**	**1611.87**
Total	1000	1926	288	6406	1301

These results look a bit more favorable at first sight. We will compare them to Test 5 a little later in this section to see what they can tell us.

Interpretation of the above data:

We make the following observations from the data:

1. The **Average** Total Transaction time is **9.629** seconds from the Edit workflow.
2. The **fastest** (Min) potential workflow is **6.710** seconds.
3. The **slowest** (Max) potential workflow was **16.012** seconds
4. The **slowest action** was the **Edit and Refresh** action as we would expect.
5. These results appear **better** than **Test 5** (150 ACU).

Comparing Test 5 to Test 6

Below are the results for Test 5 and Test 6 with some limited analysis run against them.

	Number of Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 5**	**Test 6**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**
Login	300	200	2086	848	146%	529	476	11%	5971	1731	245%	1360.39	248.37	448%
Browse Folders	300	200	3279	1897	73%	1387	1318	5%	7057	3226	119%	1530.49	347.78	340%
View Report	300	200	3871	2308	68%	1662	1601	4%	9983	3691	170%	1679.99	346.93	384%
Edit and Refresh	300	200	5307	4060	31%	2962	3027	2%	10372	6406	62%	1577.33	547.28	188%
Close and Logout	300	200	953	516	85%	281	288	2%	4931	958	415%	700.75	121.51	477%
Total Transaction	**300**	**200**	**15496**	**9629**	61%	**6821**	**6710**	**2%**	**38314**	**16012**	139%	**6848.95**	**1611.87**	325%
Total	1500	1000	3099	1926	61%	281	288	2%	10372	6406	62%	2055.81	1301	58%

Here is an image of the above table with some more advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	Test 5	Test 6	Test 5	Test 6	% Diff	Test 5	Test 6	%Diff	Test 5	Test 6	% Diff	Test 5	Test 6	% Diff
Login	300	200	2086	848	146%	529	476	11%	5971	1731	245%	1360.39	248.37	448%
Browse Folders	300	200	3279	1897	73%	1387	1318	5%	7057	3226	119%	1530.49	347.78	340%
View Report	300	200	3871	2308	68%	1662	1601	4%	9983	3691	170%	1679.99	346.93	384%
Edit and Refresh	300	200	5307	4060	31%	2962	3027	2%	10372	6406	62%	1577.33	547.28	188%
Close and Logout	300	200	953	516	85%	281	288	2%	4931	958	415%	700.75	121.51	477%
Total Transaction	**300**	**200**	**15496**	**9629**	61%	**6821**	**6710**	**2%**	**38314**	**16012**	139%	**6848.95**	**1611.87**	325%
Total	1500	1000	3099	1926	61%	281	288	2%	10372	6406	62%	2055.81	1301	58%

We can see that Test 6 ran faster in almost metric. The reason most "% Diff" columns are Red is because the deviation is greater than 15% when comparing the results. In most cases, this is actually a good thing as it proves that we hit some sort of bottleneck with 150 ACU (Test 5) that doesn't appear to cause the same sort of delays in Test 6.

Here is a comparison of the Average Total Transaction time between Test 5 and 6.

Test 5 **15.496** seconds
Test 6 **9.629** seconds
Difference **5.867** seconds (61% Better Performance with Test 6)

On comparing Test 6 to Test 2 (50 ACU), we see that they are better aligned.

Test 2 **9.066** seconds
Test 6 **9.629** seconds
Difference **0.564** seconds (6% Worse performance for Test 6 vs Test 2. Within our 15% deviance)

It definitely looks like we hit some bottleneck somewhere between 100 ACU and 150 ACU that exponentially decreases our average runtimes across the board. Let's take a look at the View workflow to see if we get similar results.

View Workflow:

For the View workflow, the values are as follows:

View Web Intelligence	Number of Samples	Average	Min	Max	Std Dev
Login	300	915	499	1875	254.66
Browse Folders	300	1903	1310	3076	349.67
View Report	300	1361	797	2483	271.78
Refresh & Prompts	300	2012	1386	3957	331.35
Close and Logout	300	564	329	1397	146.57
Total Transaction	**300**	**6755**	**4321**	**12788**	**1354.03**
Average	1500	1351	329	3957	623.32

Interpretation of the above data:

From the above data, we can make the following observations:

1. The **Average** Total Transaction time is **6.755** seconds

2. The **Fastest** potential Transaction time is approximately **4.321** seconds
3. The **Slowest** potential Transaction time is approximately **12.788** seconds
4. The **slowest action** on average is the **Refresh & Prompts** action at **3.957** seconds

Comparing Test 5 (150 ACU) to Test 6 (100 ACU)

Below is a side by side comparison of the results from our Test 5 (150 ACU) and our Test 6 (100 ACU) JMeter output.

	# Samples		Average			Min			Max			Std Dev		
View Webi	**Test 5**	**Test 6**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**	**Test 5**	**Test 6**	**% Diff**
Login	450	300	2853	915	212%	639	499	28%	6112	1875	226%	1403.54	254.66	451%
Browse Folders	450	300	4010	1903	111%	1439	1310	10%	8065	3076	162%	1449.35	349.67	314%
View Report	450	300	2900	1361	113%	893	797	12%	7174	2483	189%	1145.74	271.78	322%
Refresh & Prompt	450	300	3646	2012	81%	1477	1386	7%	7798	3957	97%	1249.15	331.35	277%
Close and Logout	450	300	1358	564	141%	334	329	2%	4845	1397	247%	801.3	146.57	447%
Total Transaction	**450**	**300**	**14767**	**6755**	119%	**4782**	**4321**	11%	**33994**	**12788**	166%	**6049.08**	**1354.03**	347%
Total	2250	1500	2953	1351	119%	334	329	2%	8065	3957	104%	1532.29	623.32	146%

Here is an image of the above table with more advanced formatting

	# Samples		Average			Min			Max			Std Dev		
View Webi	Test 5	Test 6	Test 5	Test 6	% Diff	Test 5	Test 6	% Diff	Test 5	Test 6	% Diff	Test 5	Test 6	% Diff
Login	450	300	2853	915	212%	639	499	28%	6112	1875	226%	1403.54	254.66	451%
Browse Folders	450	300	4010	1903	111%	1439	1310	10%	8065	3076	162%	1449.35	349.67	314%
View Report	450	300	2900	1361	113%	893	797	12%	7174	2483	189%	1145.74	271.78	322%
Refresh & Prompt	450	300	3646	2012	81%	1477	1386	7%	7798	3957	97%	1249.15	331.35	277%
Close and Logout	450	300	1358	564	141%	334	329	2%	4845	1397	247%	801.3	146.57	447%
Total Transaction	**450**	**300**	**14767**	**6755**	119%	**4782**	**4321**	11%	**33994**	**12788**	166%	**6049.08**	**1354.03**	347%
Total	2250	1500	2953	1351	119%	334	329	2%	8065	3957	104%	1532.29	623.32	146%

Comparing the Average Total Transaction time:

Test 5	**14.767**	seconds
Test 6	**6.755**	seconds
Difference	**8.012**	seconds (118% Better Performance)

We can see that Test 6 ran much faster than Test 5. This further confirms our theory that we hit some sort of bottleneck when running with 150 ACU with a shorter ramp-up time for View requests.

If we compare the Test 2 (50 ACU) results with Test 6, we see that the results are close.

Test 2	**5.831**	seconds
Test 6	**6.755**	seconds
Difference	**-0.934**	seconds (Test 2 was almost a full second faster than Test 6 but was within our 15% deviance guideline)

CA Introscope Output:

We have collected the same 14 dashboard screenshots as we had with the previous tests. We won't do a comparison of these in this section as the results will likely be very similar to the comparison that we did between Test 2 and Test 5. Instead, we want to rerun the 150 ACU and 100 ACU tests again to see if the results are consistent and repeatable on-demand.

We have the following dashboards captured for Test 5. Following is the list of screenshots taken.

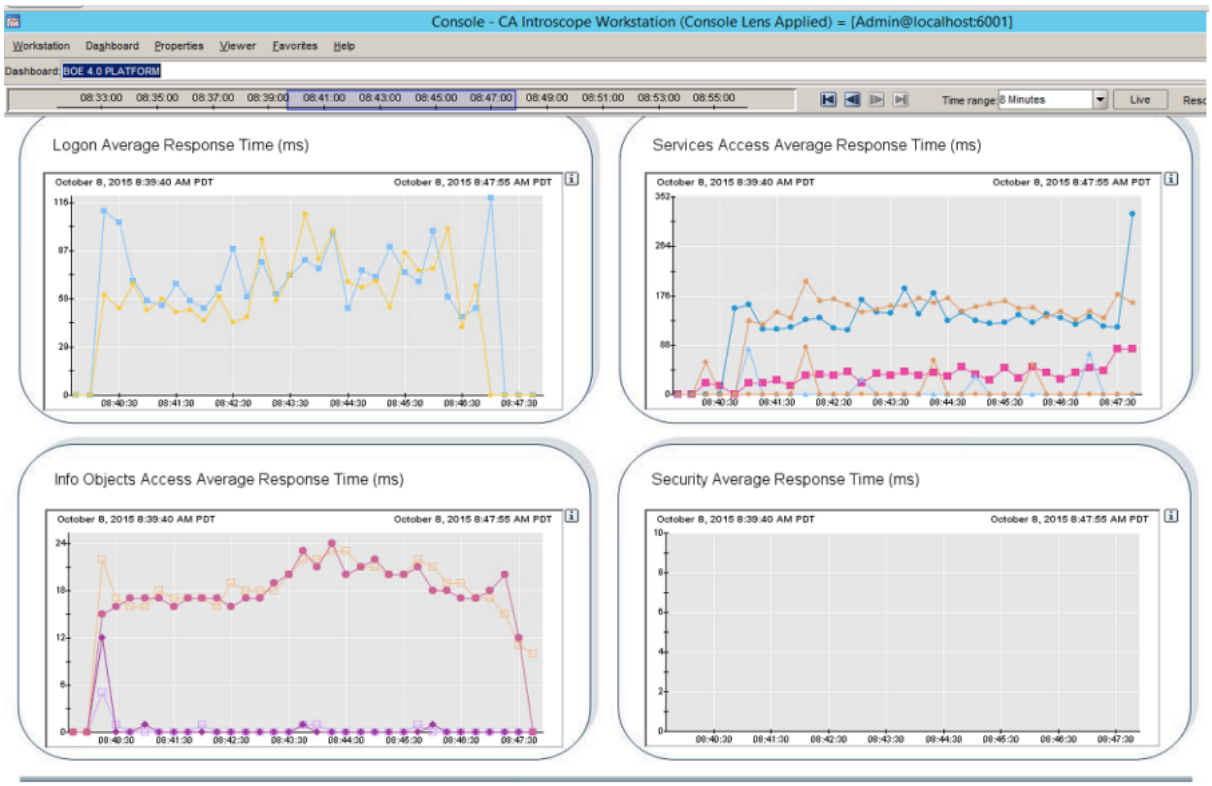
- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

Resources for Test 6 - 100 ACU

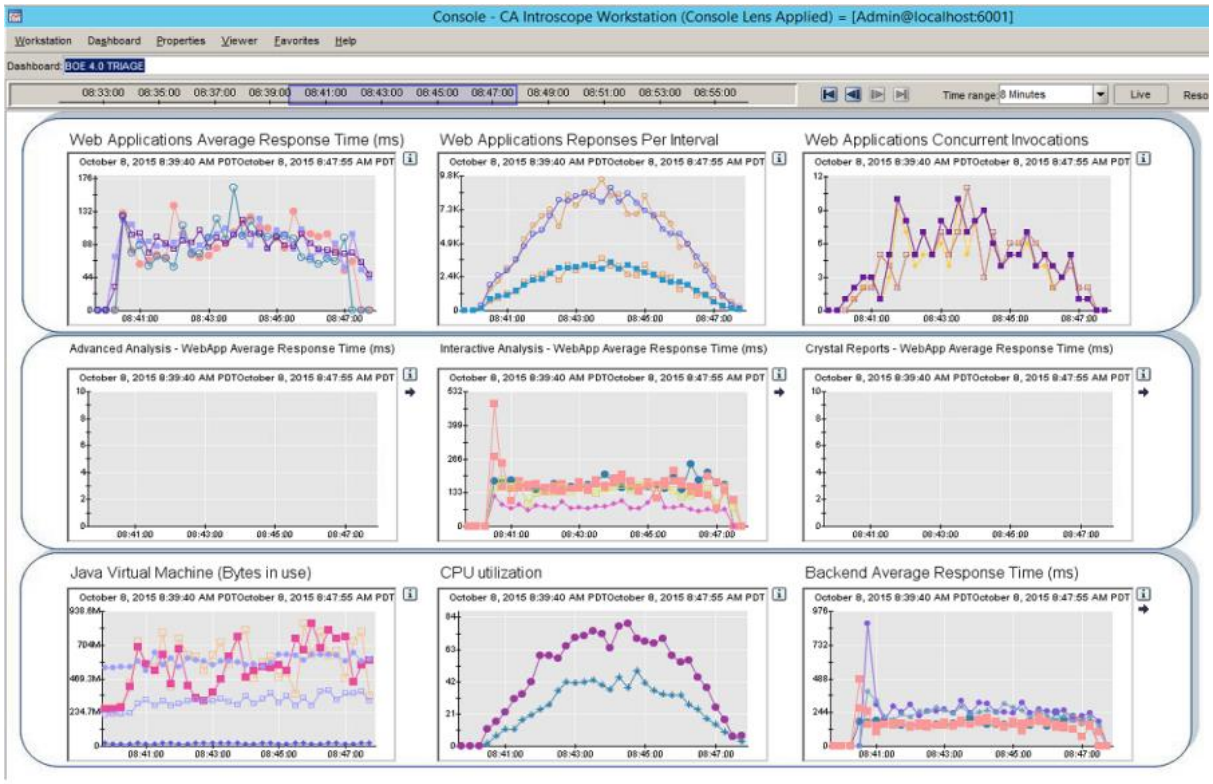
This is a list of CA Introscope dashboard screenshots that were captured during our test.

Tomcat CPU Usage – 100 ACU

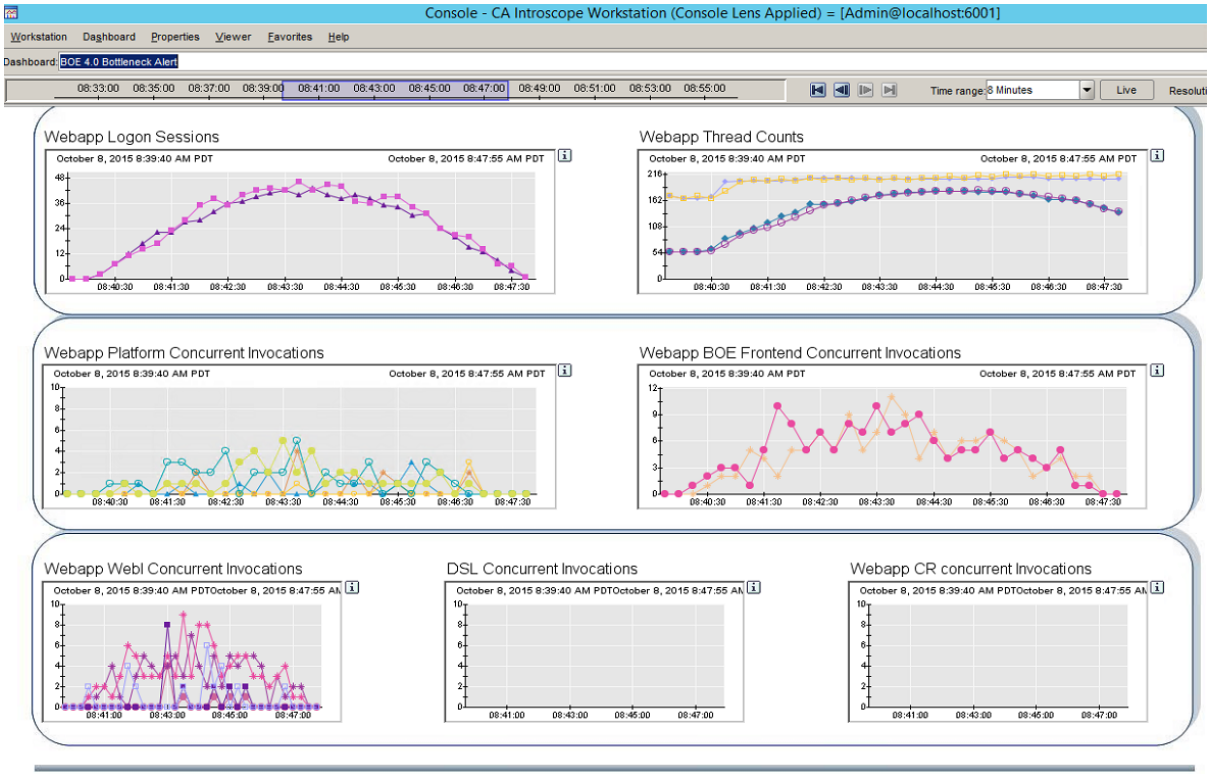
Platform Averages – 100 ACU



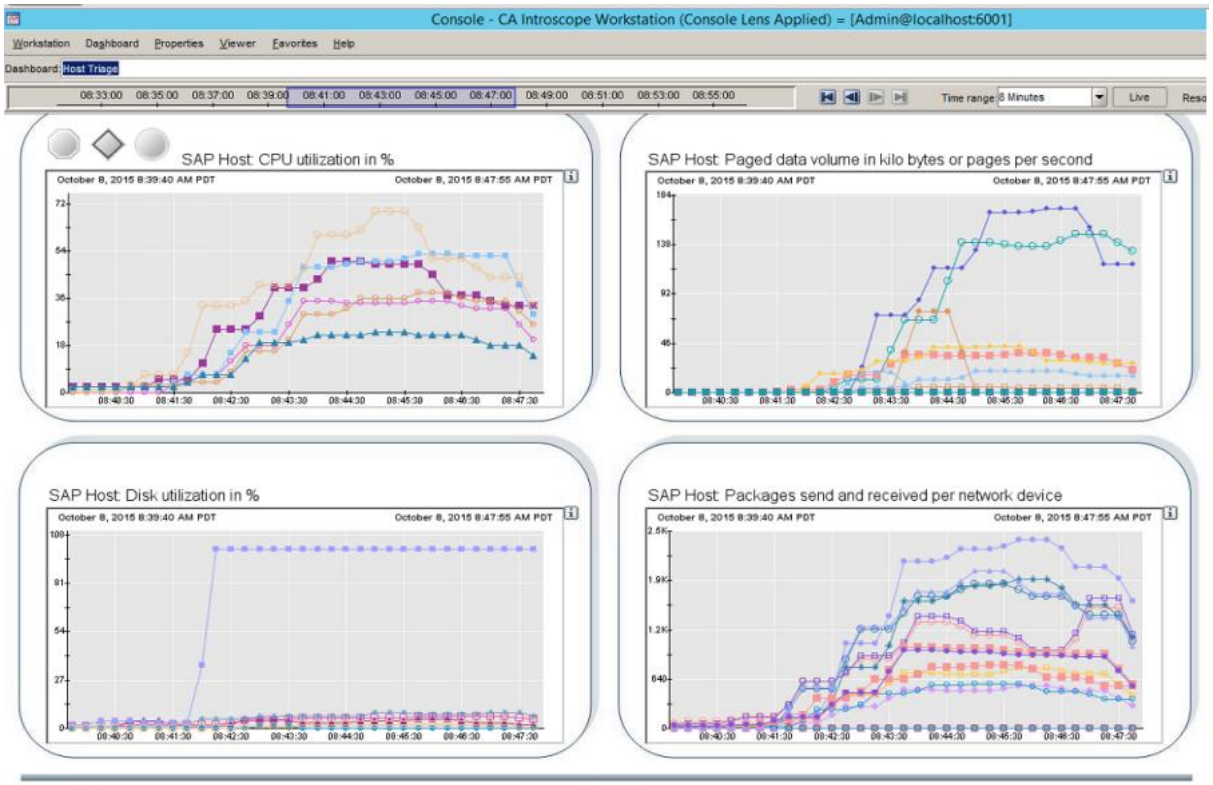
BI Triage – 100 ACU



Bottleneck Alert – 100 ACU

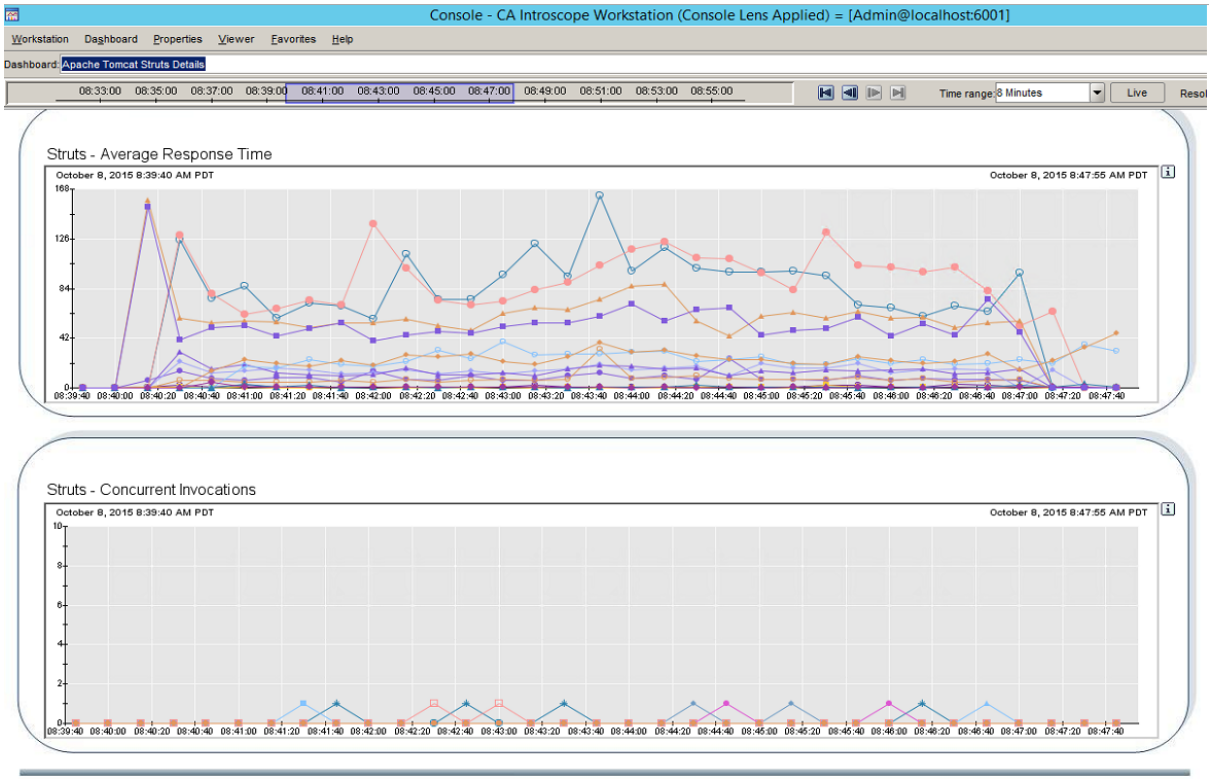


Host Triage – 100 ACU



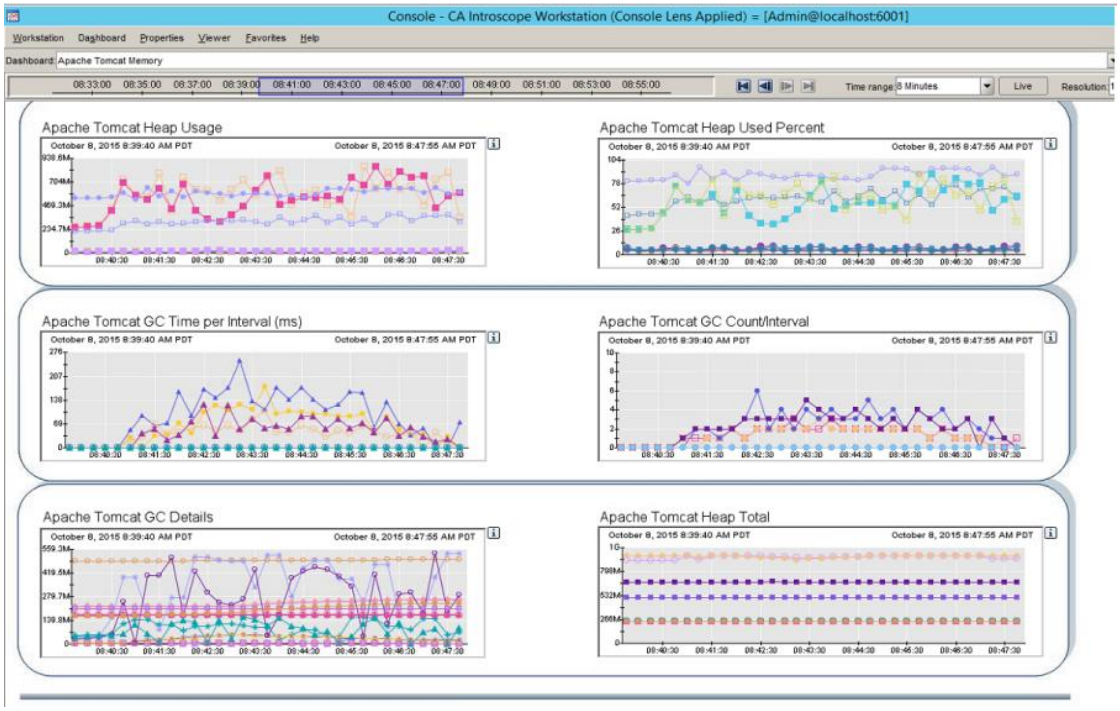
Tomcat Threads – 100 ACU

Tomcat Struts – 100 ACU

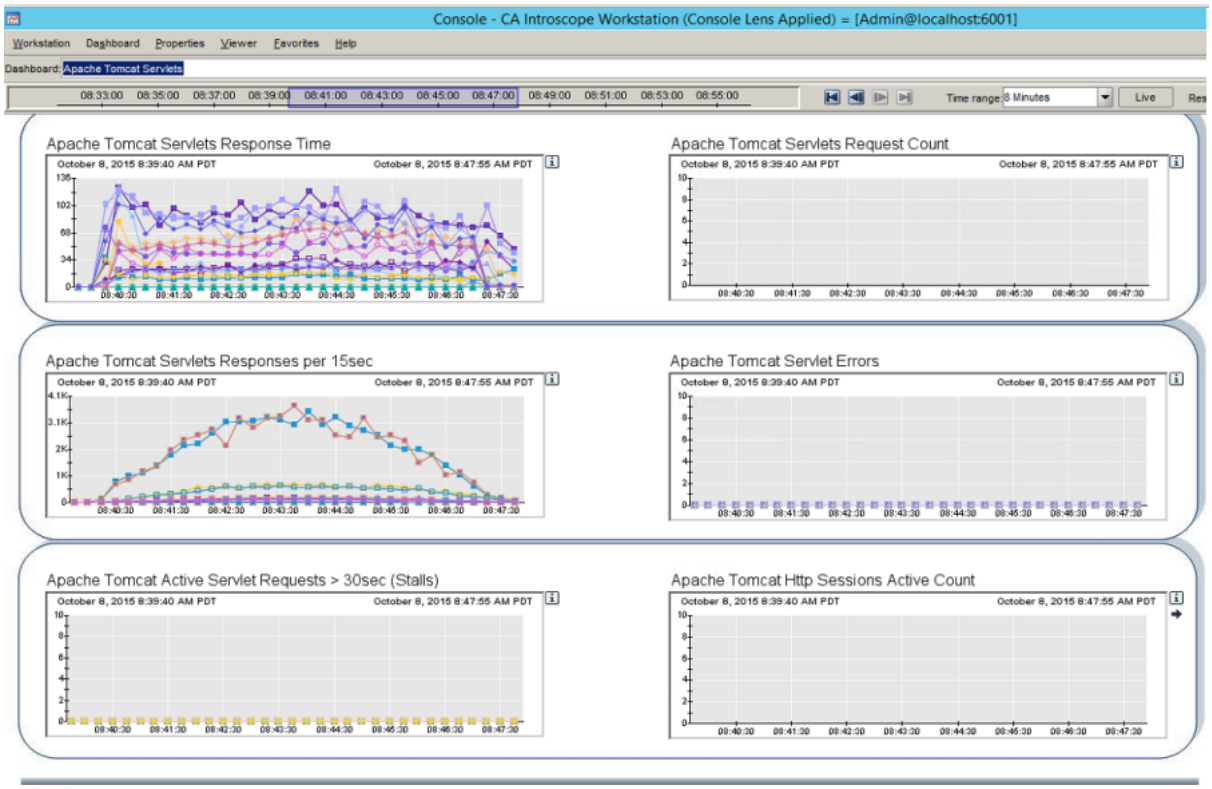


Tomcat GC Details – 100 ACU

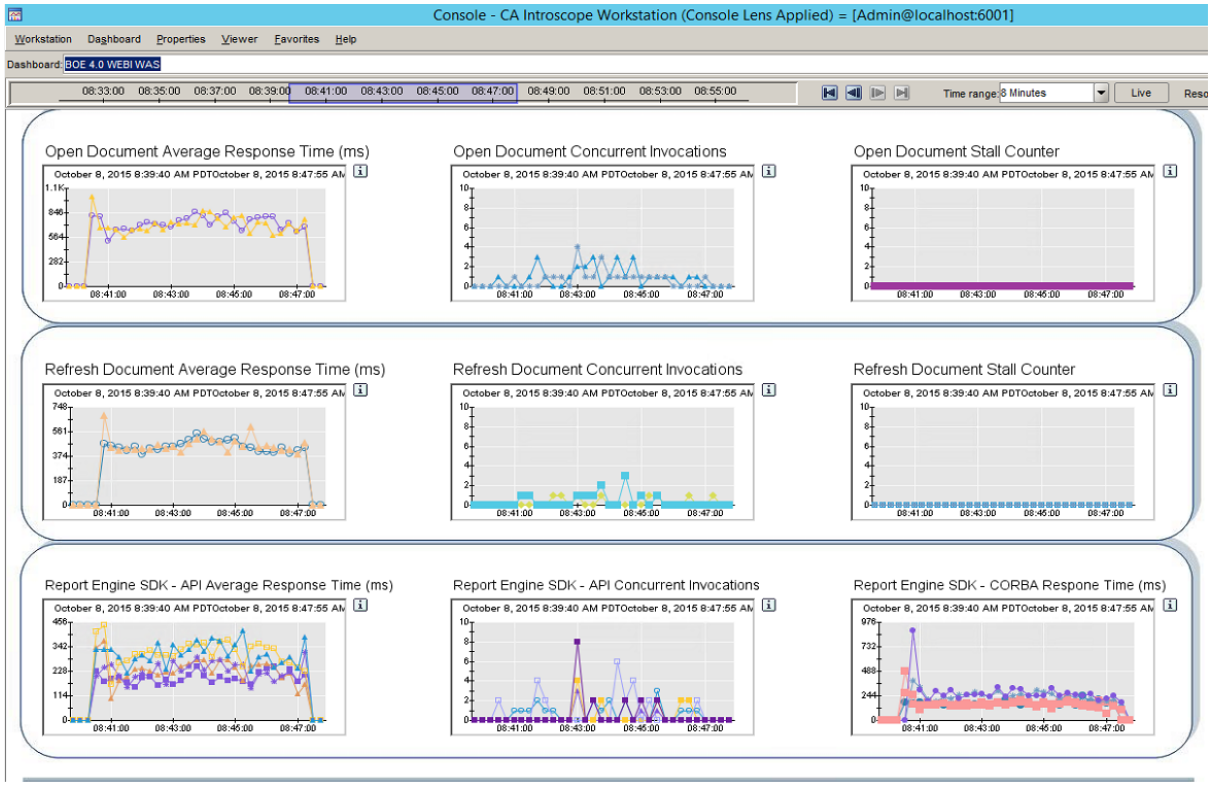
Tomcat Memory – 100 ACU



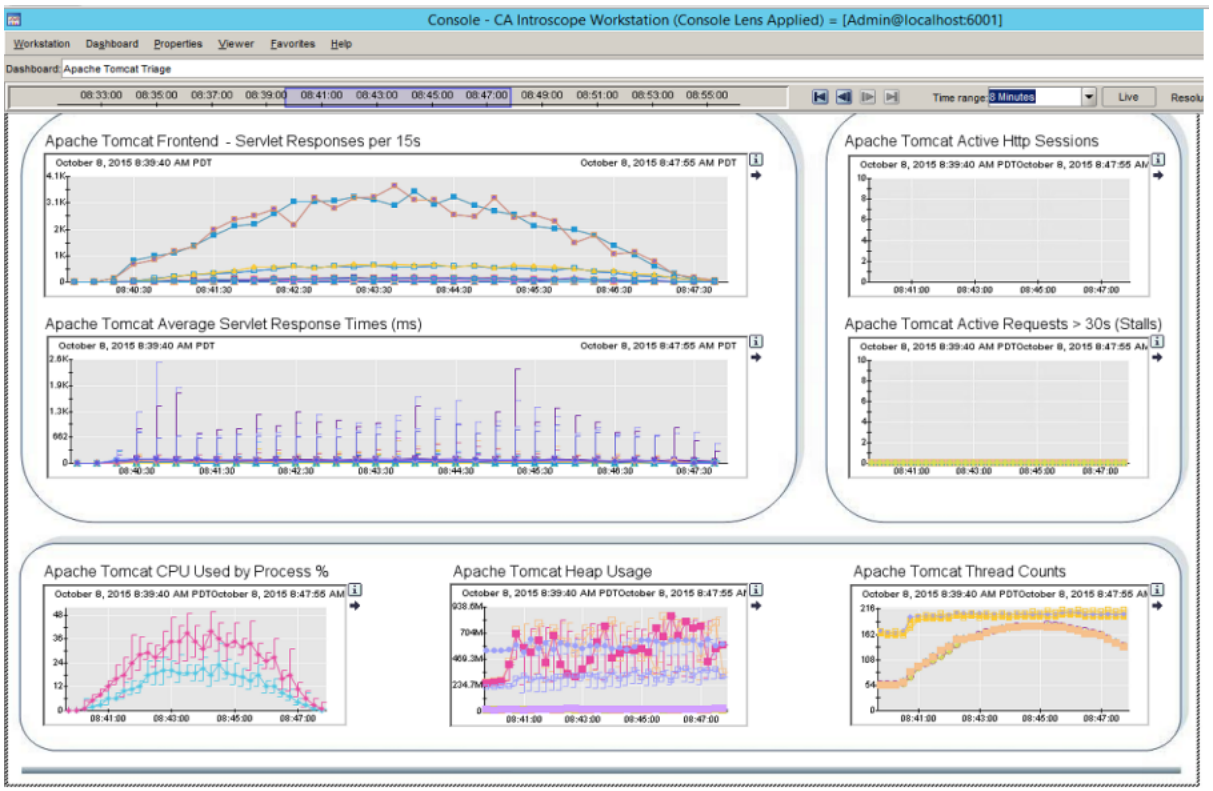
Tomcat Servlets – 100 ACU



WebI WAS – 100 ACU

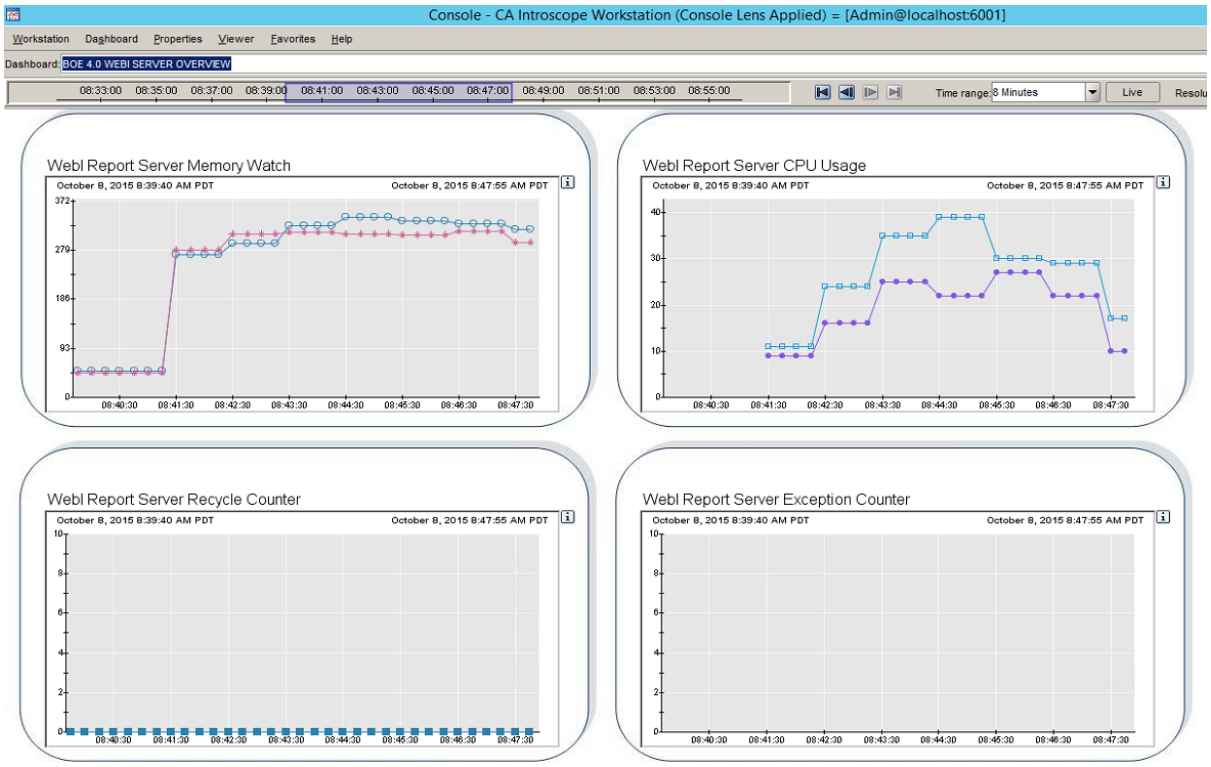


Tomcat Triage – 100 ACU



WebI Server Details – 100 ACU

WebI Server Overview – 100 ACU



Test 7 to10 - Investigating Bottleneck Root Cause

We ran a few iterations of our tests to ensure that the results were similar. We saw very similar results with subsequent runs and determined that this was not a one-off situation. The same server consistently showed 100% Disk Utilization shortly after we kicked off our tests. We began investigating the higher disk utilization by using Windows Resource Monitor during our test runs. We could see that the WIReportServer.exe was doing most of the disk I/O and this is expected, as it is our main processing server. The weird thing was that even with 100 ACU, we could see pretty close to 100% Disk Utilization on the same server.

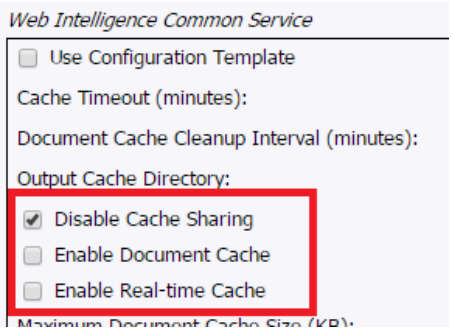
To try and limit the disk I/O, we did the following:

1. **Disabled Caching for Webi Processing Servers**

   Disabling the Caching limits the amount of disk read/writes that is carried out by the WIReportServer.exe processes. To disable the Webi Processing Servers for Each Web Intelligence Processing Server perform the following:

a. Log on to Central Management Console (CMC).
b. In the top-left of the Central Management Console window, select Servers from the dropdown.
c. Navigate to Service Categories > Web Intelligence Services.
d. Select Web Intelligence Processing Server.
e. In the Web Intelligence Common Service, uncheck the Enable Document Cache and Enable Real-time Cache options.
f. Check the Disable Cache Sharing.

When done, the options should look like the below image:



Result: After disabling caching, we saw similar results to the previous runs. It was always the same Processing Tier node that was experiencing the high Disk Utilization. This did not resolve our problem.

1. **Disabled the McAfee Antivirus On-Access scanning**

   On-Access Scanning will scan all files that are created or accessed on a system live. This is great for protection against viruses but can be quite I/O intensive in some cases. Since we were creating thousands of temp files during our test, we thought this might be part of the

problem. We asked our IT group to disable this feature for a short window so we could test this out.

Result: Re-running the test with On-Access turned off did not resolve the issue and we still experienced the spike up to 100% Disk Utilization.

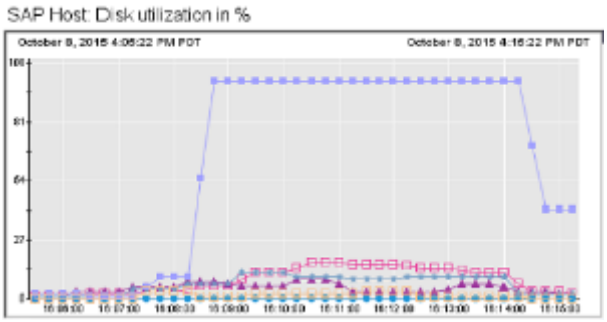
2. **Reboot all the servers in the cluster**

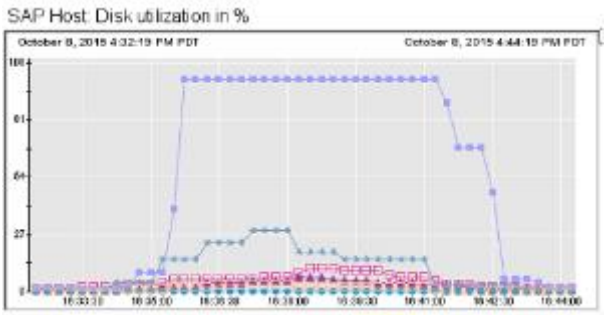
   When in doubt, reboot right! We are using a Virtualized Environment and a Windows operating system so sometimes a good old fashioned reboot is worth a shot.

   Result: In our case, it was not the solution. We saw similar results even after a reboot.

3. **Swapped around our resources on the VMWare Infrastructure**

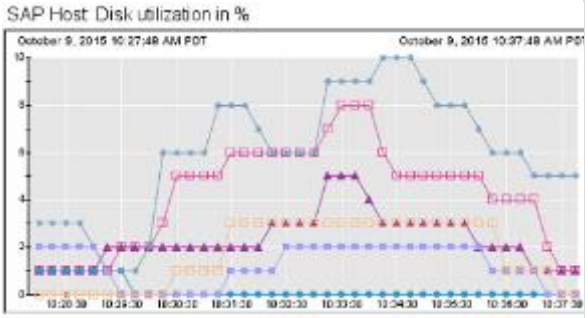
   As we pointed out in the Test 5 results, the Disk Utilization for one of our Processing Servers was showing 100% Disk utilization during the period of time where our slowdowns began. This was consistent across other tests as well and could be contributing to our Bottleneck. Below are some graphs from subsequent tests which show the consistent pattern of 100% disk utilization:

Test	Disk Utilization
Test 7 (150 ACU)	
Test 8 (100 ACU)	

Test	Disk Utilization
Test 9 (150 ACU)	

We spoke to our Infrastructure team again and he was able to find us another dedicated ESX server to add to our landscape. We moved this Processing Server on to this new server infrastructure on the backend and shuffled around some of our other resources to give us a better mix of total resources vs available resources. We then restarted all of our servers and saw much better results for this Disk Utilization value.

Results after assigning new VMWare Resources:

Test 9 (150 ACU) Before VMWare Changes	Test 10 (150 ACU) After VMWare Changes
	

As you can see, the Disk Utilization maxes out at about 10% with Test 10 (after the VMWare ESX backend changes) and the servers were showing a much more consistent result across the board. This was a great example of a hidden problem that CA Introscope can help uncover. We may not have picked up on this if we were solely looking at the OS level statistics in something like task manager.

4. **Monitoring the Apache Load Balancer**
   Much to our surprise, we found that our Load Balancer was very busy during our 150 ACU tests.

```
PuTTY
top - 08:02:32 up 13 days, 23:16,  1 user,  load average: 19.08, 9.86, 5.22
Tasks: 439 total,  11 running, 428 sleeping,   0 stopped,   0 zombie
Cpu(s): 88.8%us,  5.4%sy,  0.0%ni,  4.1%id,  0.0%wa,  0.0%hi,  1.8%si,  0.0%st
Mem:   4057068k total,  3840956k used,   216112k free,   236912k buffers
Swap:  2103292k total,    76140k used,  2027152k free,  2984896k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
10936 wwwrun    20   0  195m 2636 1456 S     4  0.1   0:01.35 httpd2-prefork
10843 wwwrun    20   0  195m 2628 1456 S     4  0.1   0:00.79 httpd2-prefork
10917 wwwrun    20   0  195m 2636 1456 R     4  0.1   0:01.07 httpd2-prefork
10941 wwwrun    20   0  195m 2756 1464 S     4  0.1   0:01.50 httpd2-prefork
10853 wwwrun    20   0  195m 2624 1460 S     3  0.1   0:00.96 httpd2-prefork
10688 wwwrun    20   0  195m 2636 1456 S     3  0.1   0:01.56 httpd2-prefork
10752 wwwrun    20   0  195m 2636 1456 S     3  0.1   0:02.11 httpd2-prefork
10871 wwwrun    20   0  195m 2492 1452 S     3  0.1   0:01.69 httpd2-prefork
10693 wwwrun    20   0  195m 2636 1456 S     3  0.1   0:01.66 httpd2-prefork
10726 wwwrun    20   0  195m 2636 1428 R     3  0.1   0:02.44 httpd2-prefork
10783 wwwrun    20   0  195m 2636 1428 S     3  0.1   0:01.67 httpd2-prefork
10787 wwwrun    20   0  195m 2632 1452 S     3  0.1   0:01.01 httpd2-prefork
10796 wwwrun    20   0  195m 2636 1428 S     3  0.1   0:01.12 httpd2-prefork
10812 wwwrun    20   0  195m 2496 1456 S     3  0.1   0:01.33 httpd2-prefork
10820 wwwrun    20   0  195m 2636 1420 S     3  0.1   0:01.53 httpd2-prefork
10866 wwwrun    20   0  195m 2624 1456 S     3  0.1   0:01.37 httpd2-prefork
10869 wwwrun    20   0  195m 2636 1456 S     3  0.1   0:01.05 httpd2-prefork
```

It was reaching close to 100% CPU and RAM during the peak of our load testing. This could very well be our bottleneck!

Our Load Balancer is really just a pass-through process so we sized it small at 1 x CPU and 2 GB of RAM. This appeared to be undersized for the load we were throwing at it so we decided to double it to see if we got better results.

We requested that our Infrastructure team increases the CPU and RAM to 2 x CPU, 4 GB of RAM.

5. **Increasing the Xms value for Apache Tomcat**

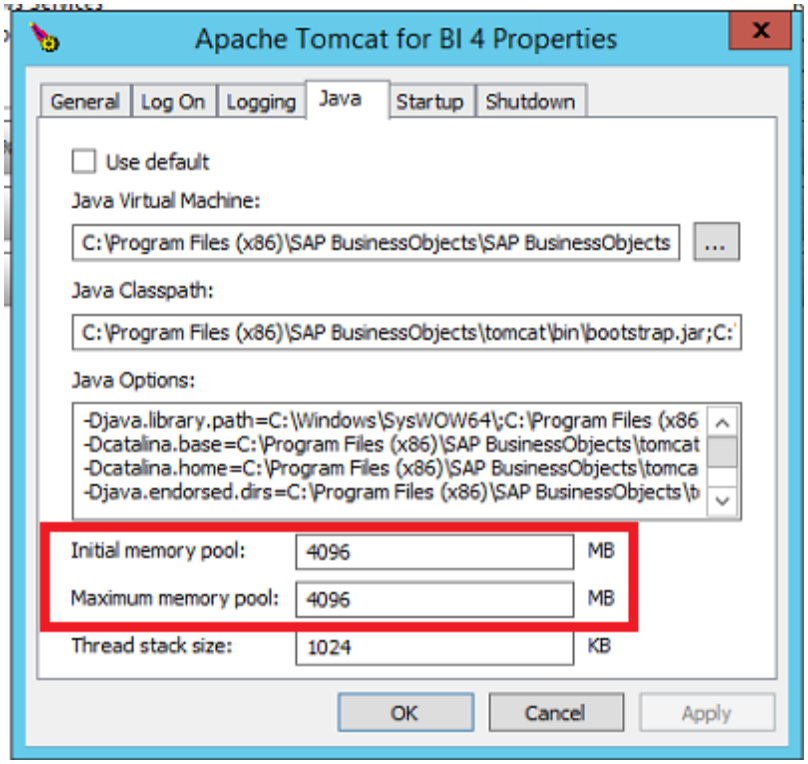
Another best practice we wanted to try was to set the Initial Heap Size (-Xms) value for our Tomcat server to the same value as our Maximum Heap Size (-Xmx) switch. What this does is reserves the maximum amount of memory that Tomcat will need for its heap memory allocations when the process starts up. If you use the out of the box settings, Tomcat will allocate a small initial heap and will grow the heap as needed until it reaches the maximum. By setting the Initial Heap Size to match the Maximum Heap Size, you are potentially saving time that the process would normal need to access and reserve memory as needed.

Since we had dedicated Application Server machines, we decided to test setting the Xmx to 4GB to save any time that we could.

Although we did not do an isolated test to see what the results are for this setting change, we could do this easily at any time in the future now that we have a proper benchmarking machine configured.

If this setting change did make a difference in our environment, it was negligible for the test plans we were running. For larger test plans, this setting change may make a bigger difference.

The easiest way to make this change is to open up the **Tomcat Configuration** tool and update the Intial memory pool setting in the Java tab:



Summary

Using a combination of JMeter, CA Introscope output and manual monitoring, we were able to identify a couple of big ticket items that were worth investigating. The two most likely candidates that we found were around Disk Utilization and the Load Balancer resources.

To resolve the Disk Utilization issues, we determined that our infrastructure team would have to help us get a better distribution of resources on the backend ESX servers which included adding another server to the mix. We were originally spread across 2 ESX servers and eventually moved to 3. These servers were dedicated to our deployment.

To resolve the Load Balancer issue, we increased the resources on the Load Balance server to 2 x CPUs and 4 GB of RAM. We also increased the MaxClients parameter in Apache to increase the concurrency. We cover this in more details in some below tests.

Let's take a look at some of our later tests AFTER we increased the Load Balancer resources AND swapped around some of our backend VMWare resources.

CA Introscope Dashboards

We did capture the dashboard screenshots for these tests as well.

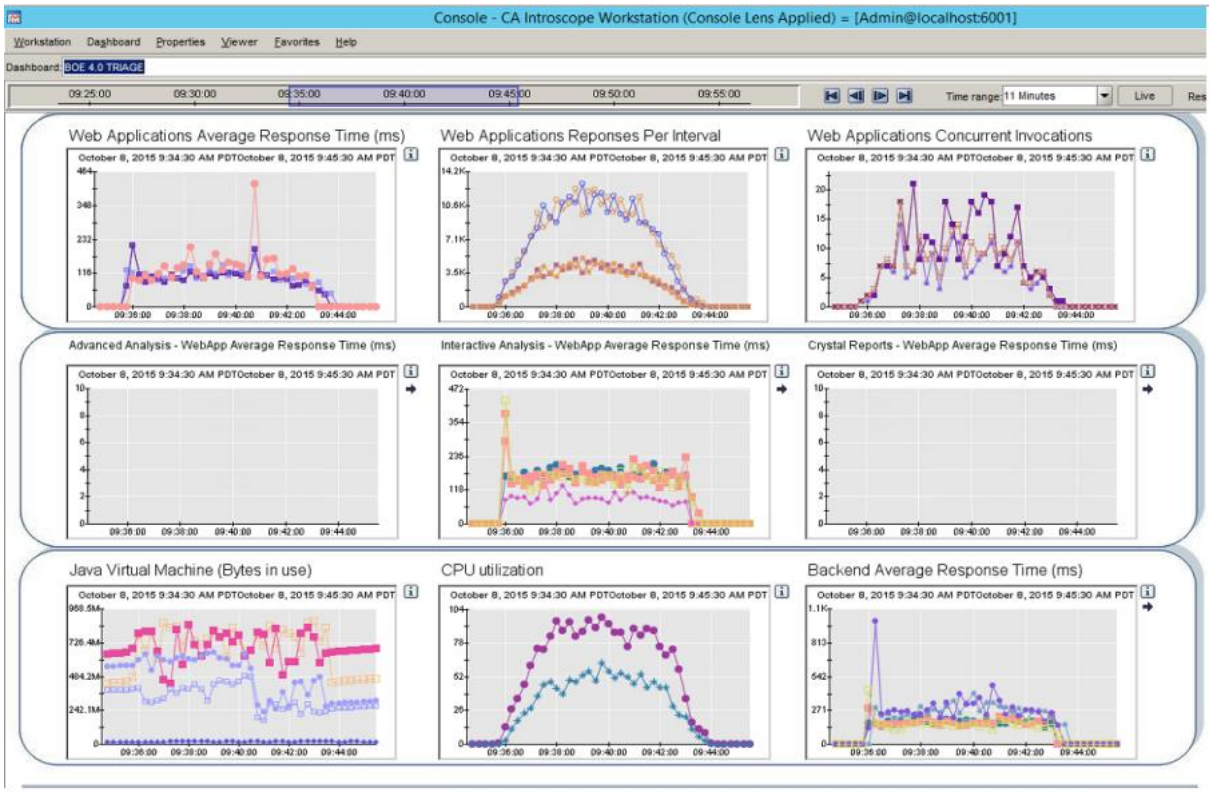
Resources for Test 7

These are the screenshots captured for Test 7 from CA Introscope Dashboards

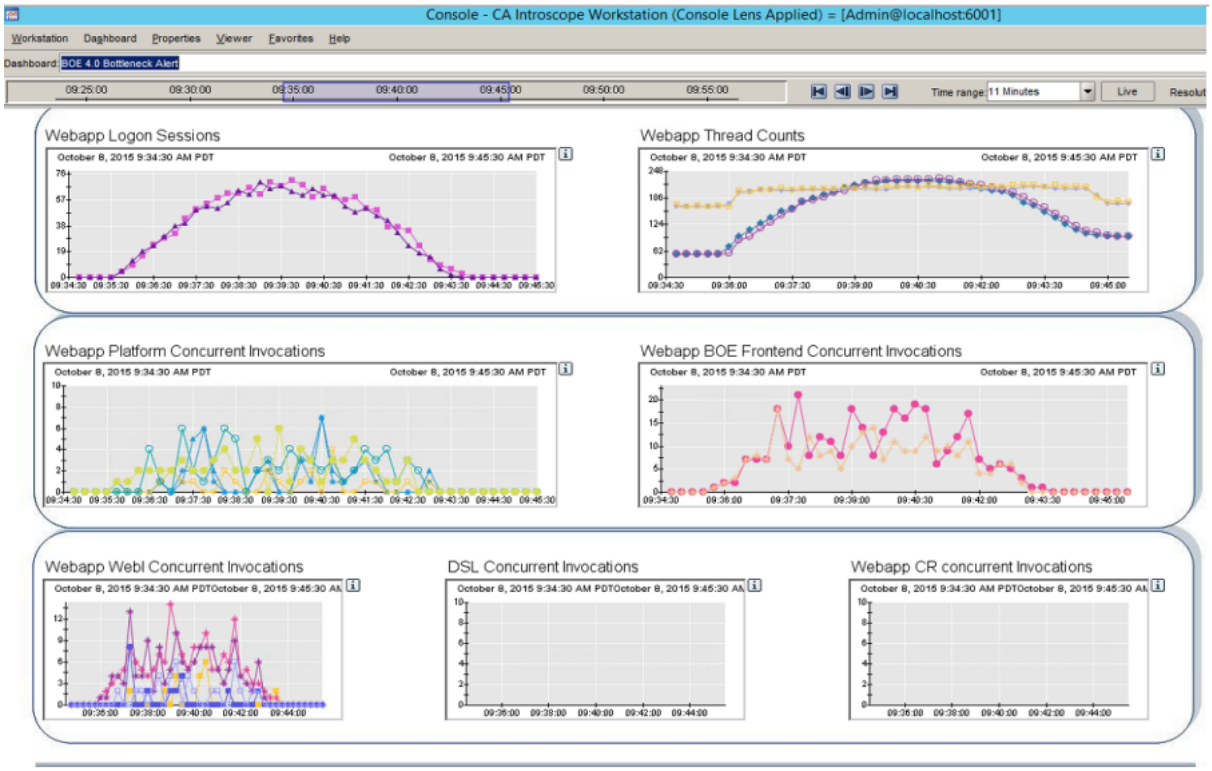
Tomcat CPU Usage – 150 ACU

Platform Averages – 150 ACU

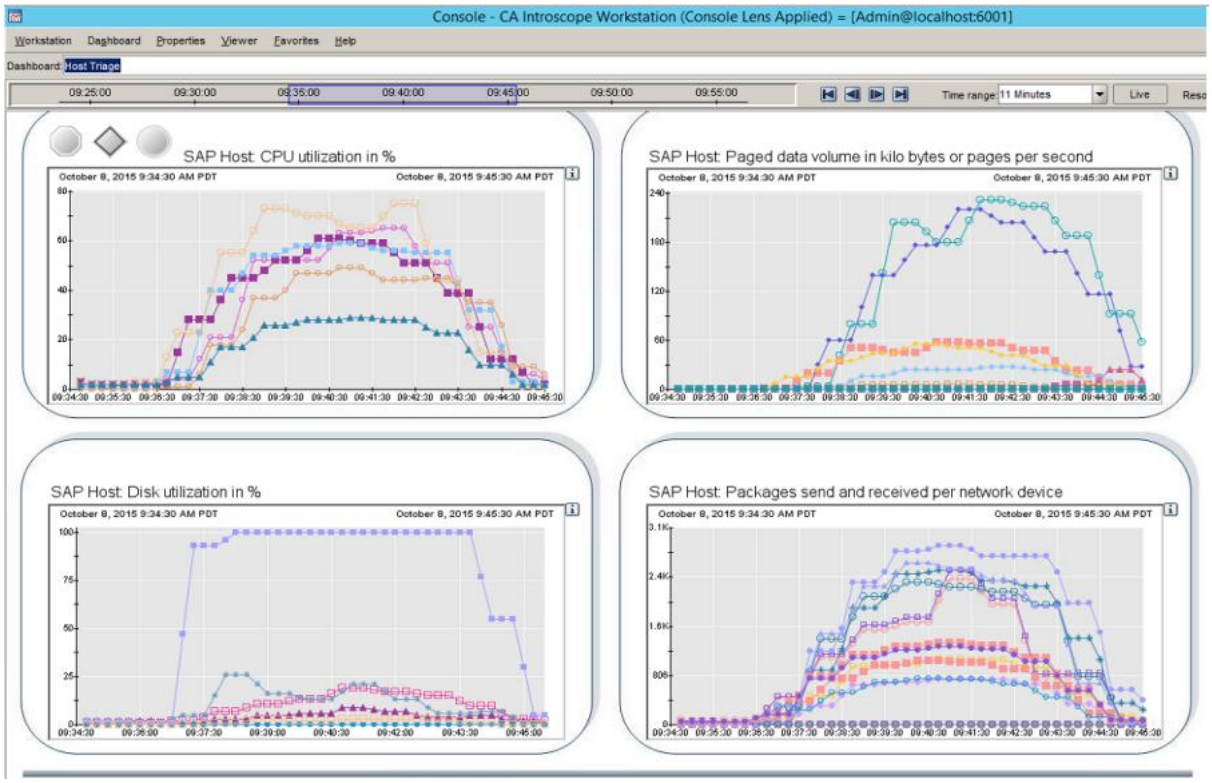
BI Triage – 150 ACU



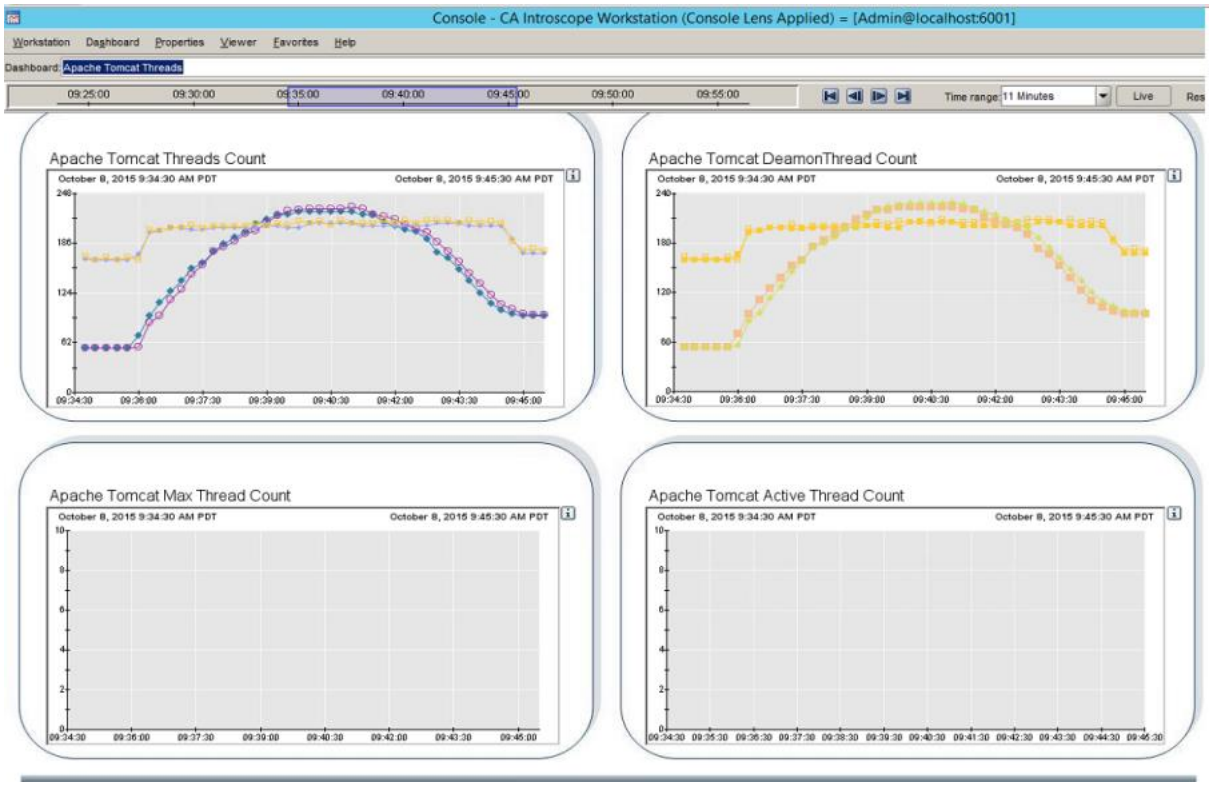
Bottleneck Alert – 150 ACU



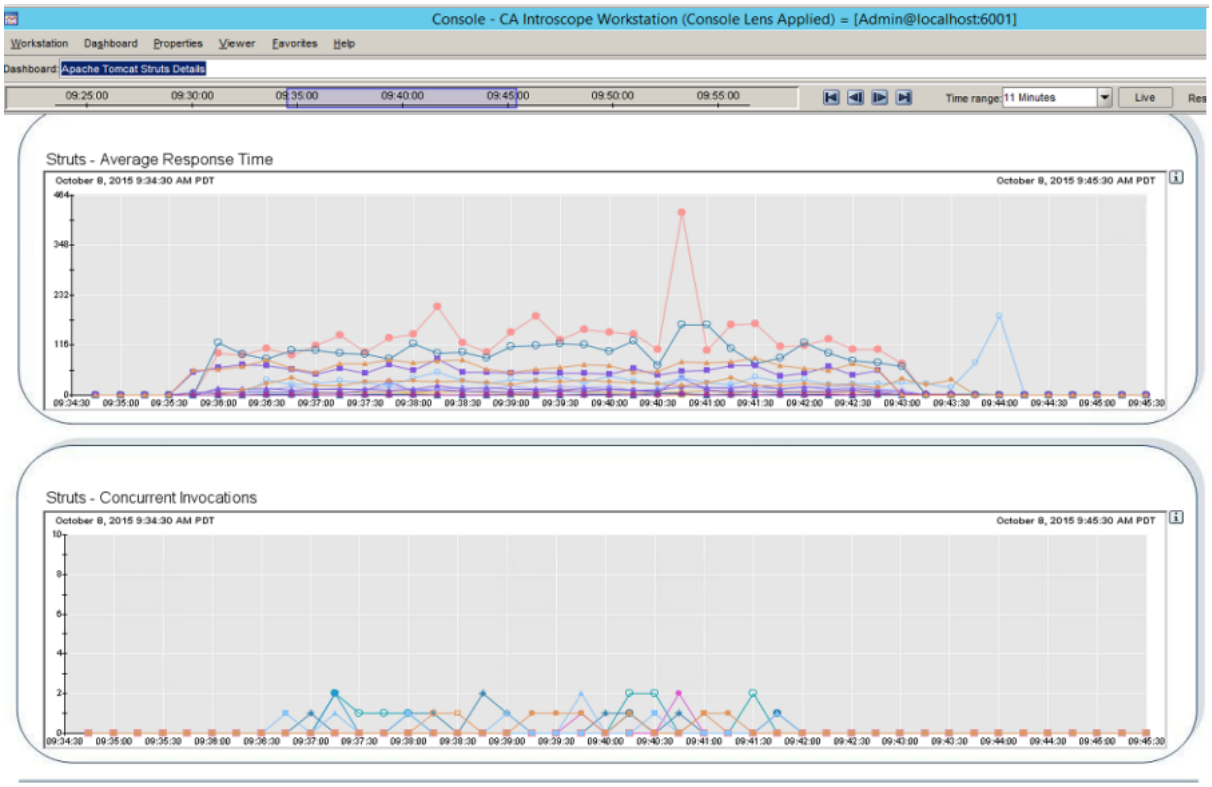
Host Triage – 150 ACU



Tomcat Threads – 150 ACU



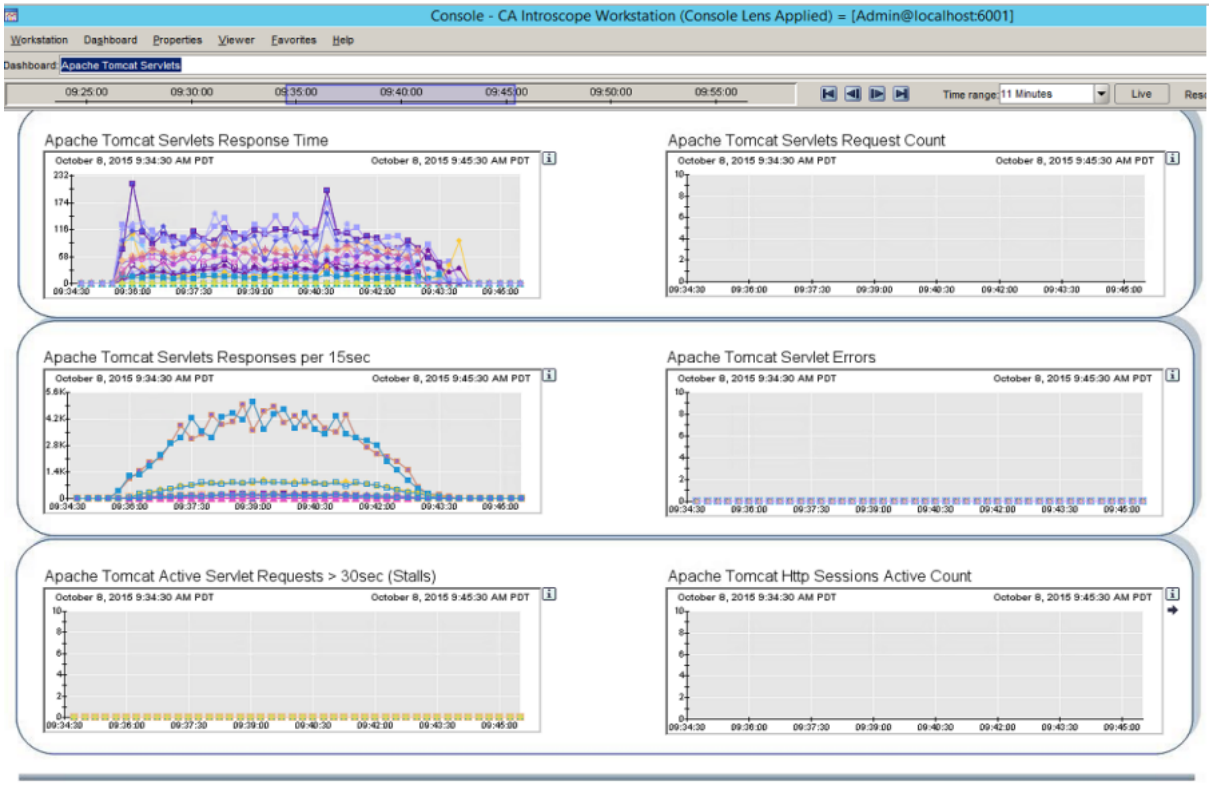
Tomcat Strut – 150 ACU



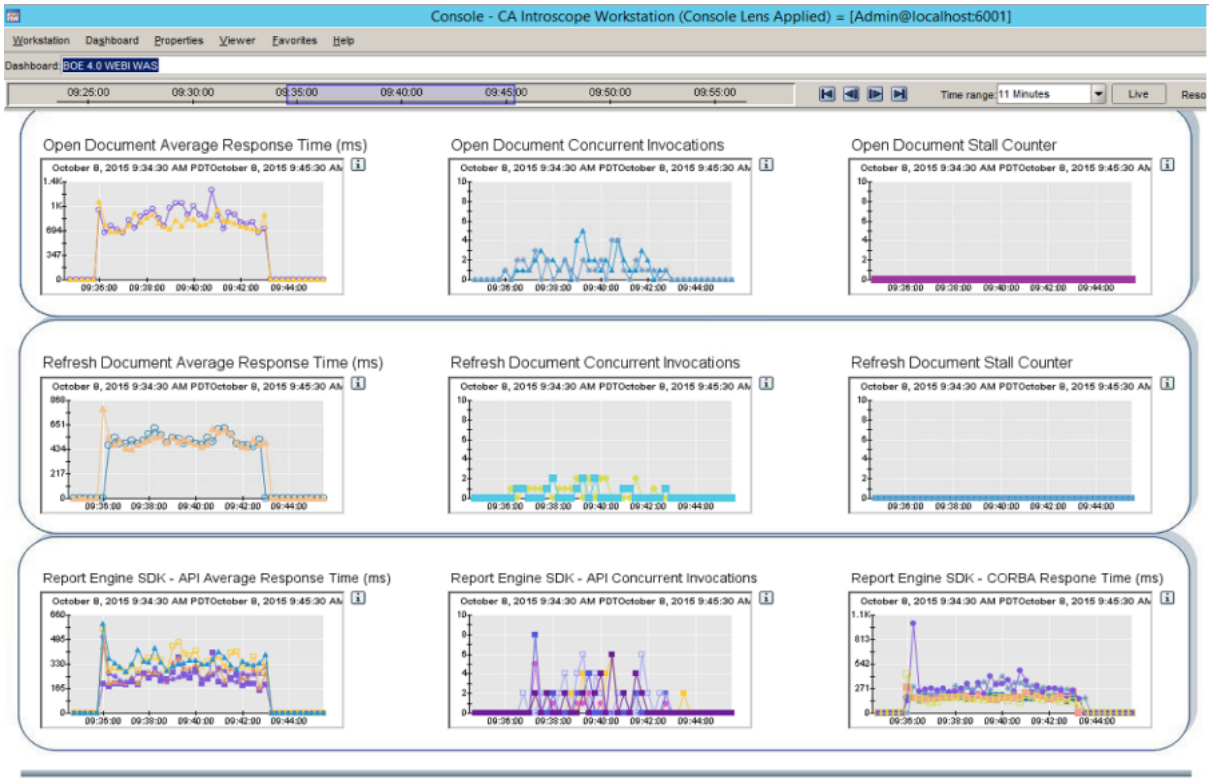
Tomcat GC Details – 150 ACU

Tomcat Memory – 150 ACU

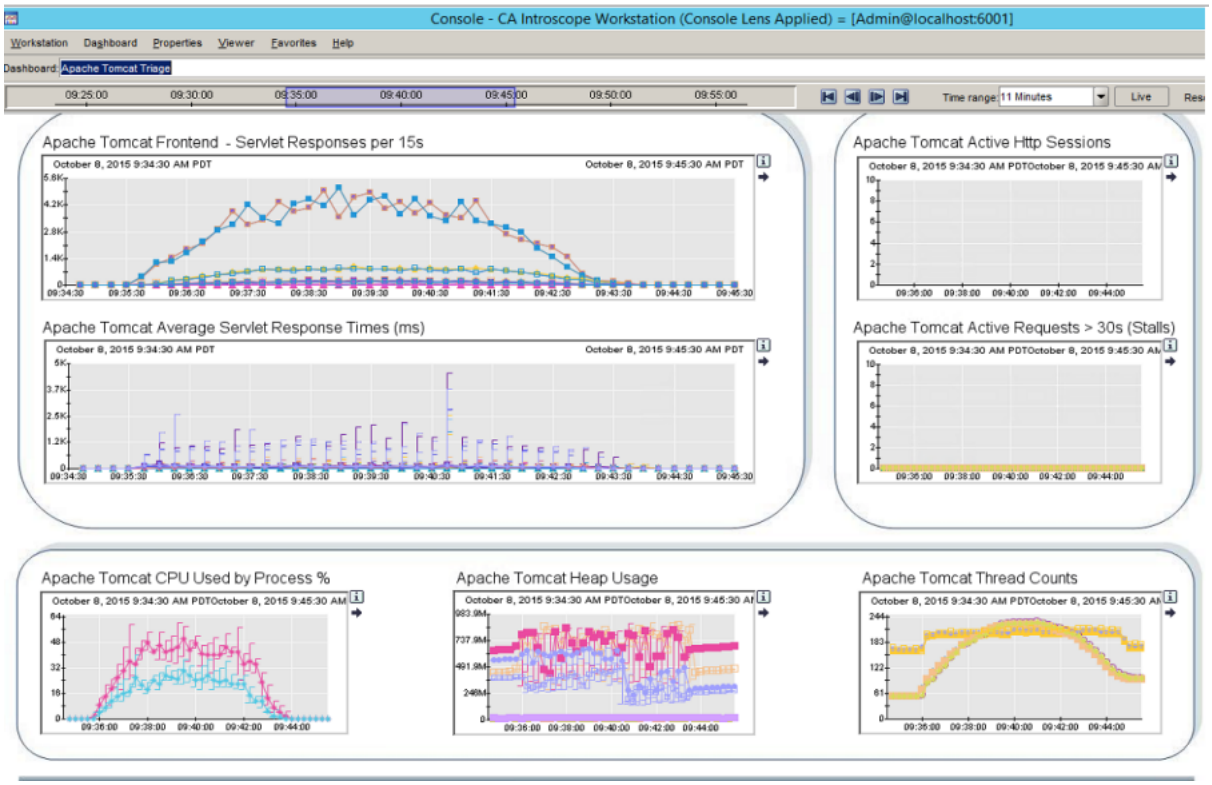
Tomcat Servlets – 150 ACU



WebI WAS – 150 ACU

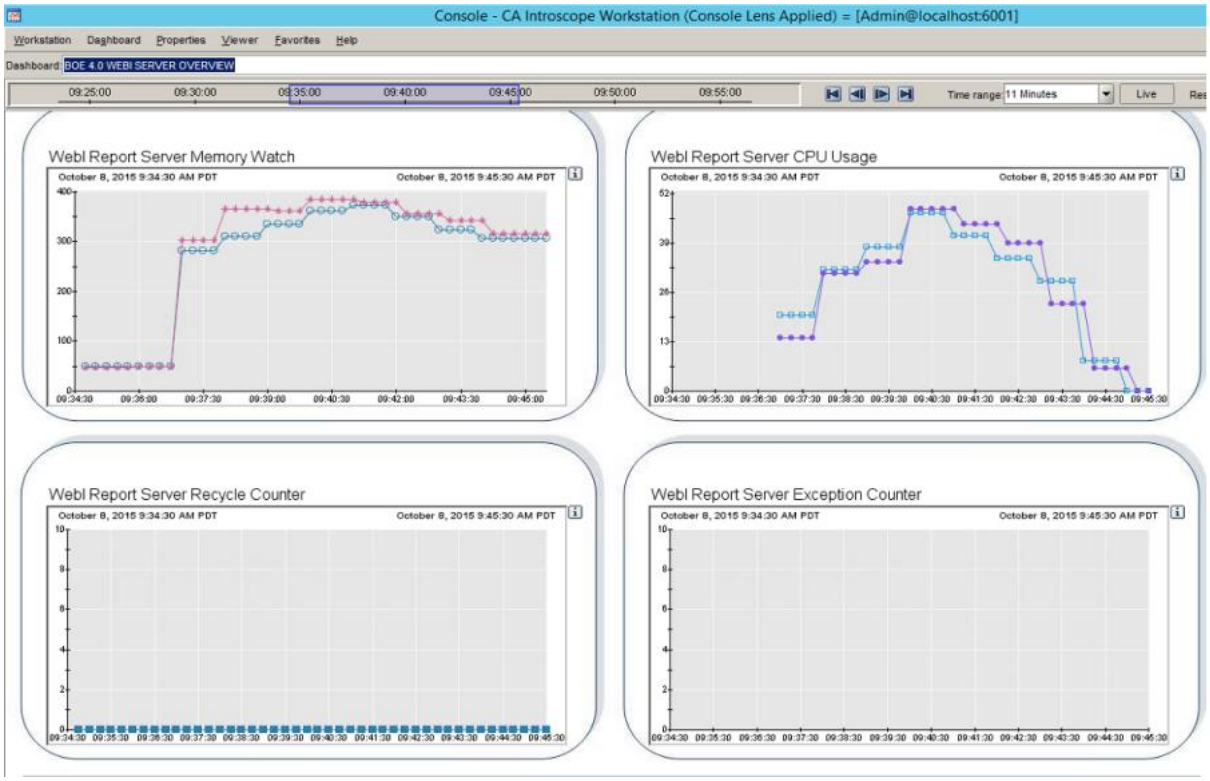


Tomcat Triage – 150 ACU



WebI Server Details – 150 ACU

WebI Server Overview – 150 ACU

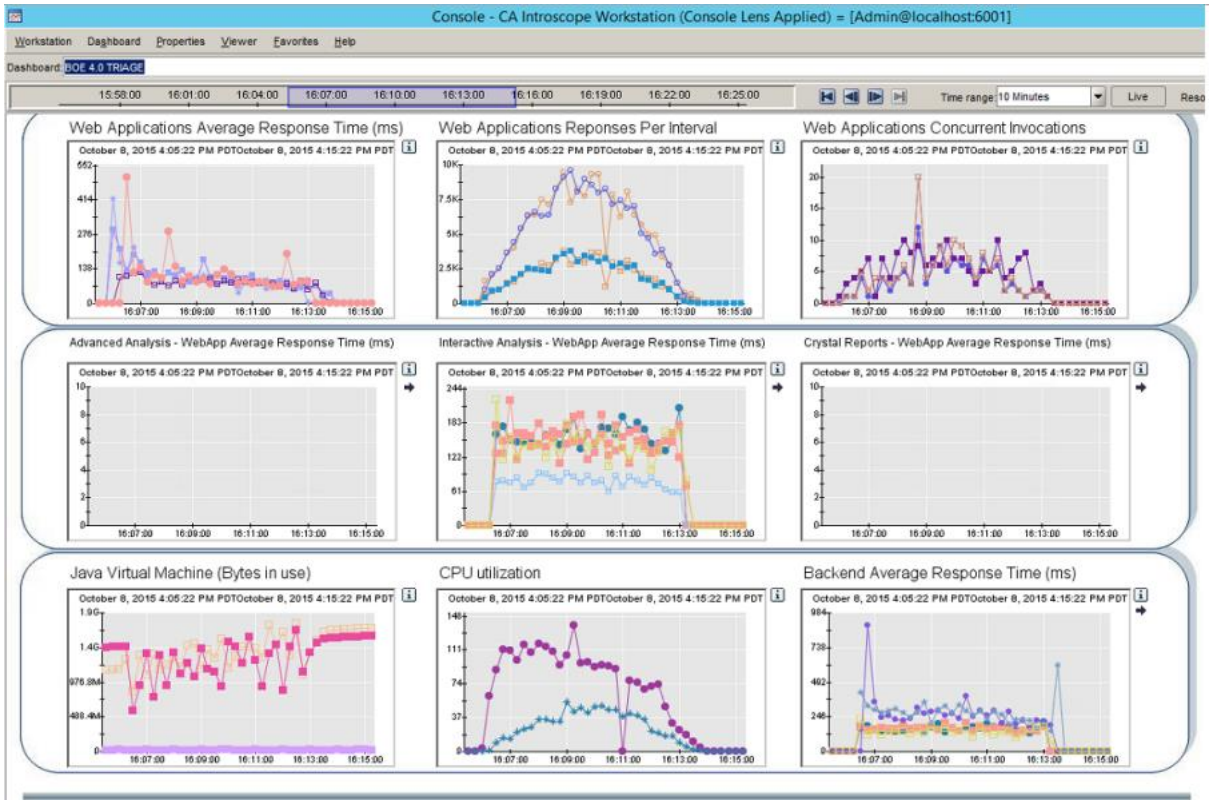


Resources for Test 8

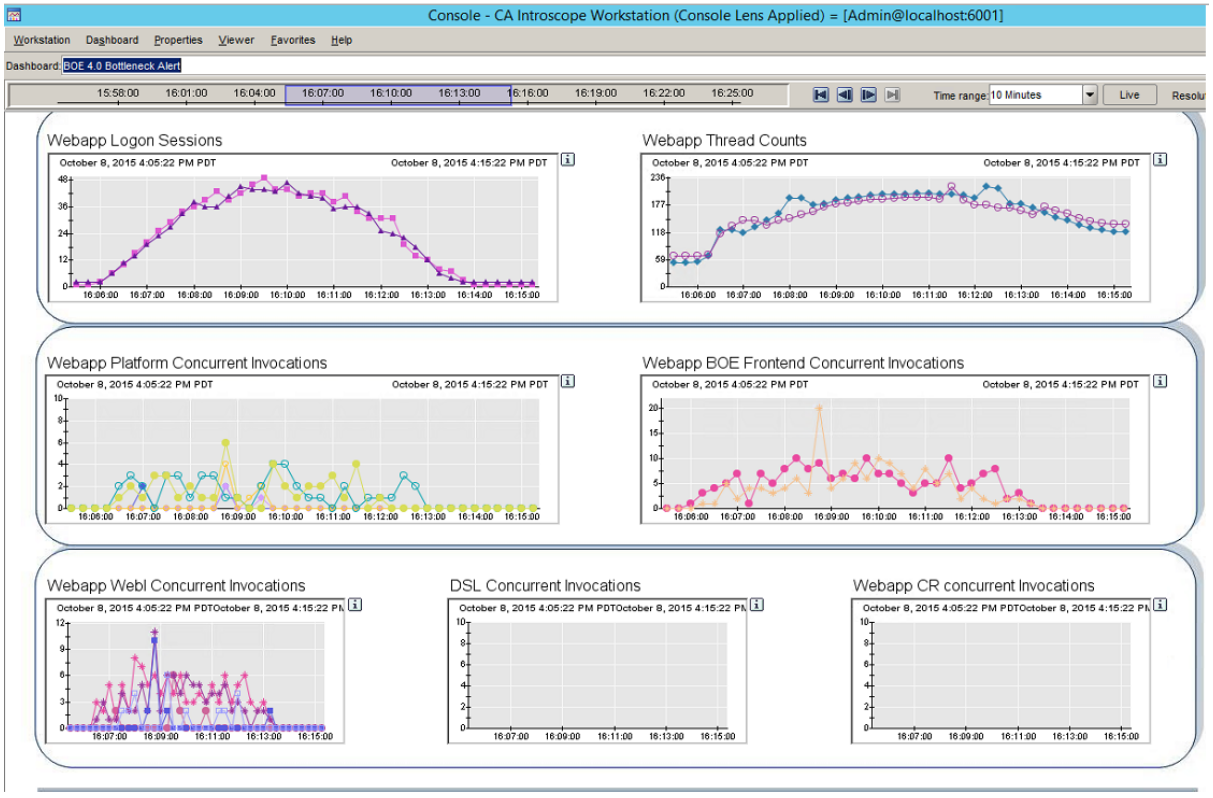
Tomcat CPU Usage – 100 ACU Tweaked

Platform Averages – 100 ACU Tweaked

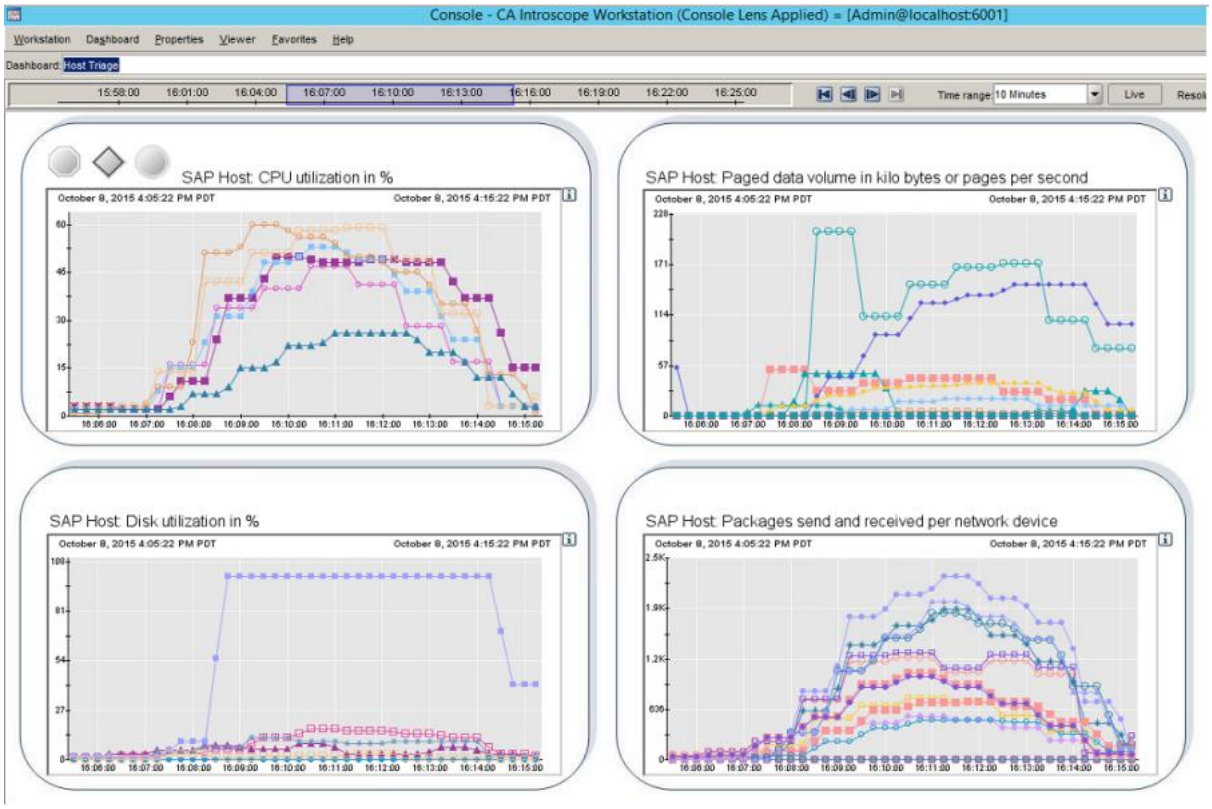
BI Triage – 100 ACU Tweaked



Bottleneck Alert – 100 ACU Tweaked



Host Triage – 100 ACU Tweaked



Tomcat Threads – 100 ACU Tweaked

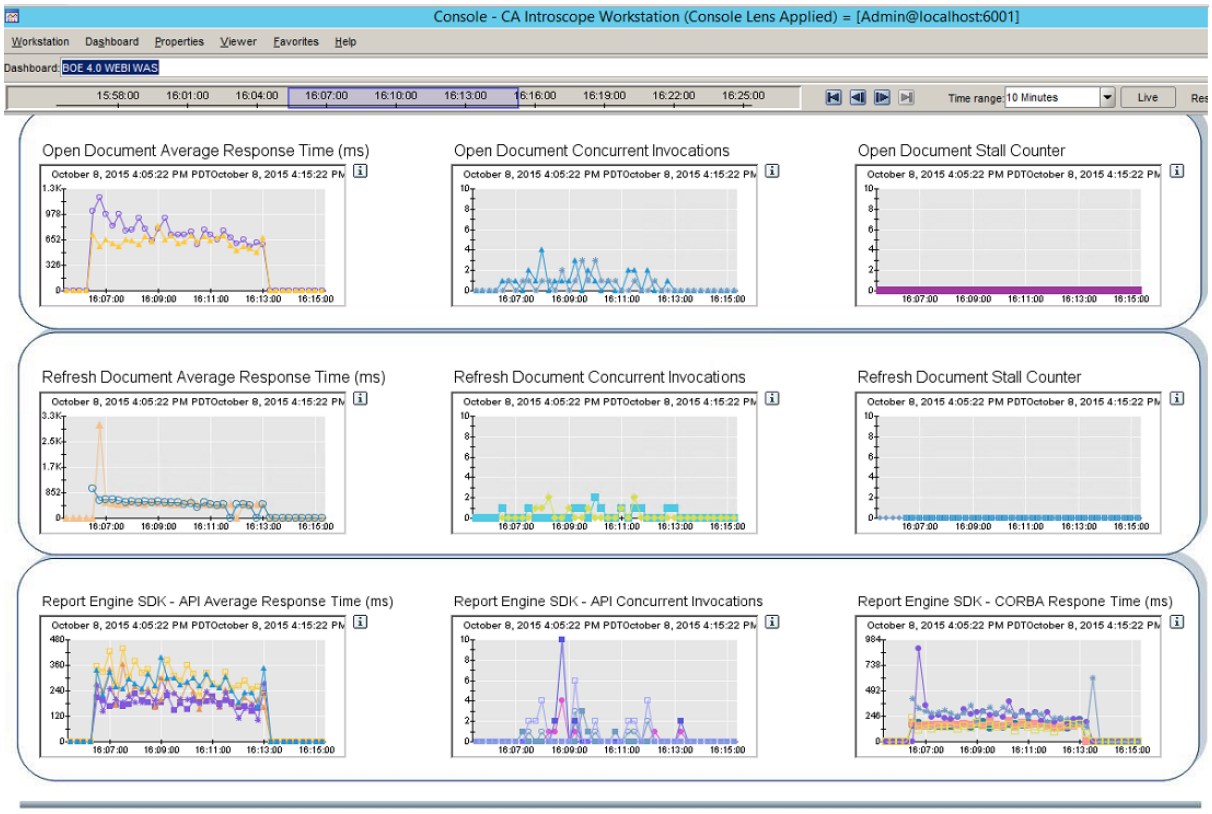
Tomcat Servlets – 100 ACU Tweaked

Tomcat GC Details – 100 ACU Tweaked

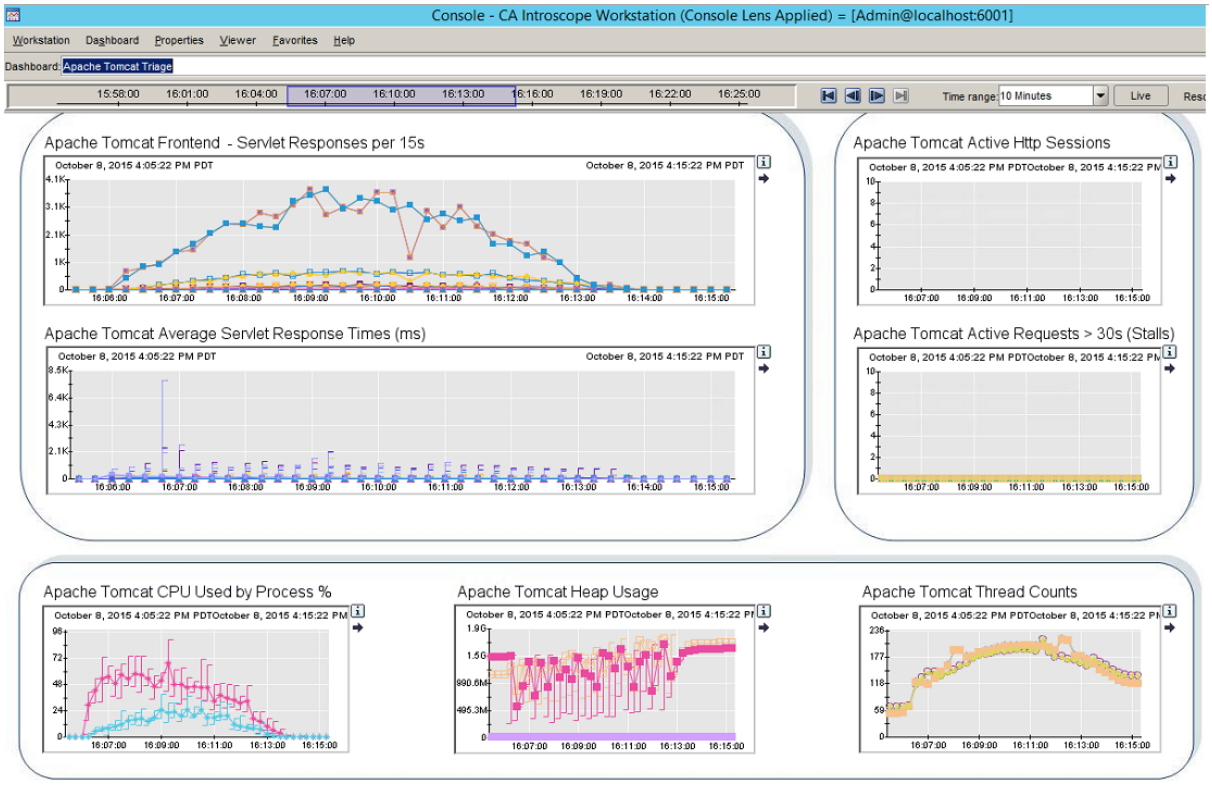
Tomcat Memory – 100 ACU Tweaked

Tomcat Strut – 100 ACU Tweaked

WebI WAS – 100 ACU Tweaked



Tomcat Triage – 100 ACU Tweaked

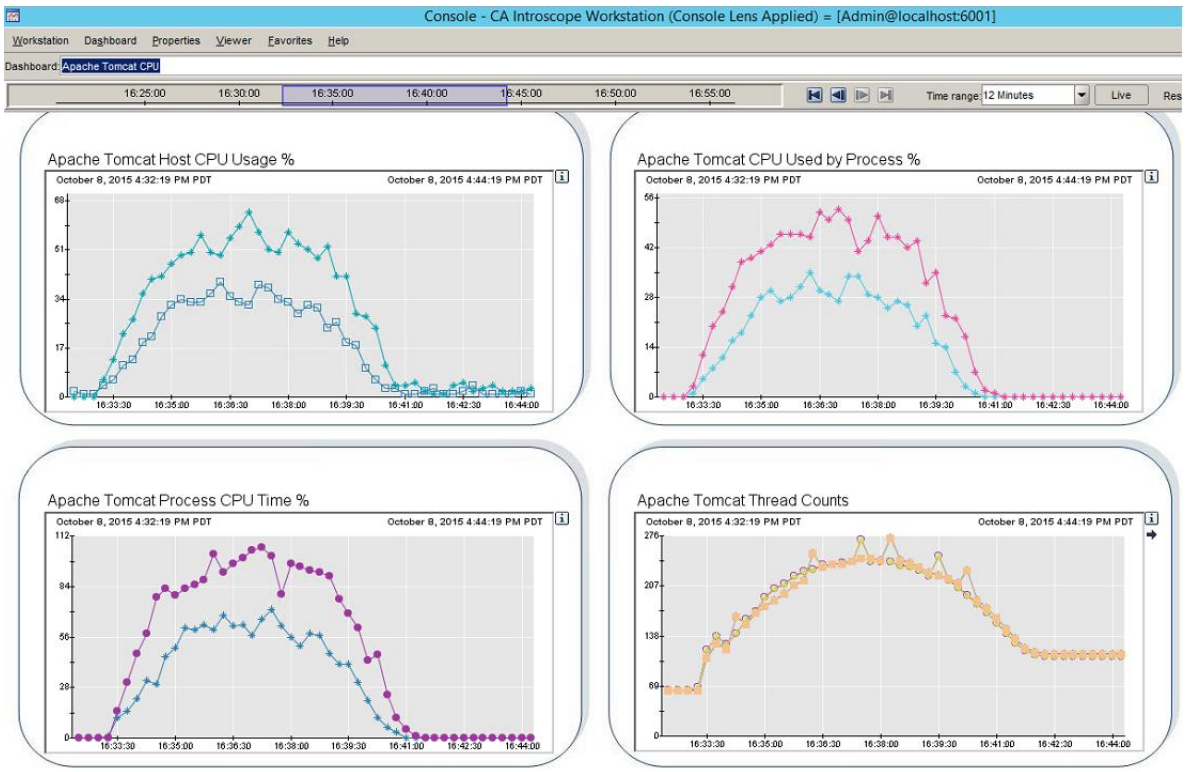


WebI Server Details – 100 ACU Tweaked

WebI Server Overview – 100 ACU Tweaked

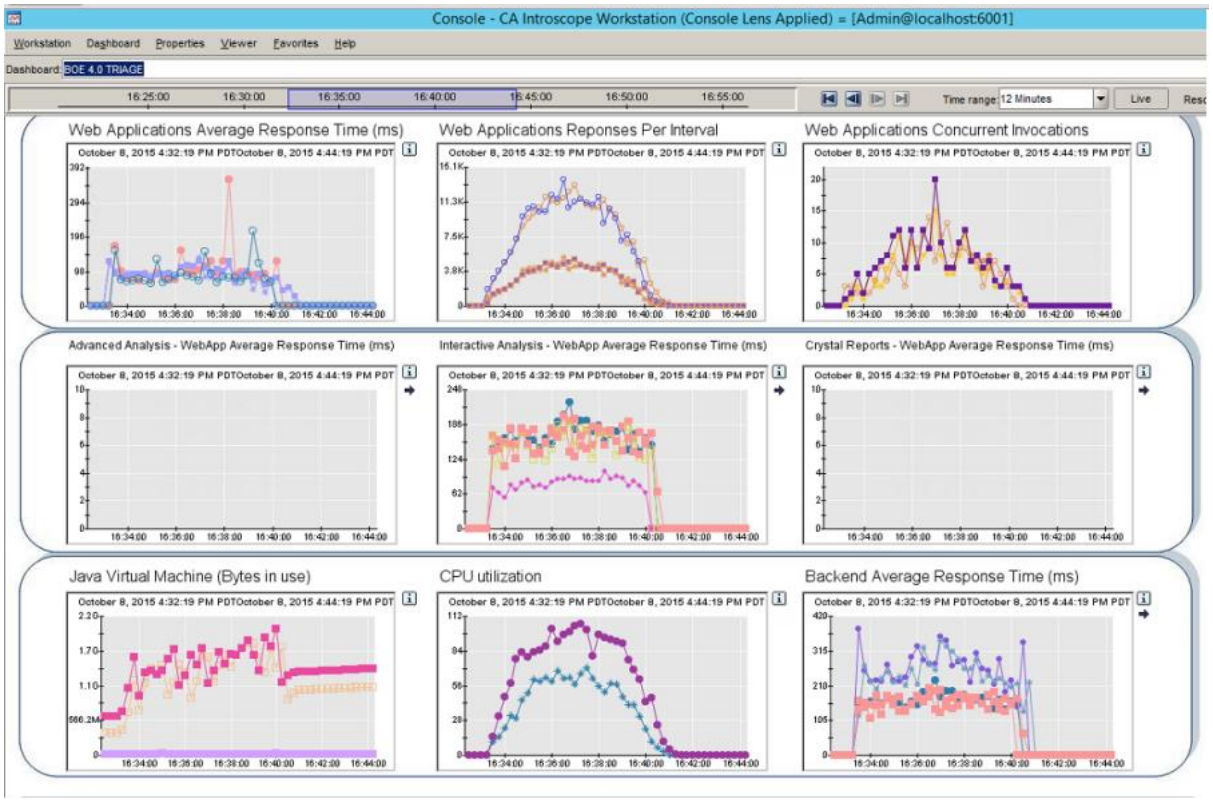
Resources for Test 9

Tomcat CPU Usage – 150 ACU Tweaked

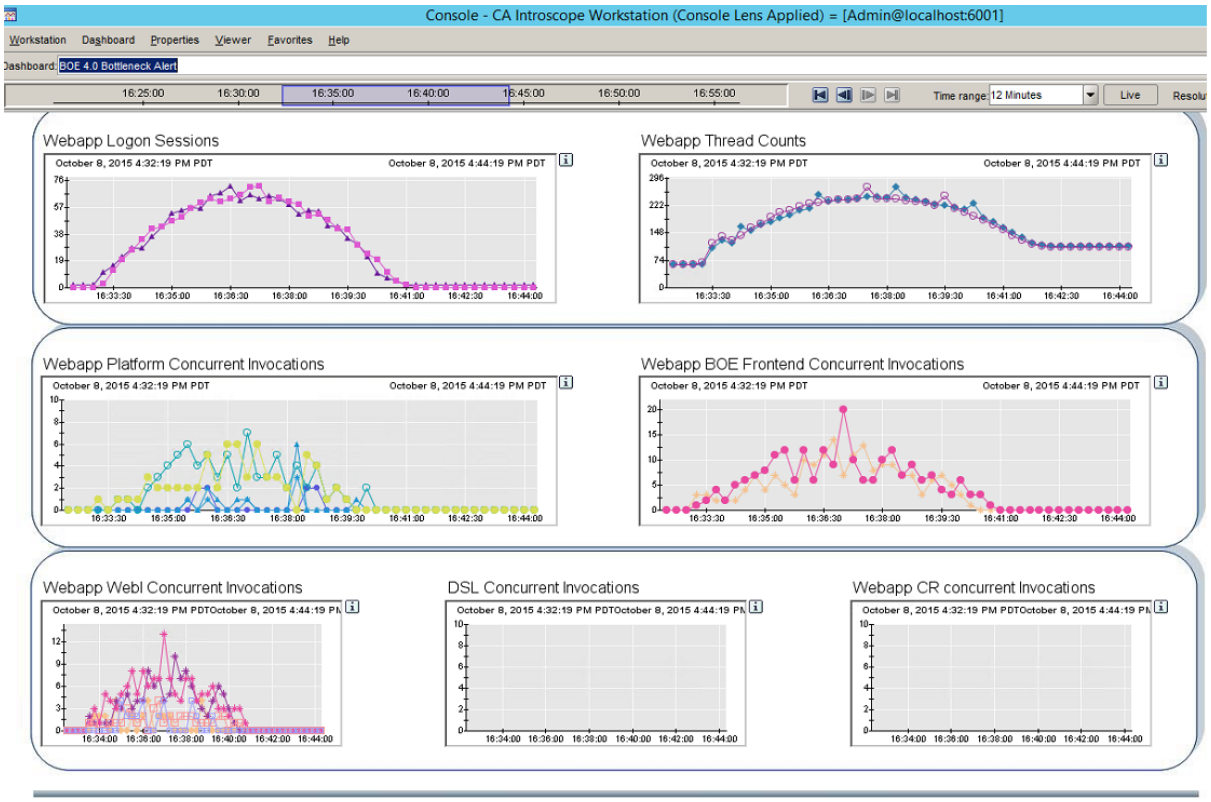


Platform Averages – 150 ACU Tweaked

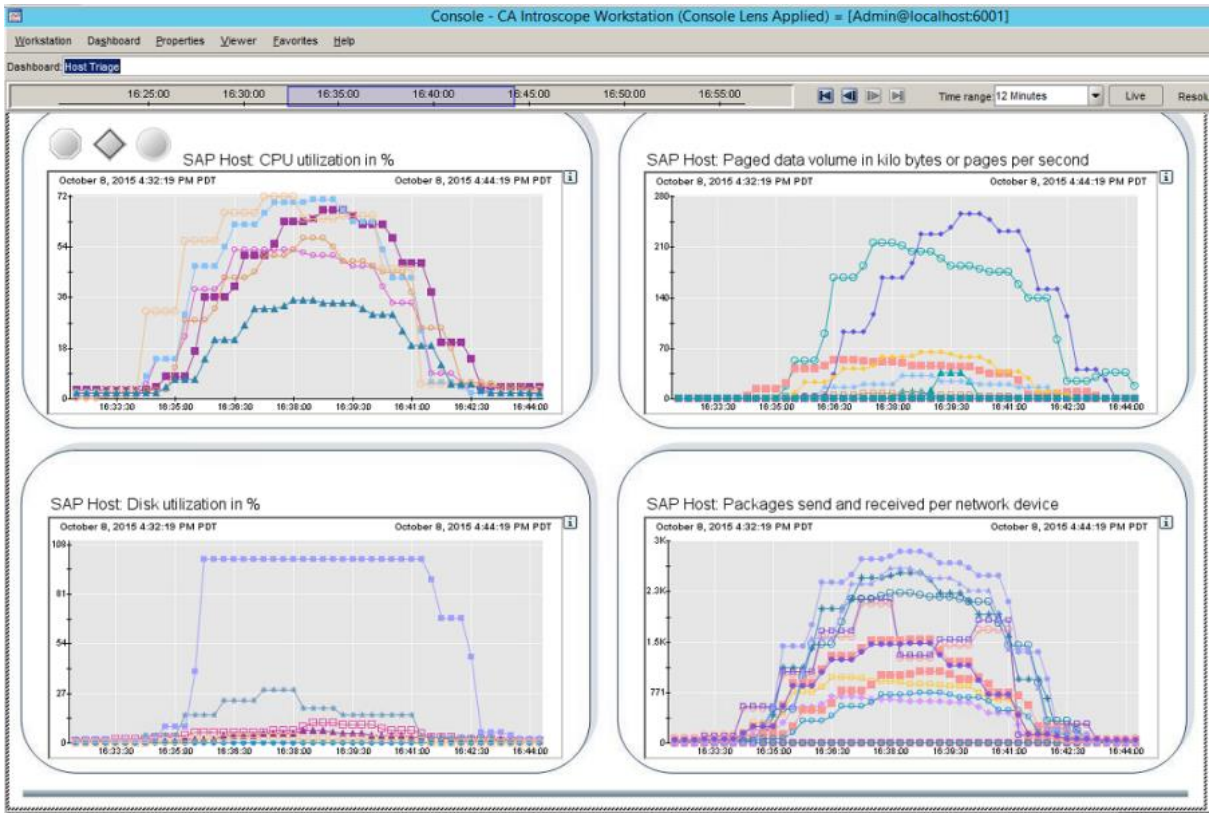
BI Triage – 150 ACU Tweaked-0003



Bottleneck Alert – 150 ACU Tweaked



Host Triage – 150 ACU Tweaked-0002

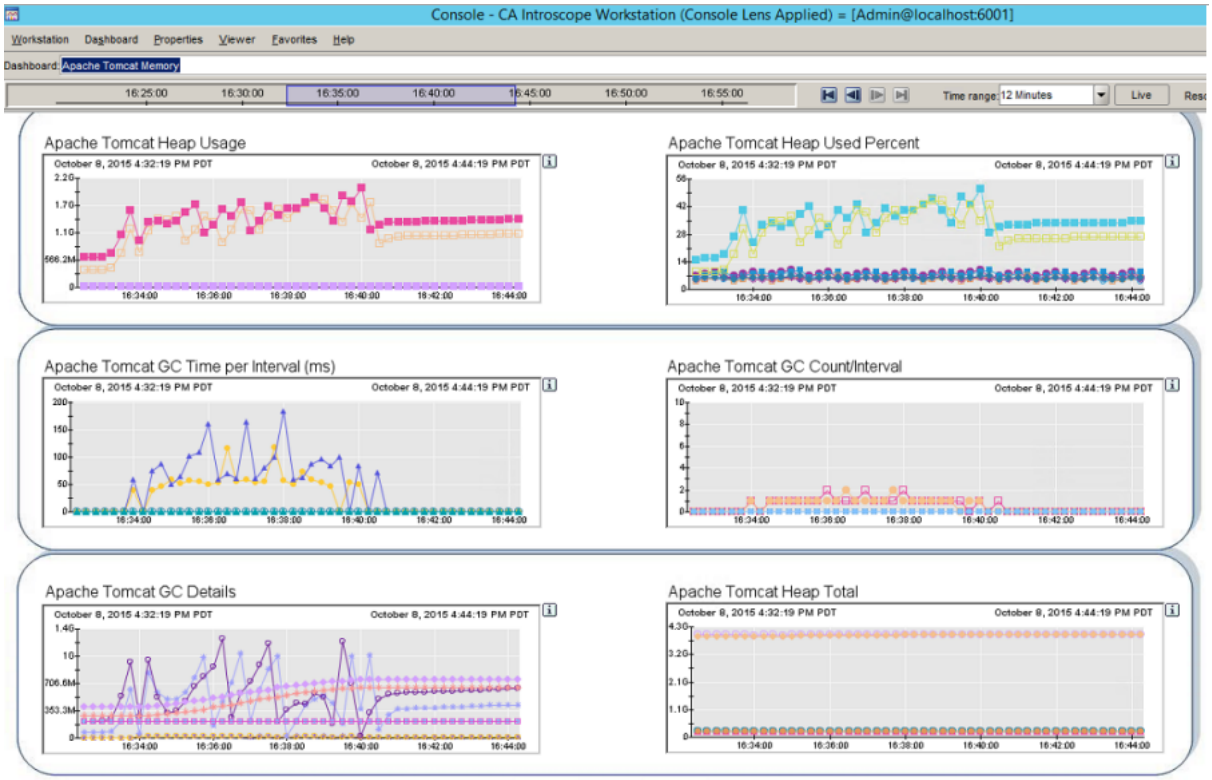


Tomcat Threads – 150 ACU Tweaked

Tomcat Strut – 150 ACU Tweaked

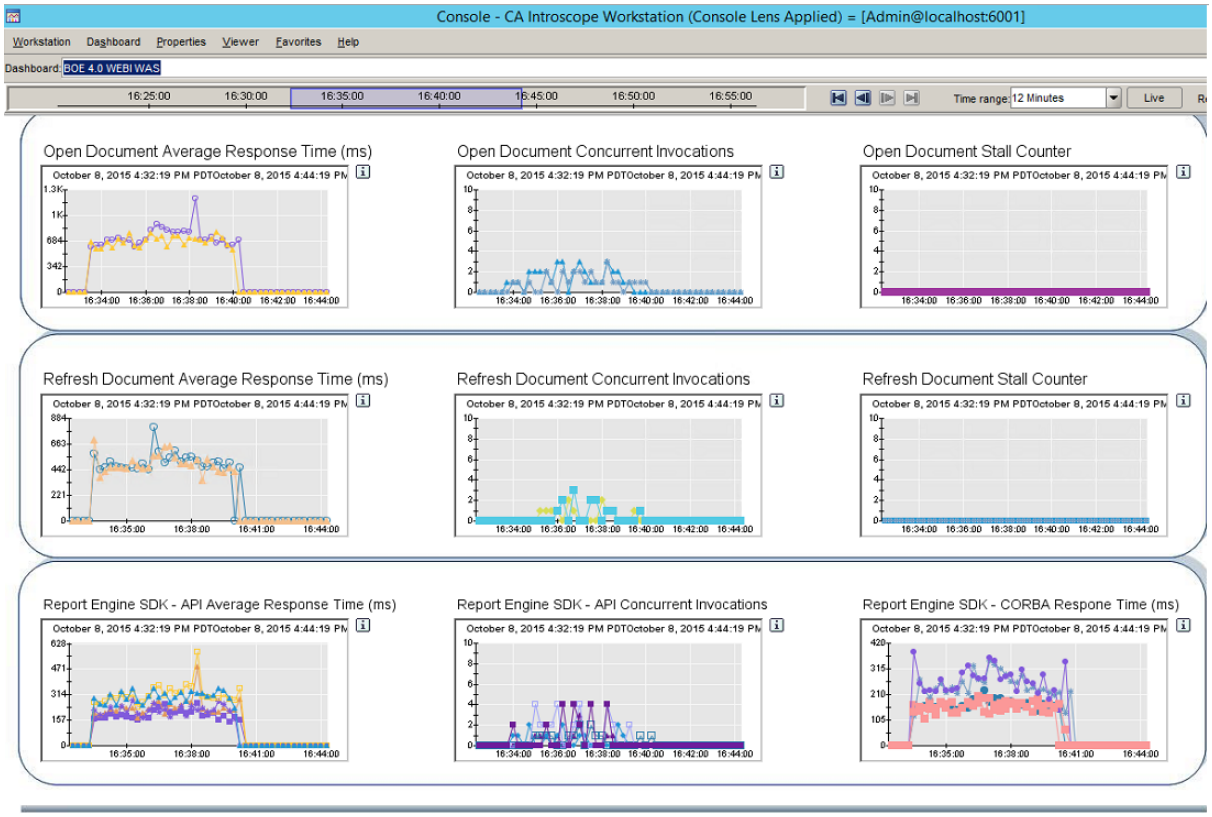
Tomcat GC Details – 150 ACU Tweaked

Tomcat Memory – 150 ACU Tweaked-0000

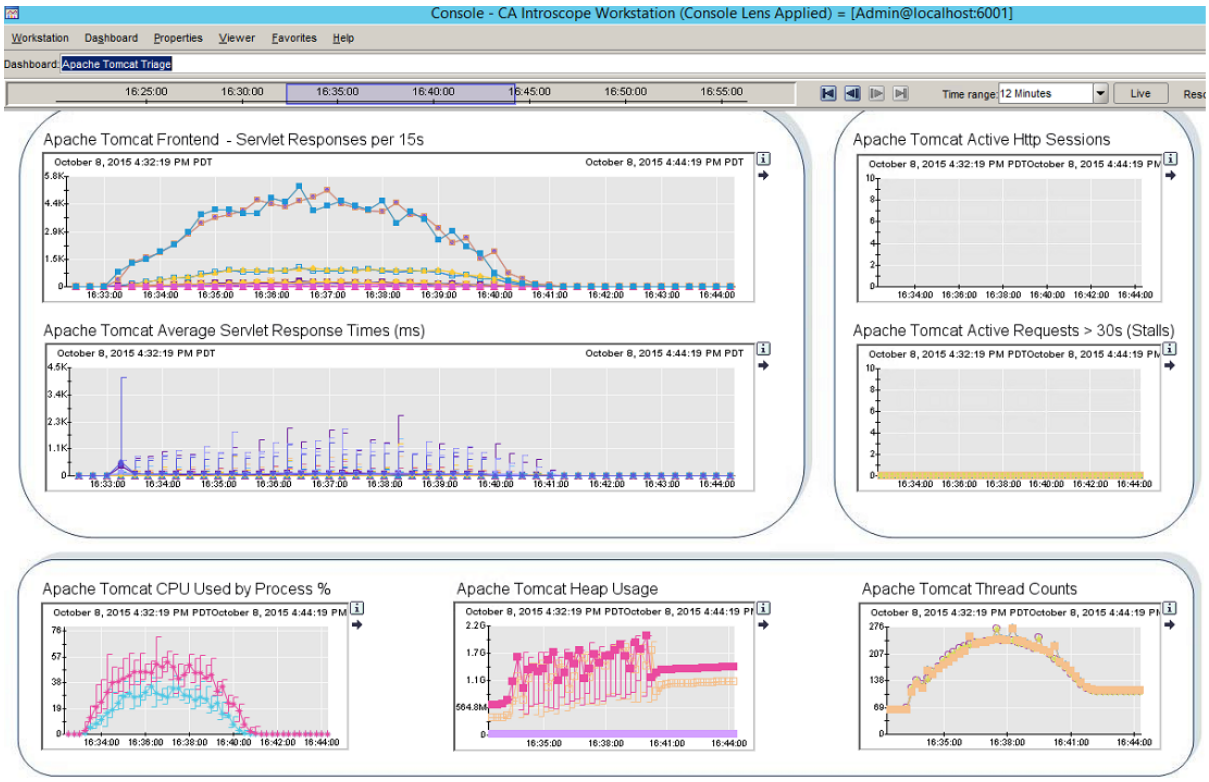


Tomcat Servlets - 150 ACU Tweaked

WebI WAS – 150 ACU Tweaked



Tomcat Triage – 150 ACU Tweaked



WebI Server Details – 150 ACU Tweaked-0001

WebI Server Overview – 150 ACU Tweaked

There are no screenshots and images for Test 10.

Test 11- 150 Active Concurrent Users - After ESX and Load Balancer Changes

We re-ran the 150 ACU test after making the changes to the ESX server mix on the backend VMWare Infrastructure and increasing the resources for our Load Balancer machine. If we did resolve the bottleneck, it should show up in these test results.

Test 11	**60 ACU Edit / 300 Transactions / 120 Second Ramp-up** **90 ACU View / 450 Transactions / 240 Second Ramp-up**
Active Concurrent Users	150
Total Transactions	750
Start Time	3:23 PM
End Time	3:31 PM
Duration	~8 Minutes

JMeter Output

Edit Workflow:

Summary Report

Name: Edit Web Intelligence Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	300	937	481	4895	423.00	0.00%	45.1/min	315.24	428996.3
Browse Folders	300	1984	1259	6146	570.44	0.00%	44.8/min	284.50	390466.3
View Report	300	2059	1249	5834	646.09	0.00%	44.2/min	684.17	952055.0
Edit and Refresh	300	3666	2544	7686	893.08	0.00%	42.5/min	382.48	552426.4
Close and Logout	300	300	92	3822	403.14	0.00%	44.8/min	22.53	30912.7
TOTAL	1500	1789	92	7686	1300.72	0.00%	3.3/sec	1494.81	470971.3

For the Edit workflow, we could see the following values:

Edit Webi	**# Samples**	**Average**	**Min**	**Max**	**Std Dev**
Login	300	937	481	4895	423.00
Browse Folders	300	1984	1259	6146	570.44
View Report	300	2059	1249	5834	646.09
Edit and Refresh	300	3666	2544	7686	893.08
Close and Logout	300	300	92	3822	403.14
Total Transaction	300	**8946**	**5625**	**28383**	**2935.75**
Average	1500	1789	92	7686	1300.72

These results appear a lot closer to our 50 & 100 ACU values. We'll do a proper comparison below to see how these fit in with out other results.

Interpretation of the above test results:

- The Average Total Transaction time was **8.946** seconds from the Edit workflow.
- The fastest (Min) potential workflow was **5.625** seconds
- The slowest (Max) potential workflow was **28.383** seconds
- The slowest action was the **Edit and Refresh** action, as we would expect.

These results look much better than our Test 5 results and this is a 150 ACU to 150 ACU comparison (apples to apples) so it should give us a good idea if our changes made a big difference or not.

Below we will do a side-by-side analysis to look at how these stack up.

Comparing Test 5 to Test 11

Below are the results for Test 5 and Test 11 shown side by side with some limited analysis run against them. Remember that Test 5 is 150 ACU before the tweaks we made and Test 11 is 150 ACU after the tweak.

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 5**	**Test 11**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**
Login	300	300	2086	937	123%	529	481	10%	5971	4895	22%	1360.39	423	222%
Browse Folders	300	300	3279	1984	65%	1387	1259	10%	7057	6146	15%	1530.49	570.44	168%
View Report	300	300	3871	2059	88%	1662	1249	33%	9983	5834	71%	1679.99	646.09	160%
Edit and Refresh	300	300	5307	3666	45%	2962	2544	16%	10372	7686	35%	1577.33	893.08	77%
Close and Logout	300	300	953	300	218%	281	92	205%	4931	3822	29%	700.75	403.14	74%
Total Transaction	**300**	300	**15496**	**8946**	73%	**6821**	**5625**	21%	**38314**	**28383**	35%	**6848.95**	**2935.75**	133%

Below is a screenshot of the table with more advanced formatting as well

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 5**	**Test 11**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**%Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**
Login	300	300	2086	937	123%	529	481	10%	5971	4895	22%	1360.39	423	222%
Browse Folders	300	300	3279	1984	65%	1387	1259	10%	7057	6146	15%	1530.49	570.44	168%
View Report	300	300	3871	2059	88%	1662	1249	33%	9983	5834	71%	1679.99	646.09	160%
Edit and Refresh	300	300	5307	3666	45%	2962	2544	16%	10372	7686	35%	1577.33	893.08	77%
Close and Logout	300	300	953	300	218%	281	92	205%	4931	3822	29%	700.75	403.14	74%
Total Transaction	**300**	300	**15496**	**8946**	73%	**6821**	**5625**	21%	**38314**	**28383**	35%	**6848.95**	**2935.75**	133%
Total	1500	1500	3099	1789	73%	281	92	205%	10372	7686	35%	2055.81	1300.72	58%

We see much better results when we compare Test 11 to Test 5. It would appear that we overcame the bottleneck for this test and received much faster response time from our environment.

Here is a comparison of the Average Total Transaction time between Test 5 and 11.

Test 5	**15.496** seconds
Test 11	**8.946** seconds
Difference	**6.523** seconds (73% Better Performance with Test 11)

If we compare Test 11 to Test 2 (50 ACU) we can see they are better aligned.

Test 2	**9.066** seconds
Test 11	**8.946** seconds
Difference	**0.120** seconds (1.3% faster results with Test 11!)

The data would certainly suggest that we have resolved the major bottlenecks in our environment and we have been able to reach our 150 ACU target without degrading performance across our test. Let's see how the View workflow held up.

View Workflow:

Name: View and Refresh Webi Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	450	993	482	4839	377.39	0.00%	1.1/sec	469.85	429065.9
Browse Folders	450	2068	1256	5626	626.35	0.00%	1.1/sec	426.49	390468.3
View Report	450	1181	735	2594	242.14	0.00%	1.1/sec	265.21	239966.9
Refresh and Prom...	450	1880	1269	5528	538.91	0.00%	1.1/sec	280.87	255693.0
Close and Logout	450	343	110	3843	353.52	0.00%	1.1/sec	36.22	33149.0
TOTAL	2250	1293	110	5626	769.09	0.00%	5.2/sec	1369.92	269668.6

For the View workflow, we could see the following values:

View Webi	# Samples	Average	Min	Max	Std Dev
Login	450	993	482	4839	377.39
Browse Folders	450	2068	1256	5626	626.35
View Report	450	1181	735	2594	242.14
Refresh & Prompts	450	1880	1269	5528	538.91
Close and Logout	450	343	220	3843	353.52
Total Transaction	450	**6465**	**3962**	**22430**	**2138.31**
Average	**2700**	1293	110	5626	769.09

Interpretation of the above test results

- The Average Total Transaction time was **6.465** seconds
- The Fastest potential Transaction time could be **3.962** seconds
- The Slowest potential Transaction time could be **22.430** seconds

- The slowest action on average is the Browse Folders action at **5.626** seconds

Comparing Test 5 to Test 11

Below is a side by side comparison of the results from our Test 5 (150 ACU) and our Test 11 (150 ACU) JMeter output.

	# Samples		Average			Min			Max			Std Dev		
View Webi	**Test 5**	**Test 11**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**
Login	450	450	2853	993	187%	639	482	33%	6112	4839	26%	1403.54	377.39	272%
Browse Folders	450	450	4010	2068	94%	1439	1256	15%	8065	5626	43%	1449.35	626.35	131%
View Report	450	450	2900	1181	146%	893	735	21%	7174	2594	177%	1145.74	242.14	373%
Refresh & Prompt	450	450	3646	1880	94%	1477	1269	16%	7798	5528	41%	1249.15	538.91	132%
Close and Logout	450	450	1358	343	296%	334	220	52%	4845	3843	26%	801.3	353.52	127%
Total Transaction	**450**	**450**	**14767**	**6465**	128%	**4782**	**3962**	21%	**33994**	**22430**	52%	**6049.08**	**2138.31**	183%
Total	2250	2250	2953	1293	128%	334	110	204%	8065	5626	43%	1532.29	769.09	99%

Below is a screenshot of the table with more advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
View Webi	**Test 5**	**Test 11**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**%Diff**	**Test 5**	**Test 11**	**% Diff**	**Test 5**	**Test 11**	**% Diff**
Login	450	450	2853	993	187%	639	482	33%	6112	4839	26%	1403.54	377.39	272%
Browse Folders	450	450	4010	2068	94%	1439	1256	15%	8065	5626	43%	1449.35	626.35	131%
View Report	450	450	2900	1181	146%	893	735	21%	7174	2594	177%	1145.74	242.14	373%
Refresh & Prompt	450	450	3646	1880	94%	1477	1269	16%	7798	5528	41%	1249.15	538.91	132%
Close and Logout	450	450	1358	343	296%	334	220	52%	4845	3843	26%	801.3	353.52	127%
Total Transaction	**450**	**450**	**14767**	**6465**	128%	**4782**	**3962**	21%	**33994**	**22430**	52%	**6049.08**	**2138.31**	183%
Total	2250	2250	2953	1293	128%	334	110	204%	8065	5626	43%	1532.29	769.09	99%

Comparing the Average Total Transaction time:

Test 5 **14.767** seconds
Test 11 **6.465** seconds
Difference **8.302** seconds (128% Better Performance for Test 11!)

This shows us a fantastic gain in performance when comparing the tweaked test results from Test 11 to the Test 5 results. This definitely confirms that we have found and resolved a major bottleneck in our environment.

If we compare the Test 2 (50 ACU) results with Test 11, we see that the results are very close again.

Test 2	**5.831**	seconds
Test 11	**6.465**	seconds
Difference	**- 0.634**	seconds (Test 2 about 10% faster than Test 11 but was within our 10-15% deviance)

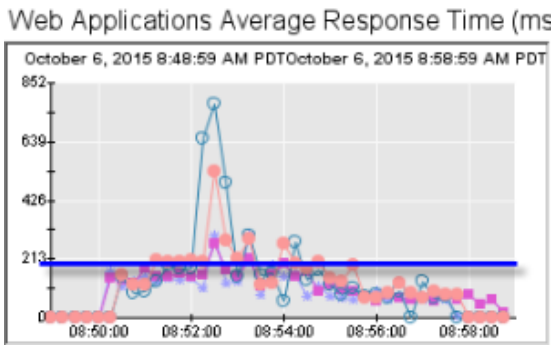
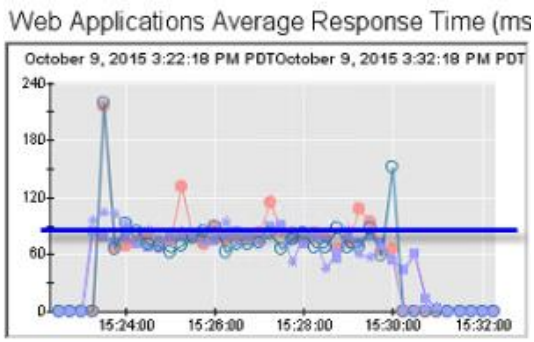
CA Introscope Output

We have collected the same 14 dashboard screenshots as we had with the previous tests. We will compare a few of these dashboards to Test 5 to see if we can easily tell a few areas where our tweaks have helped reduce response time.

- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

BOE 4.0 Triage Dashboard

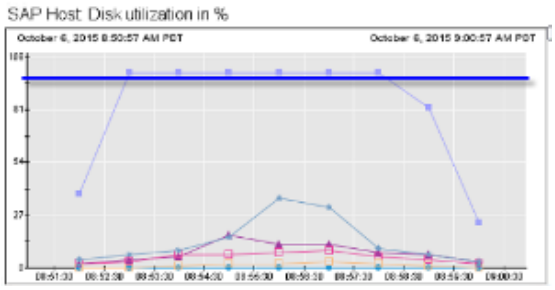
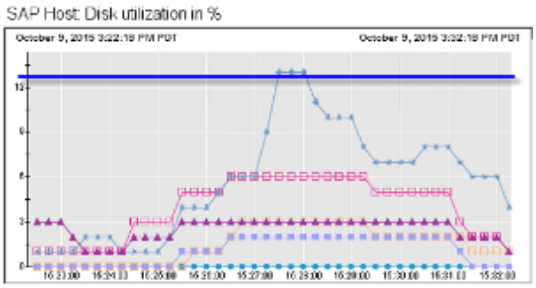
Web Applications Average Response Time (ms)

Test 5 – 150 ACU	Test 11 – 150 ACU
	

Comparison Notes: Test 5 had an average response time of around 200ms whereas Test 11 showed an average of about 90ms. That is over 100% increase in response time between tests. This definitely falls in line with what we were seeing in the JMeter tests.

Host Triage Dashboard

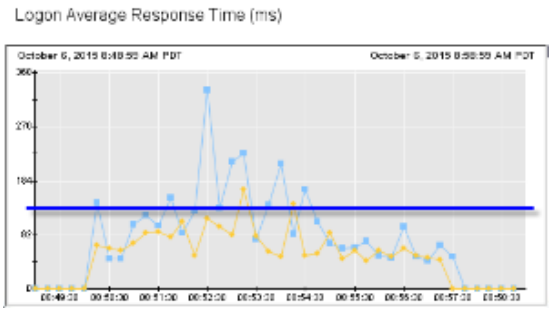
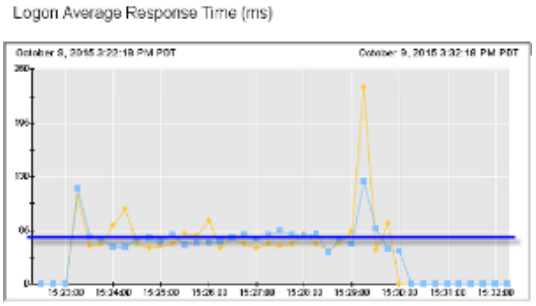
SAP Host: Disk Utilization in %

Test 5 – 150 ACU	Test 11 – 150 ACU
	

Comparison Notes: We can see that Disk Utilization has settled down a lot for at least one of our servers.It went from 100% to around 14% after our changes. All the other servers are showing lower averages as well.

Warning: The time interval shown in these two screenshots is different due to CA Introscopes default settings. After 2 weeks, it will start aggregating data into bigger time intervals to save space in the database layer. The left screenshot shows 1 minute intervals/averages whereas the right screenshot shows 15 minute intervals

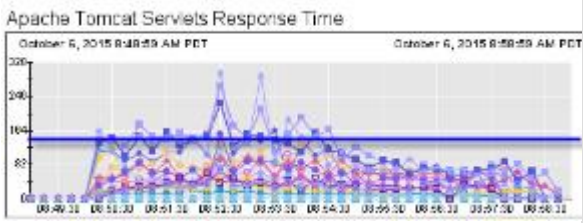
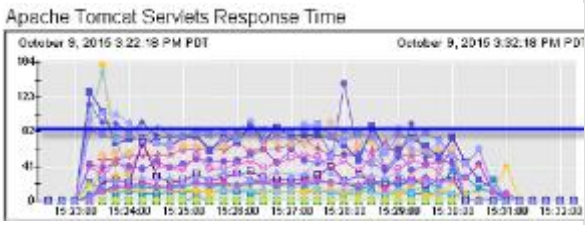
BOE 4.0 Platform Dashboard
Logon Average Response Time (ms)

Test 5 – 150 ACU	Test 11 – 150 ACU
	

Comparison Notes: Test 11 showed a much more consistent average logon response time that was sitting at around 60ms. Test 5 was all over the board with an average somewhere in the range of about 100-120ms. Our tweaked test (11) definitely outperformed for Logon ART.

Apache Tomcat Servlets

Apache Tomcat Servlets Response Time

Test 5 – 150 ACU	Test 11 – 150 ACU
	

Comparison Notes: Test 11 showed a much more consistent average logon response time that was sitting at around 60ms. Test 5 was all over the board with an average somewhere in the range of about 100-120ms. Our tweaked test (11) definitely outperformed for Logon ART.

Test 11 Summary

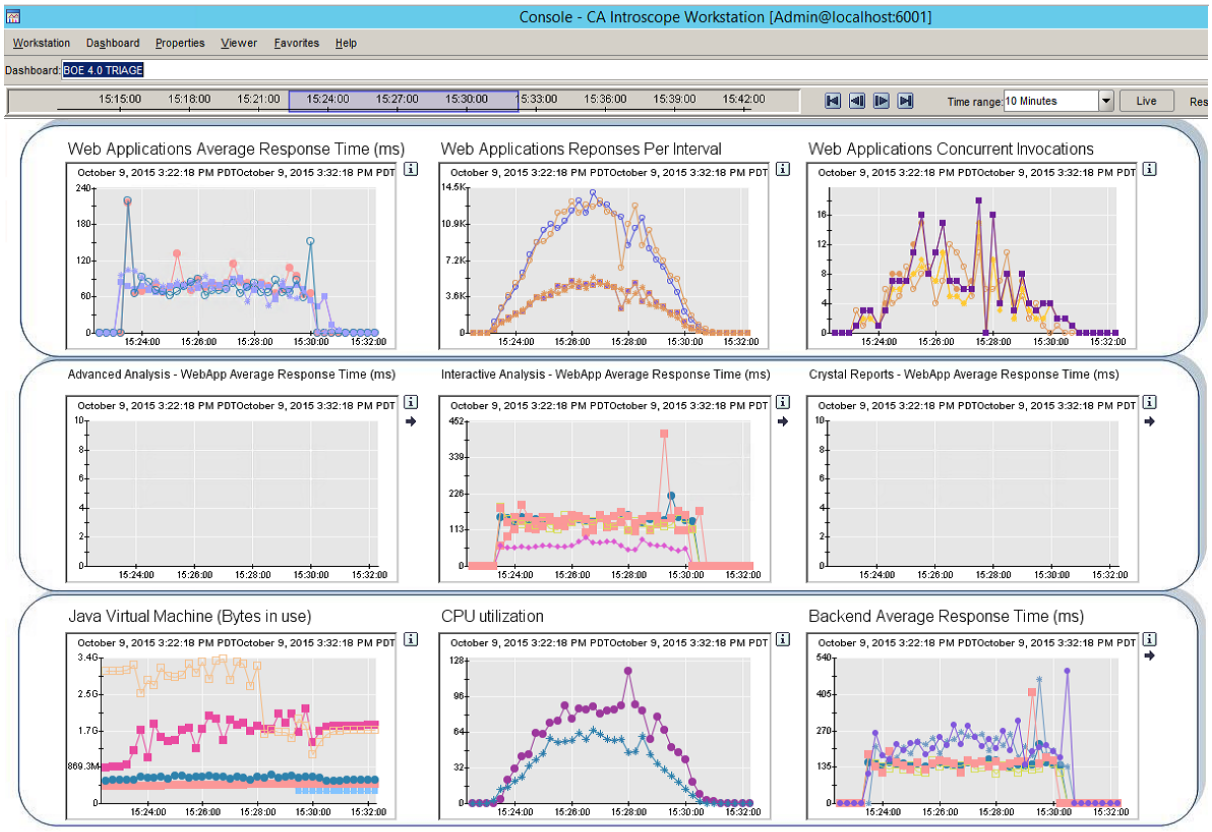
As you can see from the above data, we were able to run a test with 150 ACU at roughly the same speed as a 50 or 100 ACU test after a few tweaks. Furthermore, we were able to validate that our changes made a difference using this performance testing environment. This can be used to validate the impact of future changes to our environment as well.

Resources for Test 11

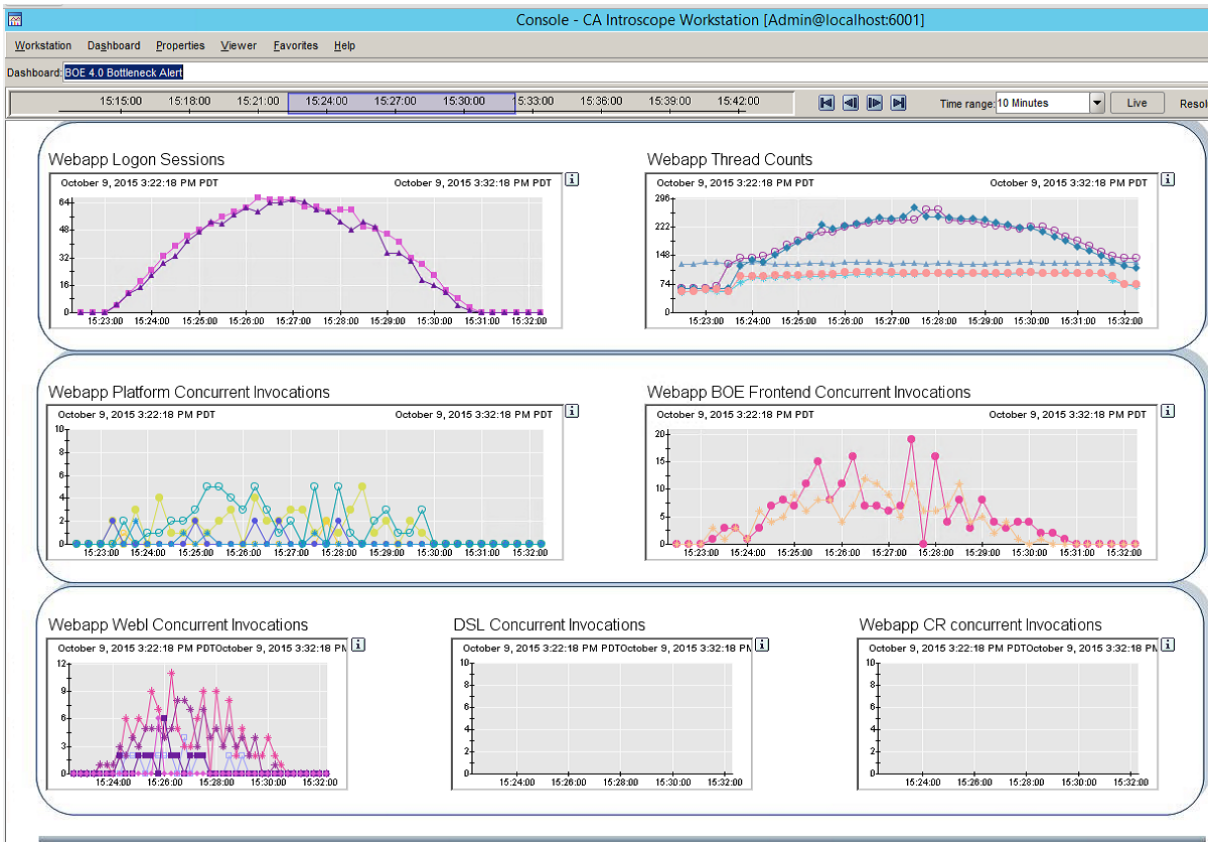
Tomcat CPU Usage - 150 ACU LB Changes

Platform Averages - 150 ACU LB Changes

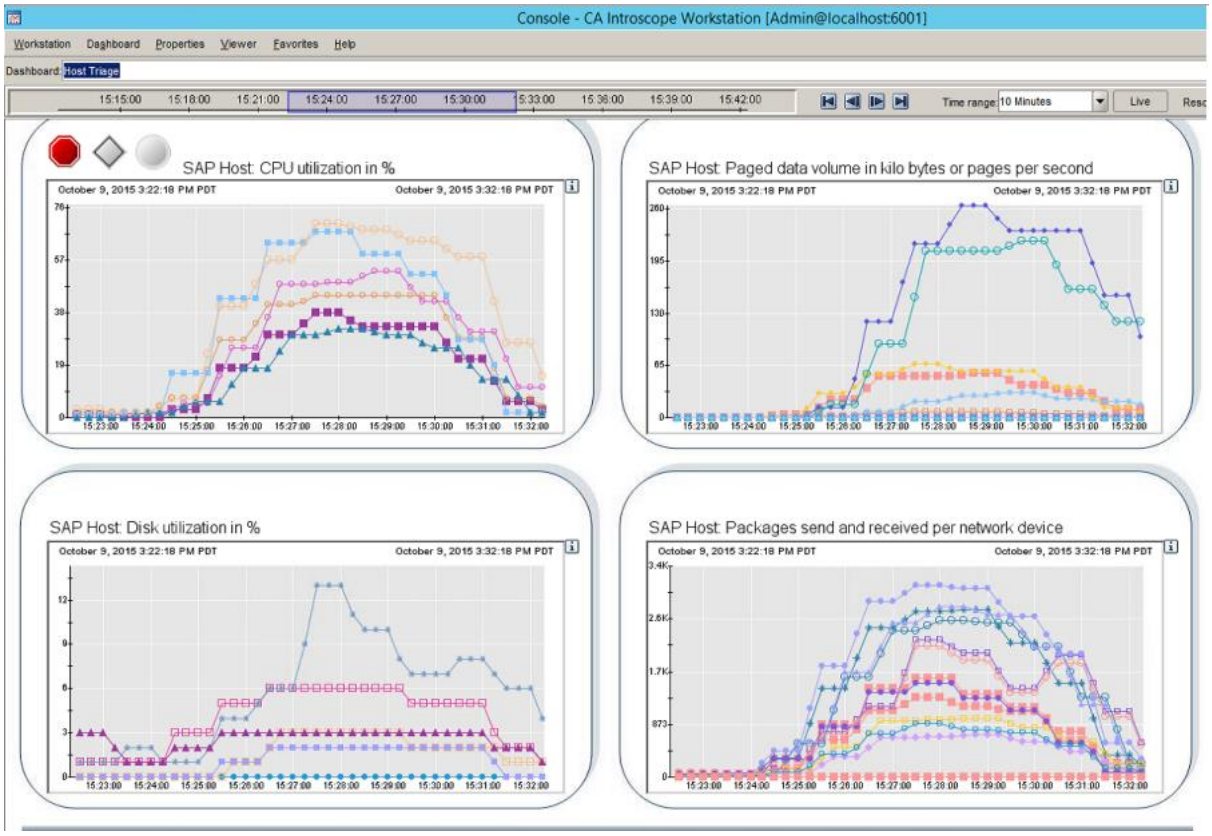
BI Triage - 150 ACU LB Changes



Bottleneck Alert - 150 ACU LB Changes

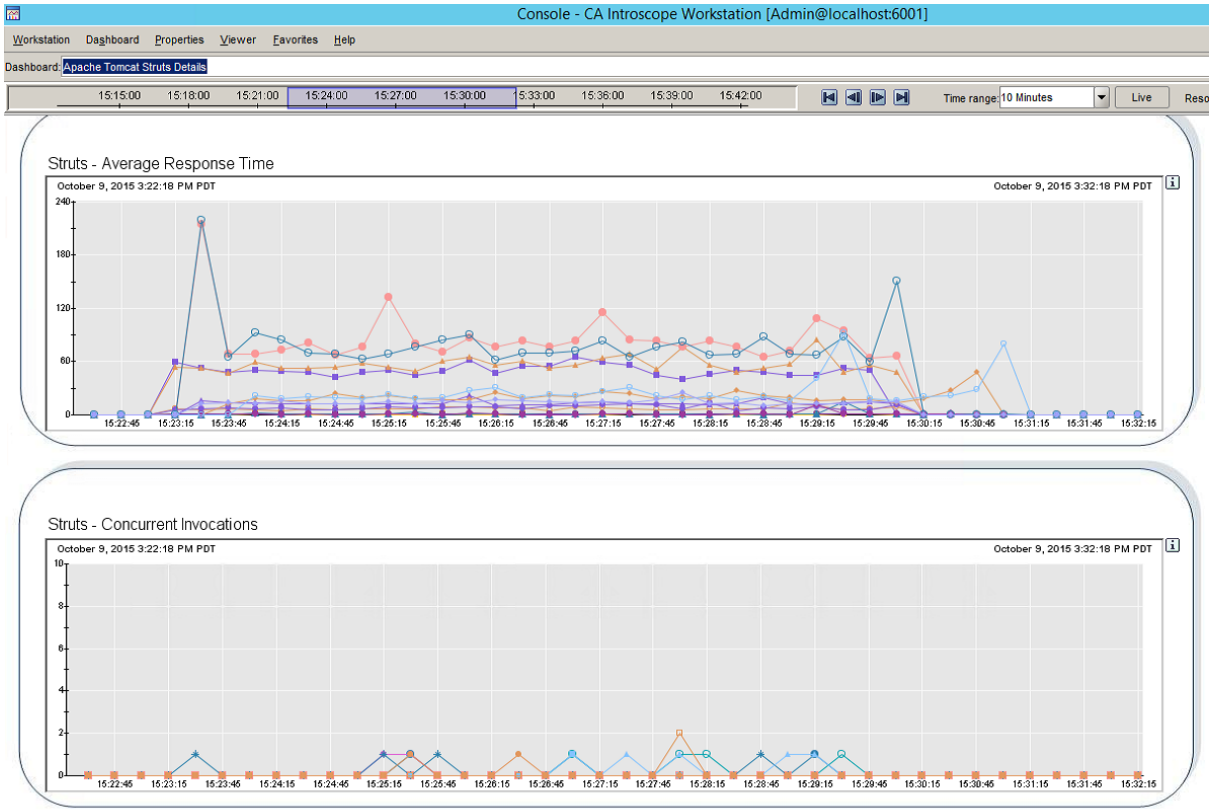


Host Triage - 150 ACU LB Changes-0000



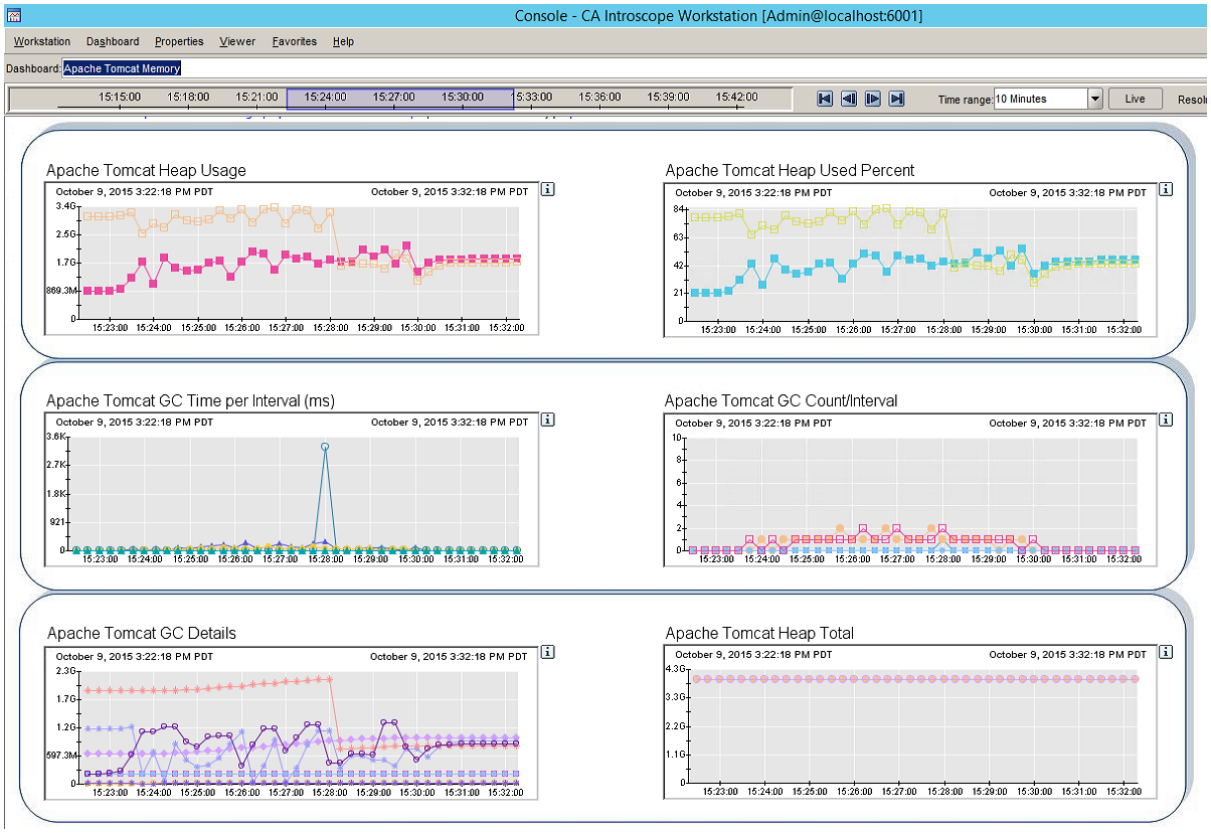
Tomcat Threads - 150 ACU LB Changes

Tomcat Strut - 150 ACU LB Changes

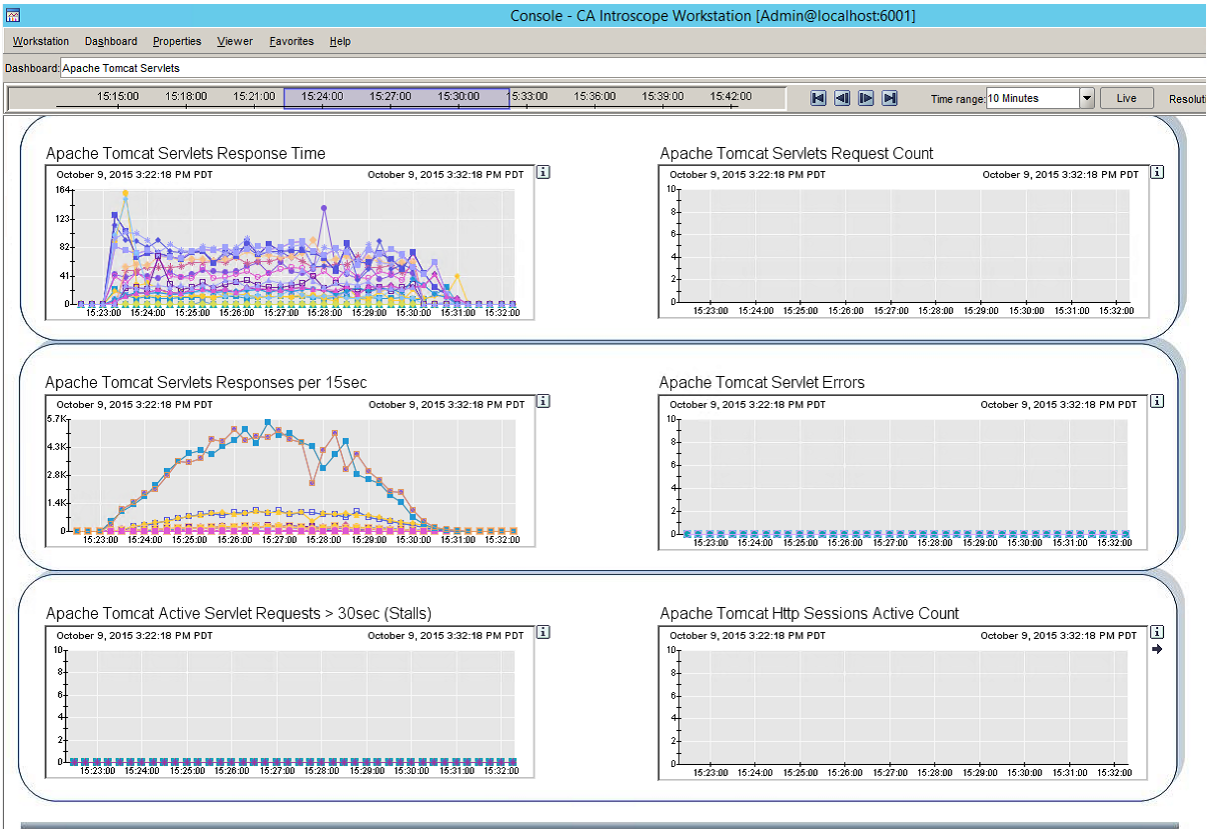


Tomcat GC Details - 150 ACU LB Changes

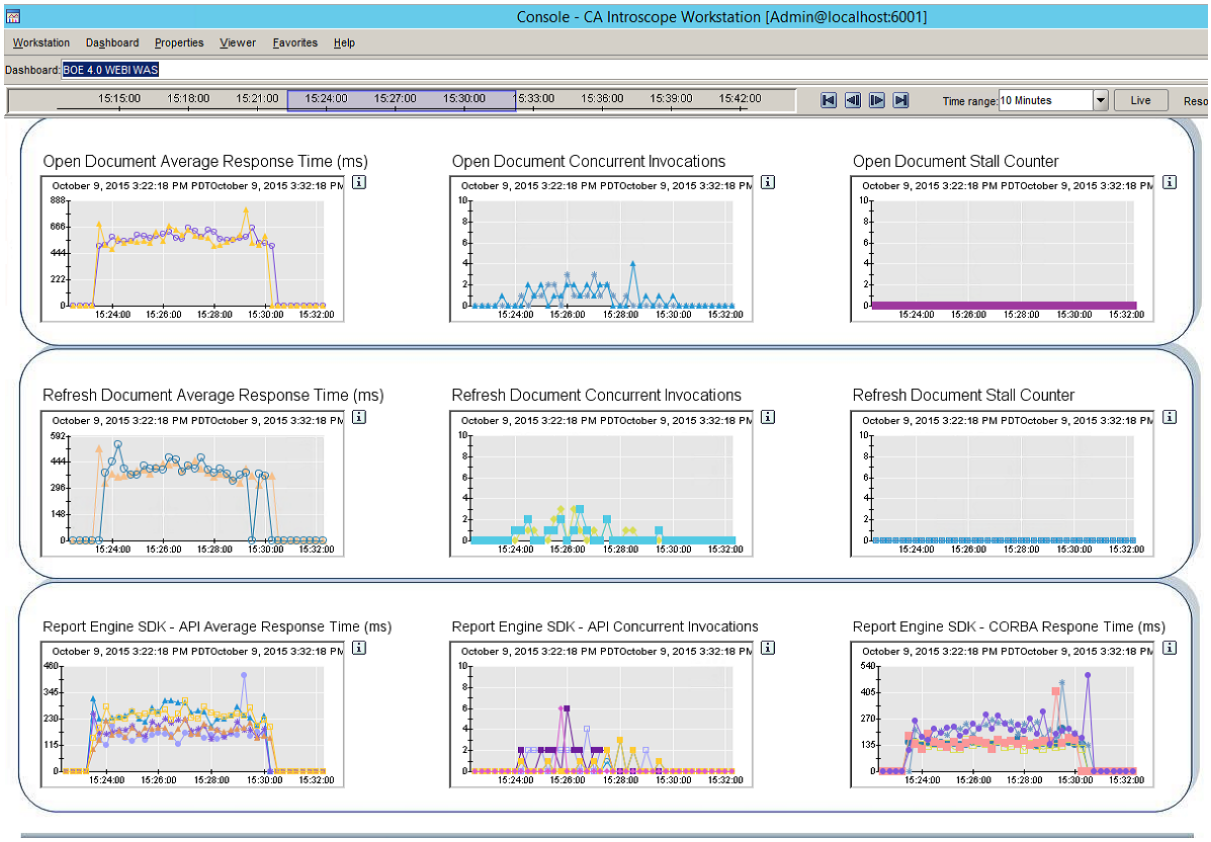
Tomcat Memory - 150 ACU LB Changes



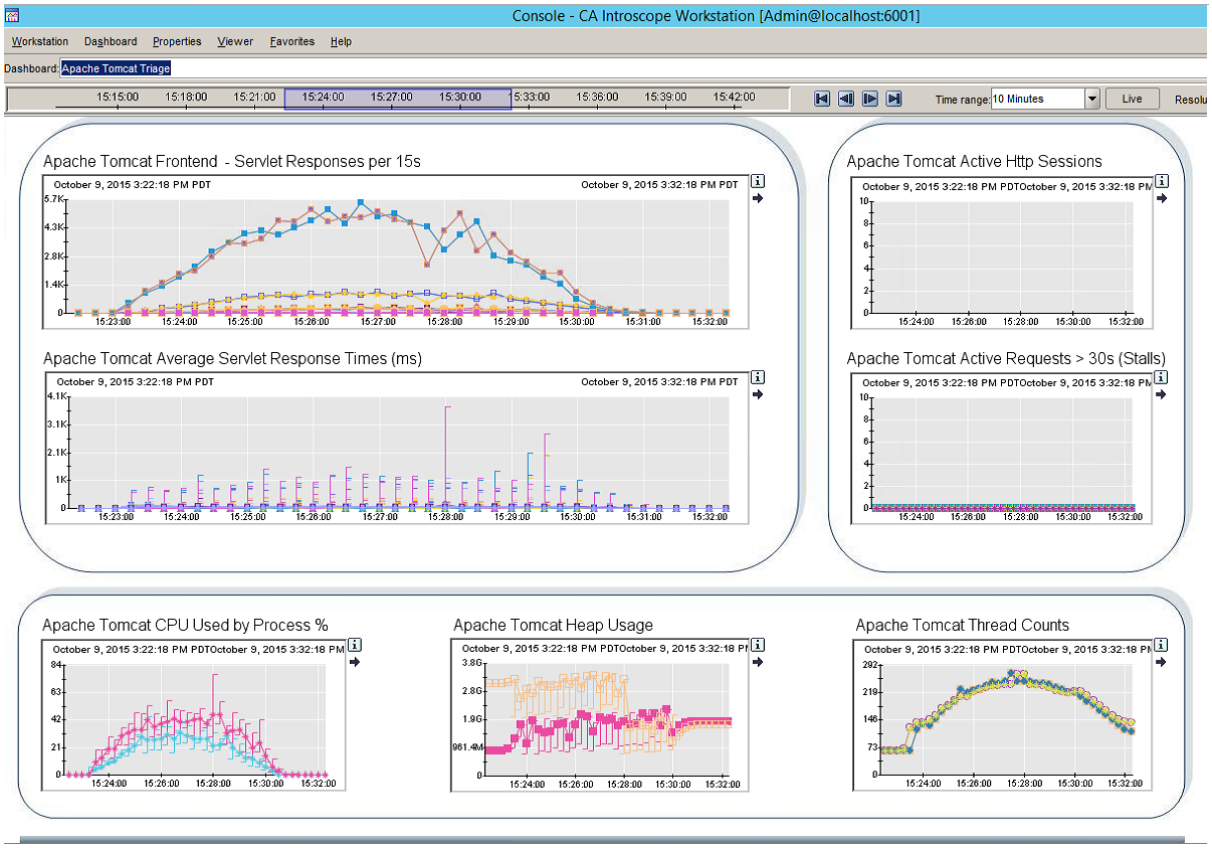
Tomcat Servlets - 150 ACU LB Changes



Webi WAS - 150 ACU LB Changes



Tomcat Triage - 150 ACU LB Changes



Webi Server Details - 150 ACU LB Changes-0001

Webi Server Overview - 150 ACU LB Changes

Test 12- Screenshots and Data

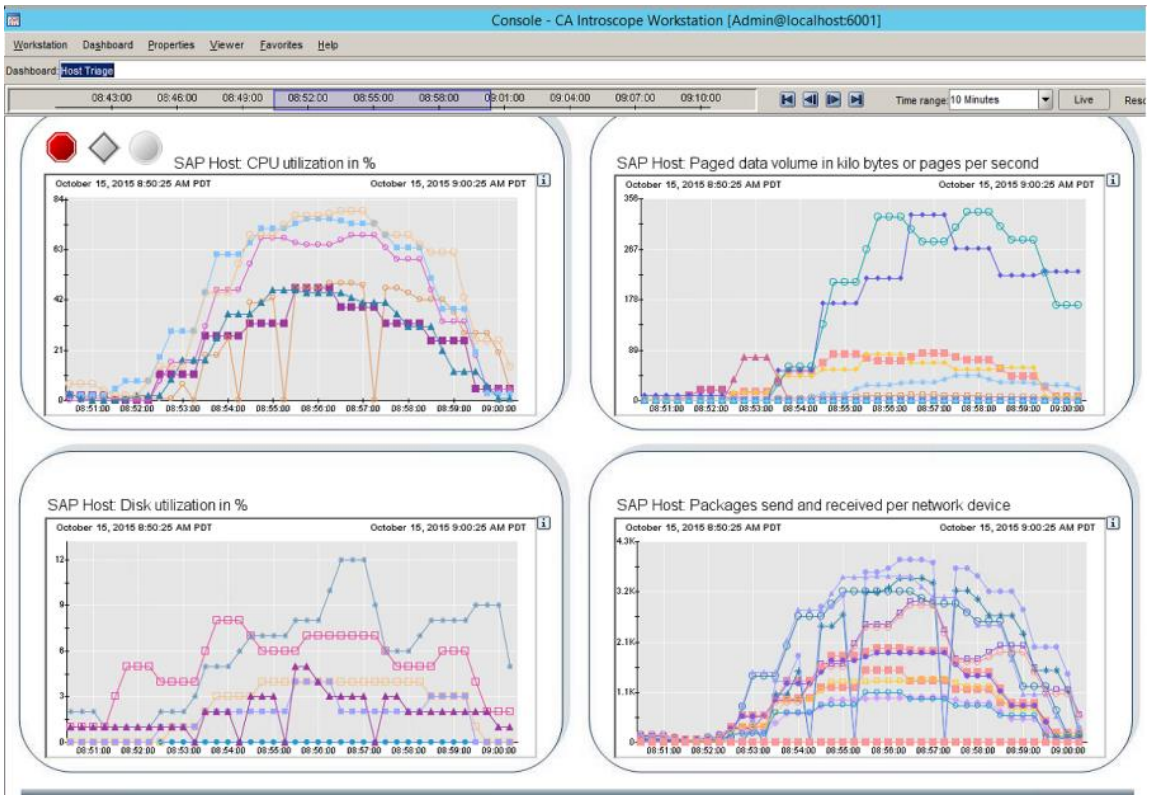
We have provided the details of Test 12 below, but have not done the analysis in this pattern book. Feel free to download this data to take a look if you are interested in practicing.

Test 12	**80 ACU Edit / 400 Transactions / 120 Second Ramp-up** **120 ACU View / 600 Transactions / 240 Second Ramp-up**
Active Concurrent Users	200
Total Transactions	1000
Start Time	8:51 AM
End Time	8:58 AM
Duration	~7 Minutes

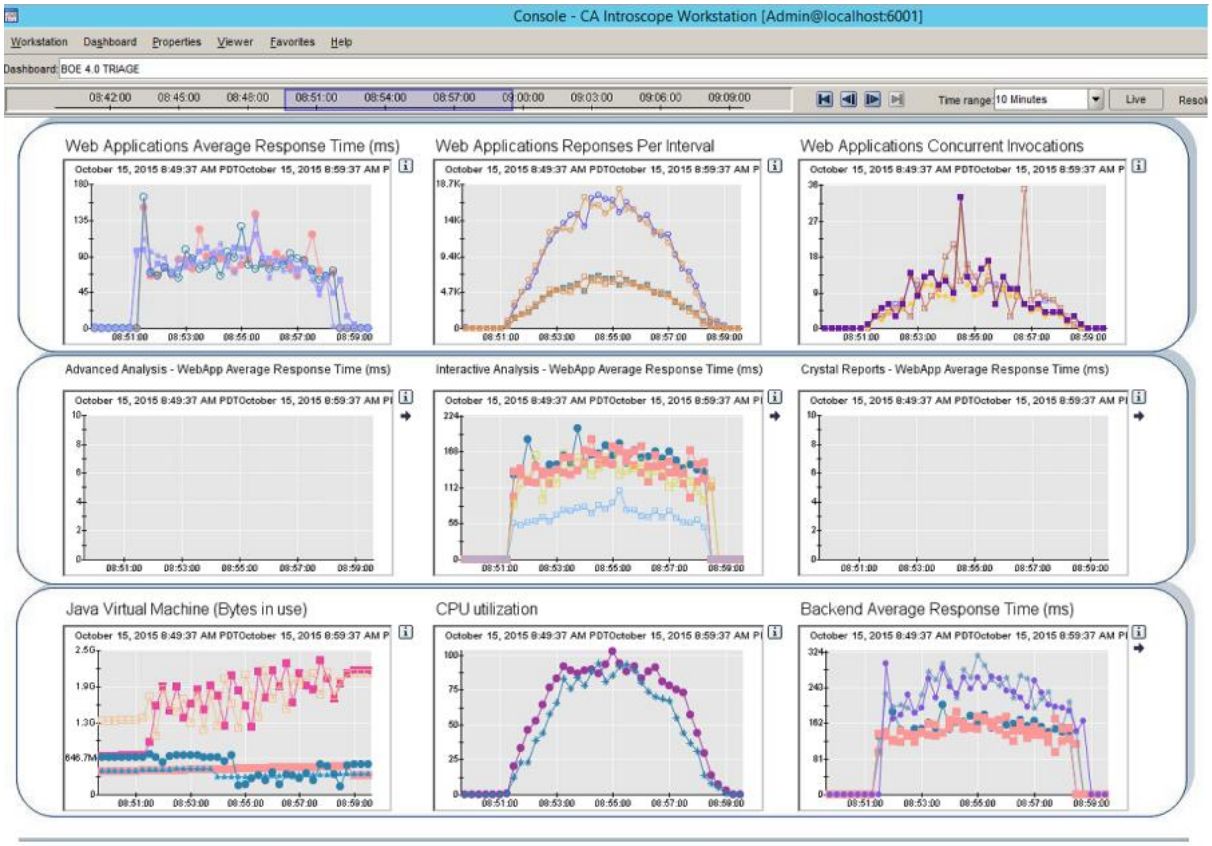
Information: The resource download includes the CA Introscope screenshots as well as the JMeter output.

Tomcat CPU Usage - 200 ACU server config

Host Triage - 200 ACU server config

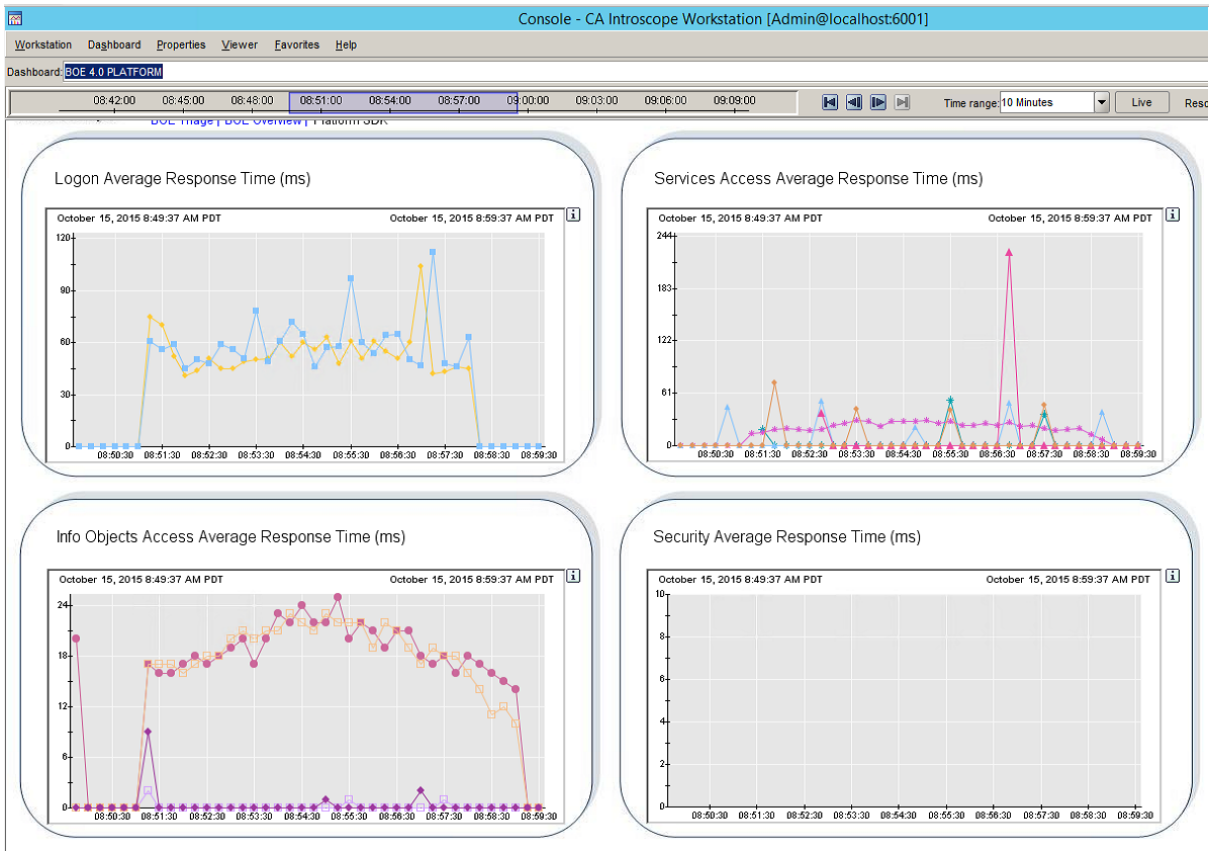


BI Triage - 200 ACU server config

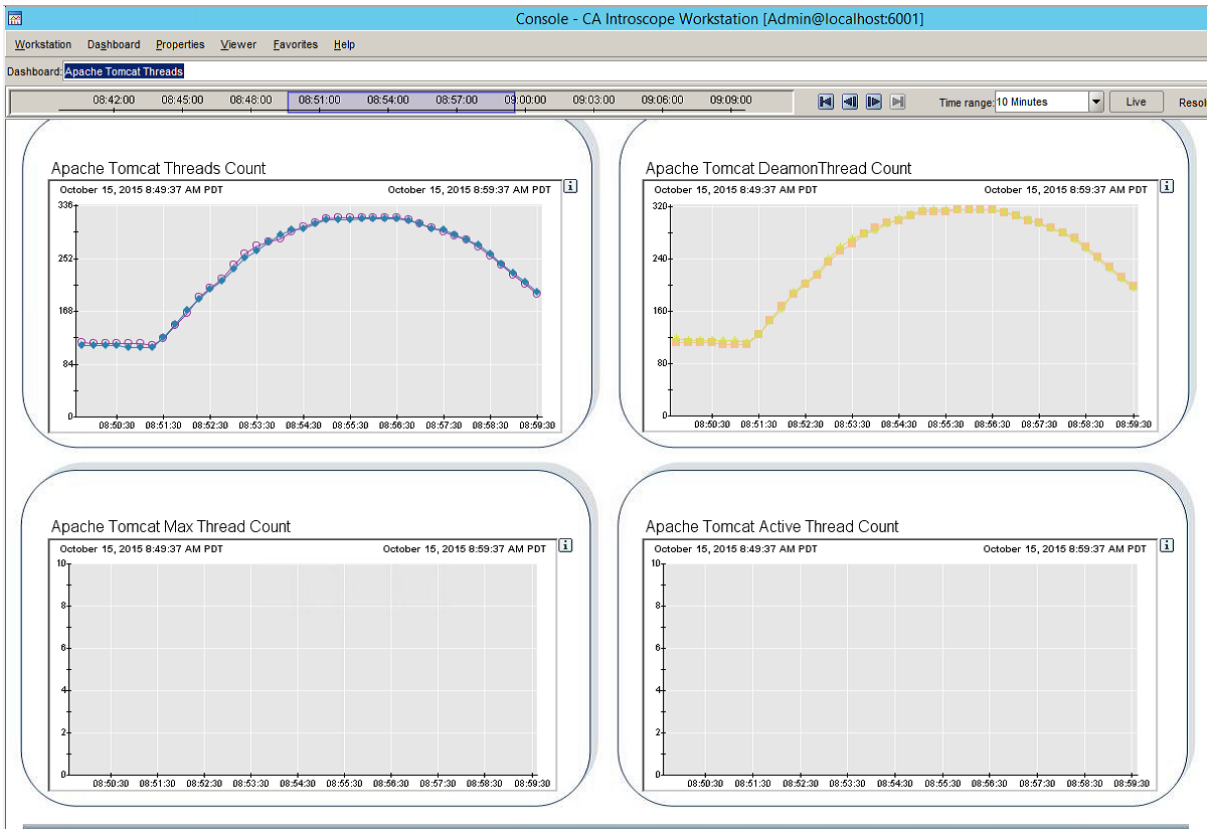


Bottleneck Alert - 200 ACU server config

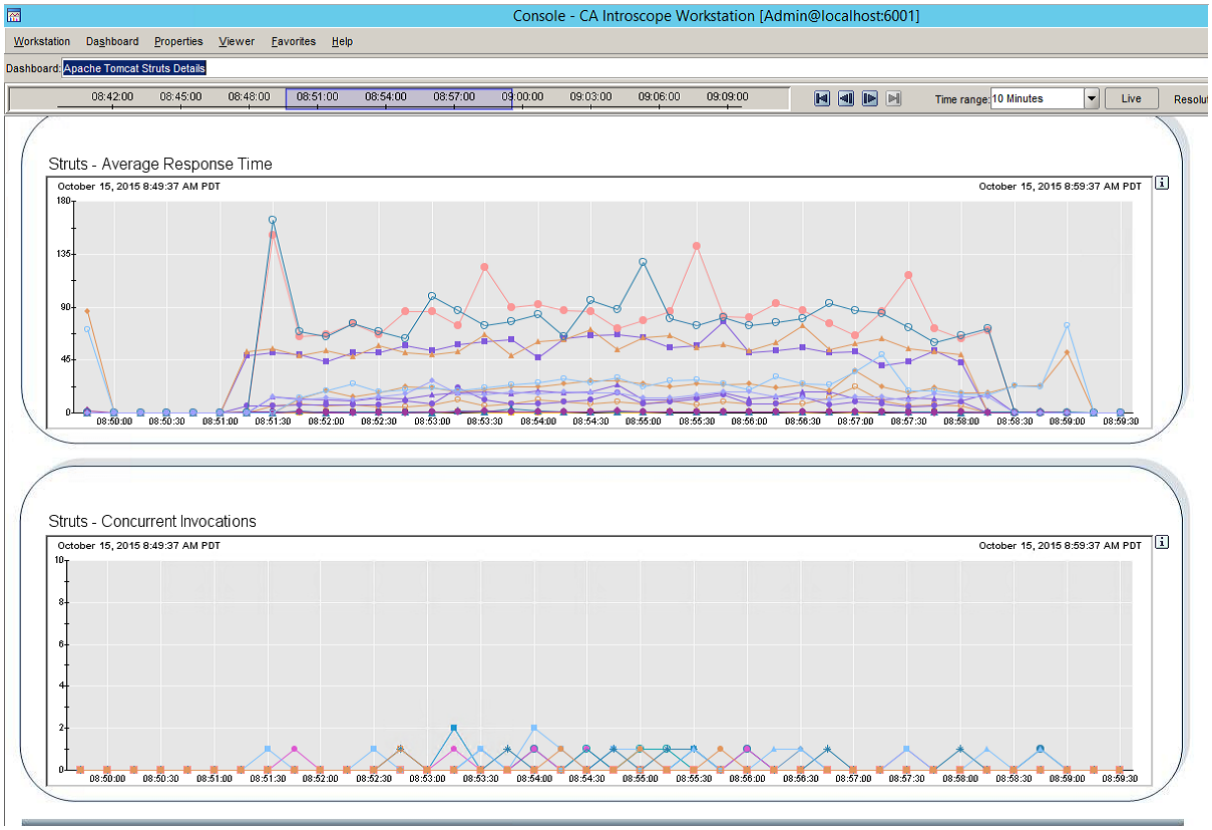
Platform Averages - 200 ACU server config



Tomcat Threads - 200 ACU server config

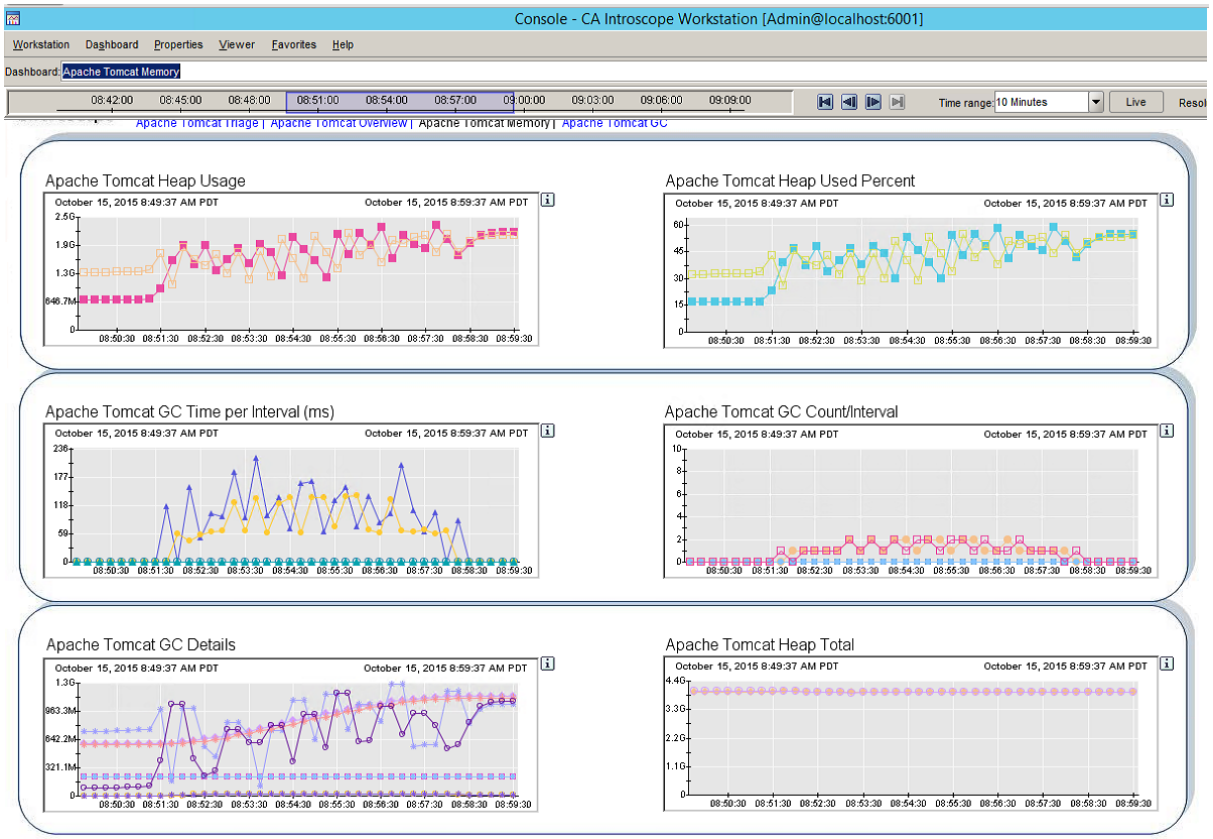


Tomcat Strut - 200 ACU server config

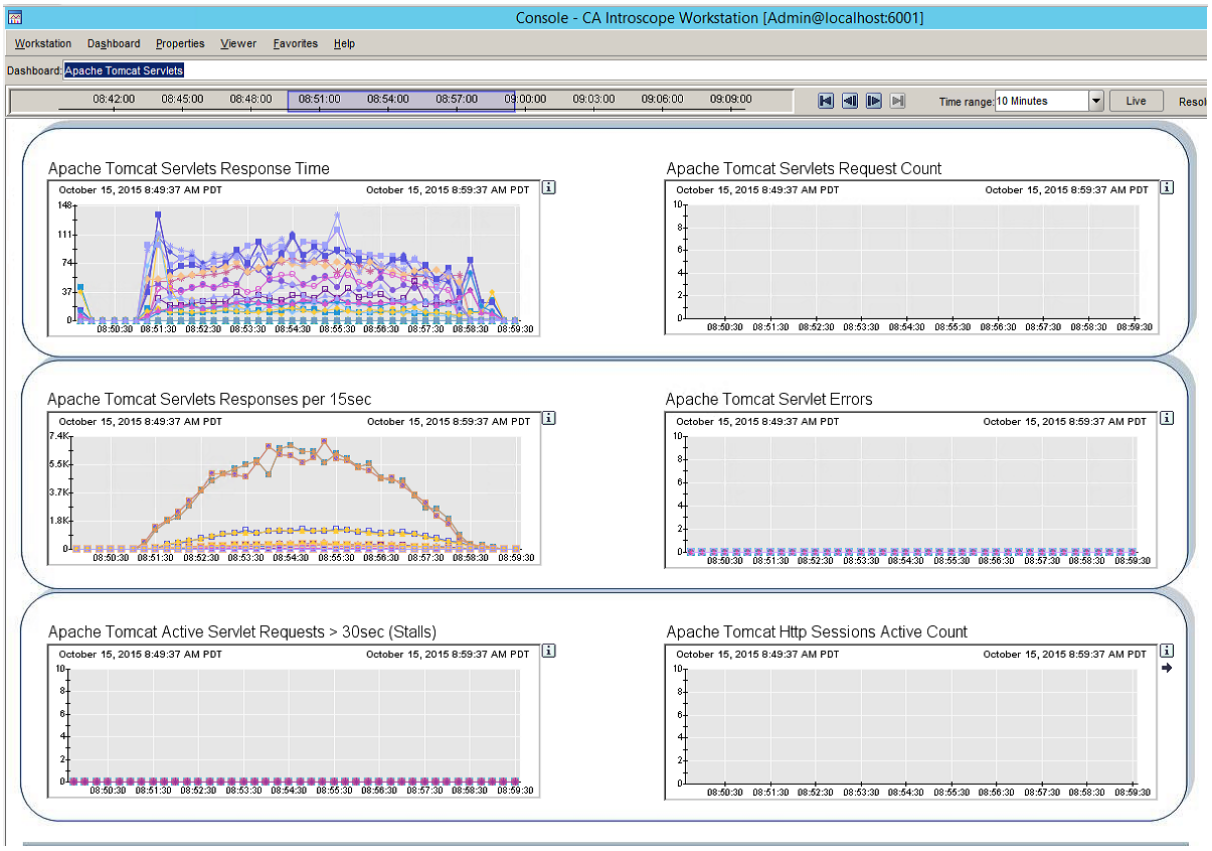


Tomcat GC Details - 200 ACU server config

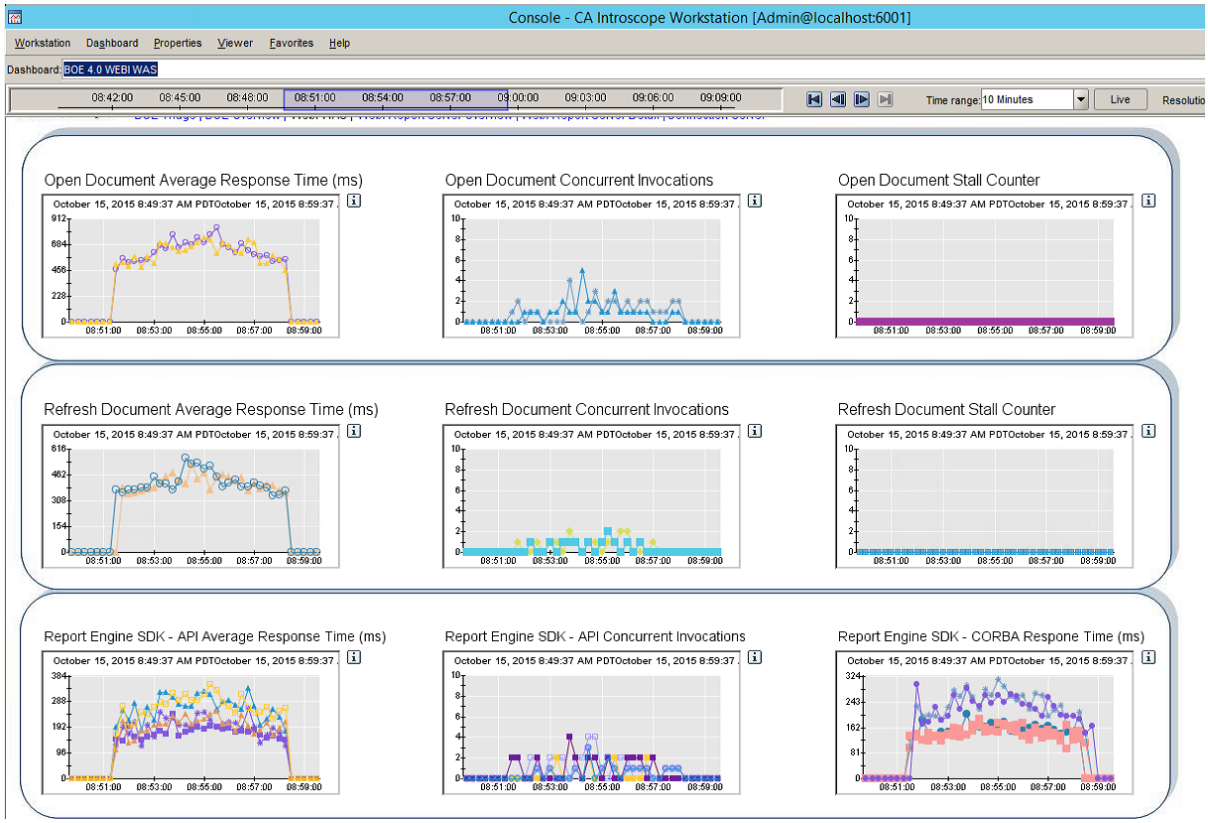
Tomcat Memory - 200 ACU server config



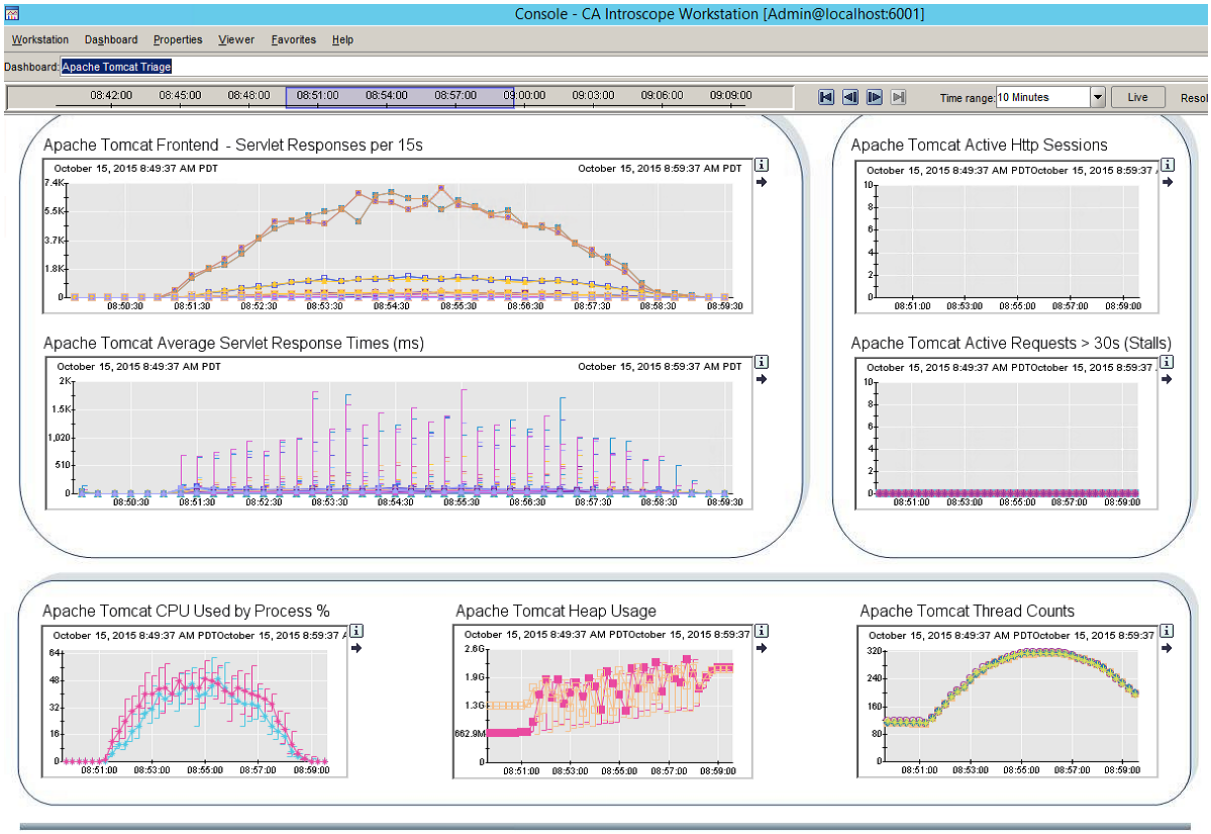
Tomcat Servlets - 200 ACU server config



Webi WAS - 200 ACU server config

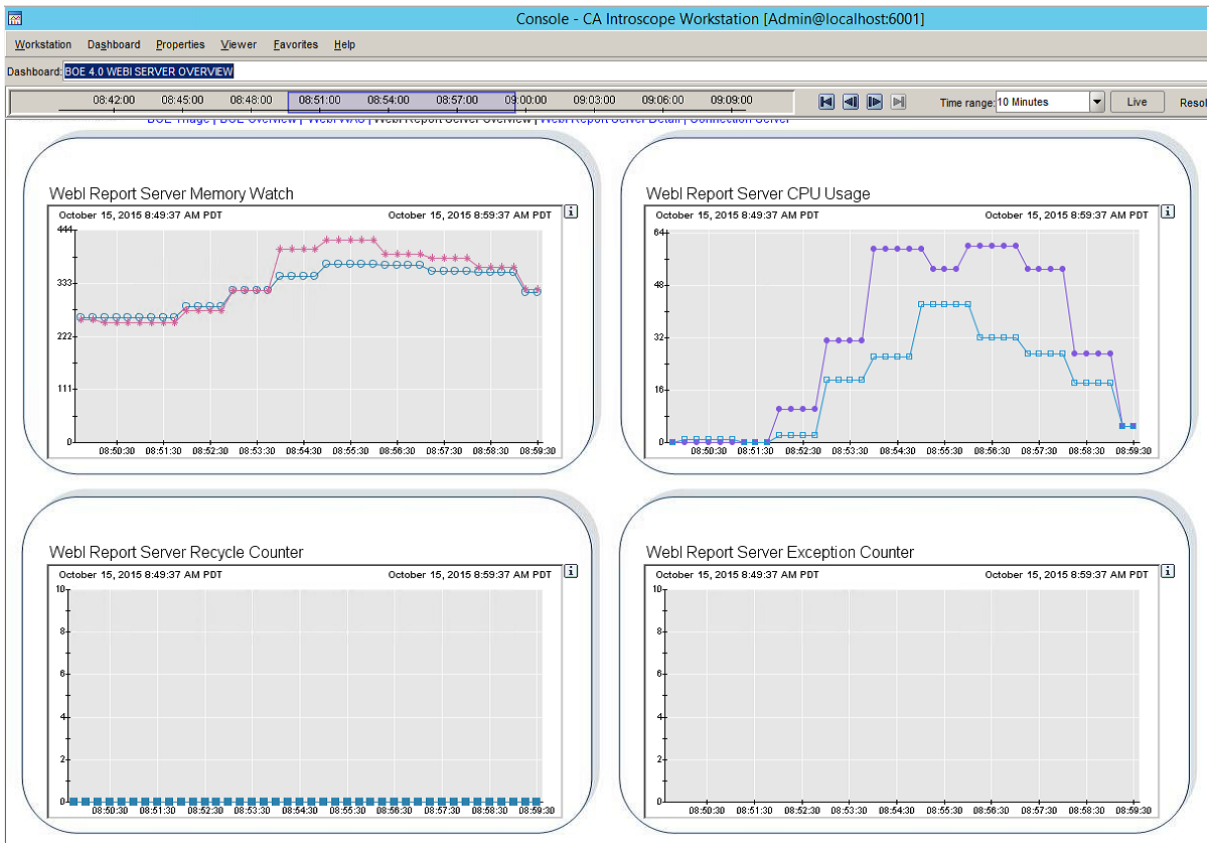


Tomcat Triage - 200 ACU server config



Webi Server Details - 200 ACU server config

Webi Server Overview - 200 ACU server config



Jmeter View Output

Name: View and Refresh Webi Report

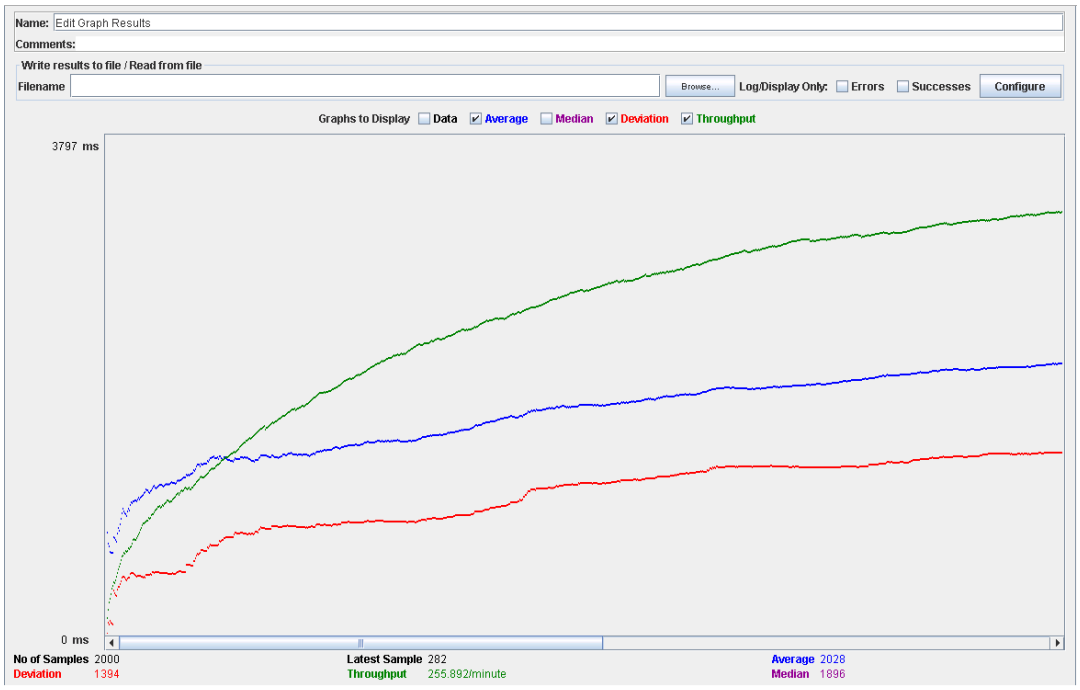
Comments:

Write results to file / Read from file

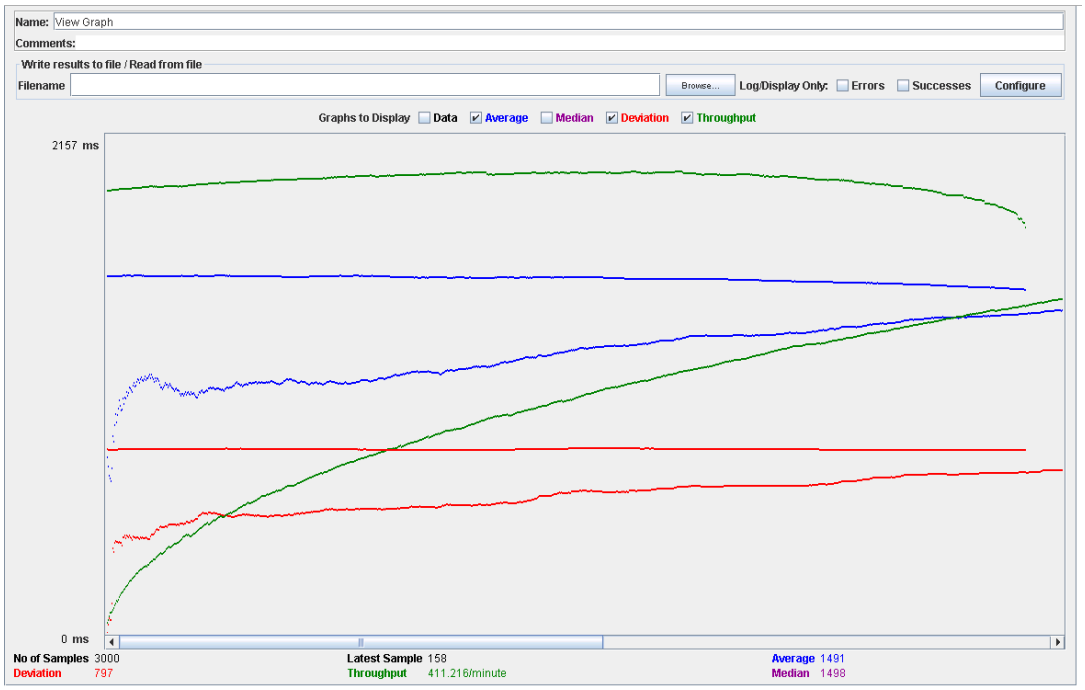
Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	600	1253	473	2923	415.20	0.00%	1.5/sec	621.29	429064.2
Browse Folders	600	2349	1373	4594	526.29	0.00%	1.5/sec	559.29	390472.2
View Report	600	1396	723	3498	388.18	0.00%	1.5/sec	349.94	239968.3
Refresh and Prom...	600	2062	1252	5013	459.35	0.00%	1.5/sec	369.85	255696.9
Close and Logout	600	392	99	1643	177.86	0.00%	1.5/sec	47.46	33148.7
TOTAL	3000	1491	99	5013	797.18	0.00%	6.9/sec	1804.89	269670.1

Jmeter Edit Output Graph



JMeter View Output Graph



JMeter Output Edit Graph

Name: Edit Web Intelligence Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	400	1112	486	2556	374.75	0.00%	58.2/min	406.52	428991.3
Browse Folders	400	2243	1289	4213	530.95	0.00%	58.1/min	369.55	390472.6
View Report	400	2415	1282	4919	641.95	0.00%	57.3/min	887.67	952053.0
Edit and Refresh	400	4060	2558	7237	865.35	0.00%	55.5/min	499.05	552423.8
Close and Logout	400	310	83	1675	167.99	0.00%	58.6/min	29.49	30913.0
TOTAL	2000	2028	83	7237	1394.68	0.00%	4.3/sec	1961.55	470970.7

Test 13- 300 Active Concurrent Users

In Test 11, we were able to run 150 ACUs at the same performance levels as a 50 or 100 ACU test. We then did 200 ACU in Test 12 to see how that went. The Test 12 went well so we decided to up the ante to 300 ACU for Test 13. Before doing so, we had to ensure that our Load Balancer "MaxClients" was increased to 300 to accommodate the maximum Active Concurrent Users. After doing that, we were ready to test this system. Below are the results.

Test 13	**120 ACU Edit / 600 Transactions / 120 Second Ramp-up** **180 ACU View / 900 Transactions / 240 Second Ramp-up**
Active Concurrent Users	300
Total Transactions	1500
Start Time	9:36 AM
End Time	9:45 AM
Duration	~9 Minutes

JMeter Output

Edit Workflow:

Name: Edit Web Intelligence Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	600	2312	496	6724	1190.04	0.00%	1.3/sec	543.55	428992.0
Browse Folders	600	3995	1378	9568	1708.63	0.00%	1.3/sec	489.72	390469.2
View Report	600	4193	1373	11638	1805.95	0.00%	1.3/sec	1172.64	952053.0
Edit and Refresh	600	5802	2562	11890	1941.41	0.00%	1.2/sec	664.41	552422.1
Close and Logout	600	613	109	3144	419.05	0.00%	1.3/sec	38.80	30913.0
TOTAL	3000	3383	109	11890	2334.30	0.00%	5.7/sec	2628.85	470969.9

For the Edit workflow, we see the following values:

	# Samples	Average	Min	Max	Std Dev
Login	600	2312	496	6724	1190.04
Browse Folders	600	3995	1378	9568	1708.63
View Report	600	4193	1373	11638	1805.95
Edit and Refresh	600	5802	2562	11890	1941.41
Close and Logout	600	613	109	3144	419.05
Total Transaction	**600**	**16915**	**5918**	**42964**	**7065.08**
Average	3000	3383	109	11890	2334.30

These results appear to be similar to Test 5 (150 ACU before tweaking) so it is suspected that we see some sort of bottleneck hit again. This should show up somewhere in our CA Introscope dashboards.

Interpretation of the above test results

- The Average Total Transaction time is **16.915** seconds from the Edit workflow
- The fastest (Min) potential workflow is **5.918** seconds
- The slowest (Max) potential workflow is **42.964** seconds
- The slowest action is the **Edit and Refresh** action

At a glance, these look to be almost double what we have seen in our Test 11 results. We'll compare them below to see how accurate that is.

Comparing Test 13 to Test 11

Test 13 was an attempt to stress the system by adding 300 active concurrent users. The system is sized for less than this so we expect it should perform worse than our 150 ACU test that we did in Test 11. Below we compare the results side by side for Test 11 and Test 13 to see how they compare.

	Number of Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 11**	**Test 13**	**Test 11**	**Test 13**	**% Diff**	**Test 11**	**Test 13**	**% Diff**	**Test 11**	**Test 13**	**% Diff**	**Test 11**	**Test 13**	**% Diff**
Login	300	600	937	2312	147%	481	496	3%	4895	6724	37%	423	1190.04	181%
Browse Folders	300	600	1984	3995	101%	1259	1378	9%	6146	9568	56%	570.44	1708.63	200%
View Report	300	600	2059	4193	104%	1249	1373	10%	5834	11638	99%	646.09	1805.95	180%
Edit and Refresh	300	600	3666	5802	58%	2544	2562	1%	7686	11890	55%	893.08	1941.41	117%
Close and Logout	300	600	300	613	104%	92	109	18%	3822	3144	22%	403.14	419.05	4%
Total Transaction	**300**	**600**	**8946**	**16915**	**89%**	**5625**	**5918**	**5%**	**28383**	**42964**	**51%**	**2935.75**	**7065.08**	**141%**
Total	1500	3000	1789	3383	89%	92	109	18%	7686	11890	55%	1300.72	2334.3	79%

Below is an image of the above table with more advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
Edit Webi	**Test 11**	**Test 13**	**Test 11**	**Test 13**	**% Diff**	**Test 11**	**Test 13**	**%Diff**	**Test 11**	**Test 13**	**% Diff**	**Test 11**	**Test 13**	**% Diff**
Login	300	600	937	2312	147%	481	496	3%	4895	6724	37%	423	1190.04	181%
Browse Folders	300	600	1984	3995	101%	1259	1378	9%	6146	9568	56%	570.44	1708.63	200%
View Report	300	600	2059	4193	104%	1249	1373	10%	5834	11638	99%	646.09	1805.95	180%
Edit and Refresh	300	600	3666	5802	58%	2544	2562	1%	7686	11890	55%	893.08	1941.41	117%
Close and Logout	300	600	300	613	104%	92	109	18%	3822	3144	22%	403.14	419.05	4%
Total Transaction	**300**	**600**	**8946**	**16915**	**89%**	**5625**	**5918**	**5%**	**28383**	**42964**	**51%**	**2935.75**	**7065.08**	**141%**
Total	1500	3000	1789	3383	89%	92	109	18%	7686	11890	55%	1300.72	2334.3	79%

Test 13 performed worse than Test 11 in almost every action and metric. It is clear from these results that some sort of bottleneck is hit and performance is affected across the board.

Here is a comparison of the Average Total Transaction time between Test 11 and 13

Test 11 **8.946** seconds
Test 13 **16.915** seconds
Difference **7.969** seconds (89% worse performance than Test 11)

If we compare Test 13 to Test 5 (150 ACU, before the system tweaks), we can see they are similarly aligned.

Test 5 **15.499** seconds
Test 13 **16.915** seconds
Difference **1.416** seconds (~9% faster results with Test 5!)

It would appear that our 300 ACU Test is performing worse than our 150 ACU test before the changes. This is still pretty good though as it indicates we may have come close to doubling out throughput with a couple of core changes. We know that 300 ACU is outside of

our system sizing anyways so this result was expected. Let's see how the View Workflow stacked up next.

View Workflow:

Name: View and Refresh Webi Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	900	2736	515	8964	1318.04	0.00%	2.0/sec	851.60	429064.1
Browse Folders	900	4509	1290	10632	1850.34	0.00%	2.0/sec	769.07	390470.0
View Report	900	2465	807	6645	979.43	0.00%	2.0/sec	477.69	239968.4
Refresh and Prom...	900	3465	1252	7673	1168.12	0.00%	2.0/sec	507.22	255697.4
Close and Logout	900	812	144	4383	548.10	0.00%	2.0/sec	65.36	33149.0
TOTAL	4500	2797	144	10632	1744.62	0.00%	9.4/sec	2488.03	269669.8

For the View workflow, we see the following values:

View Webi	# Samples	Average	Min	Max	Std Dev
Login	900	2736	515	8964	1318.04
Browse Folders	900	4509	1290	10632	1850.34
View Report	900	2465	807	6645	979.43
Refresh & Prompts	900	3465	1252	7673	1168.12
Close and Logout	900	812	144	4383	548.1
Total Transaction	**900**	**13987**	**4008**	**38297**	**5864.03**
Average	4500	2797	144	10632	1744.62

Interpretation of the above test results:

- The Average Total Transaction time was **13.987** seconds
- The Fastest potential Transaction time could be **4.008** seconds
- The Slowest potential Transaction time could be **38.297** seconds
- The slowest action on average is the Browse Folders action at **4.509** seconds

Comparing Test 13 to Test 11

Below is a side by side comparison of our Test 11 (150 ACU tweaked system) vs Test 13 (300 ACU tweaked system)

	Number of Samples		Average			Min			Max			Std Dev		
View Webi	Test 11	Test 13	Test 11	Test 13	% Diff	Test 11	Test 13	% Diff	Test 11	Test 13	% Diff	Test 11	Test 13	% Diff
Login	450	900	993	2736	176%	482	515	7%	4839	8964	85%	377.39	1318.04	249%
Browse Folders	450	900	2068	4509	118%	1256	1290	3%	5626	10632	89%	626.35	1850.34	195%
View Report	450	900	1181	2465	109%	735	807	10%	2594	6645	156%	242.14	979.43	304%
Refresh & Prompt	450	900	1880	3465	84%	1269	1252	1%	5528	7673	39%	538.91	1168.12	117%
Close and Logout	450	900	343	812	137%	220	144	53%	3843	4383	14%	353.52	548.1	55%
Total Transaction	**450**	**900**	**6465**	**13987**	**116%**	**3962**	**4008**	**1%**	**22430**	**38297**	**71%**	**2138.31**	**5864.03**	**174%**
Total	2250	4500	1293	2797	116%	110	144	31%	5626	10632	89%	769.09	1744.62	127%

Below is an image of the above table with more advanced formatting:

	# Samples		Average			Min			Max			Std Dev		
View Webi	Test 11	Test 13	Test 11	Test 13	% Diff	Test 11	Test 13	%Diff	Test 11	Test 13	% Diff	Test 11	Test 13	% Diff
Login	450	900	993	2736	176%	482	515	7%	4839	8964	85%	377.39	1318.04	249%
Browse Folders	450	900	2068	4509	118%	1256	1290	3%	5626	10632	89%	626.35	1850.34	195%
View Report	450	900	1181	2465	109%	735	807	10%	2594	6645	156%	242.14	979.43	304%
Refresh & Prompt	450	900	1880	3465	84%	1269	1252	1%	5528	7673	39%	538.91	1168.12	117%
Close and Logout	450	900	343	812	137%	220	144	53%	3843	4383	14%	353.52	548.1	55%
Total Transaction	**450**	**900**	**6465**	**13987**	**116%**	**3962**	**4008**	**1%**	**22430**	**38297**	**71%**	**2138.31**	**5864.03**	**174%**
Total	2250	4500	1293	2797	116%	110	144	31%	5626	10632	89%	769.09	1744.62	127%

Comparing the Average Total Transaction time:

Test 11 **6.465** seconds
Test 13 **13.987** seconds
Difference **7.522** seconds (116% worse performance on Test 13)

This shows us that Test 13 runs at about ½ the speed of Test 11 per transaction.

If we compare the Test 5 (150 ACU before tweaking the server) results with Test 13, we see that the results are similar

Test 5 **14.767** seconds
Test 13 **13.987** seconds
Difference **0.780** seconds (Test 13 ran slightly faster on average than test 5 but within our 15% deviance)

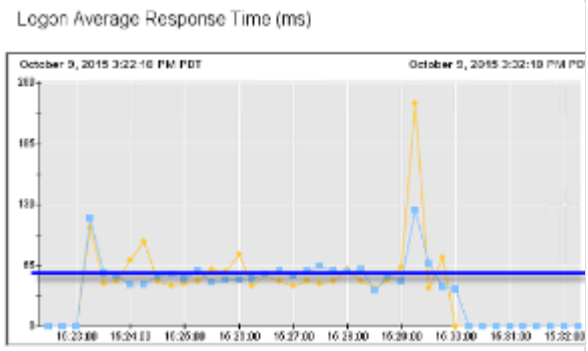
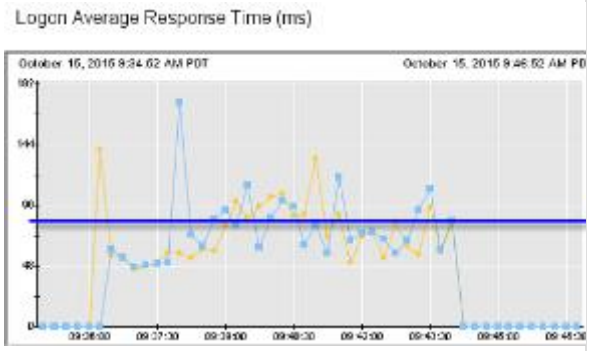
CA Introscope Output:

We have collected the same 14 dashboard screenshots as we had with the previous tests. We will compare a few of these dashboards to Test 11 to see if we can easily tell a few areas where our tweaks have helped reduce response time.

- Tomcat CPU
- Tomcat GC
- Tomcat Struts
- Tomcat Memory
- Tomcat Triage
- Tomcat Servlets
- Tomcat Threads
- BI 4.0 Bottleneck Alerts
- BI 4.0 Triage
- BI 4.0 Platform
- BI 4.0 Webi WAS
- BI 4.0 Webi Server Overview
- BI 4.0 Webi Server Details
- Host Triage

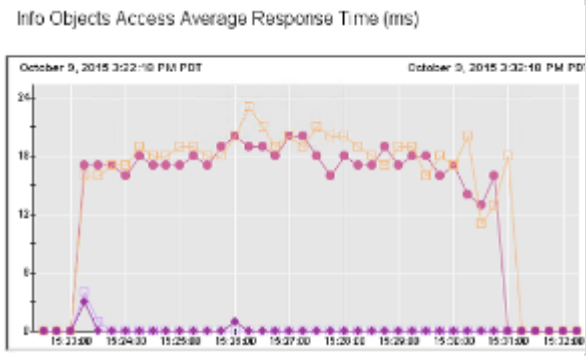
BOE 4.0 Platform Dashboard

Logon Average Response Time (ms)

Test 11 – 150 ACU	Test 13 – 300 ACU
	

Comparison Notes: Test 13 showed a significantly higher Logon ART. About a 50% degradation in performance for this metric.

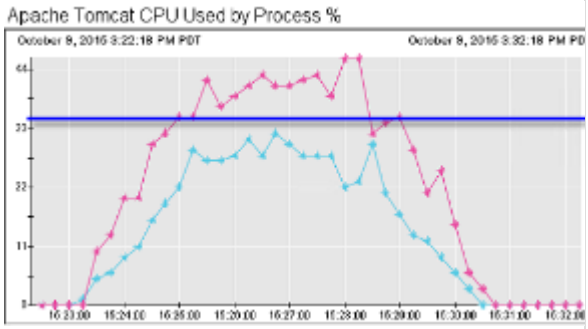
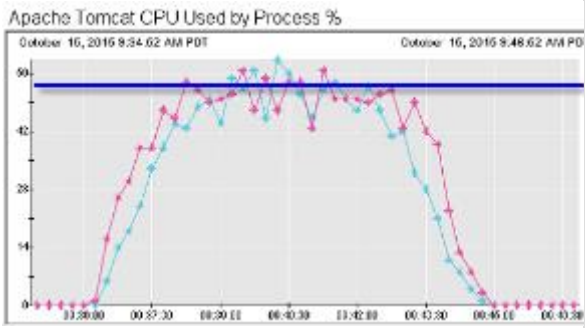
Info Objects Access Average Response Time (ms)

Test 11 – 150 ACU	Test 13 – 300 ACU
	

Comparison Notes: Test 13 showed a spike throughout the middle of test where Info Objects access was 100-200% slower than Test 11. This represents a bottleneck during the peak of the test when we had all 300 ACU on the system. Test 11 shows a nice stead ART throughout the whole test. This indicates a sizing issue with our environment.

Apache Tomcat CPU Dashboard

Apache Tomcat CPU Used Per Process %

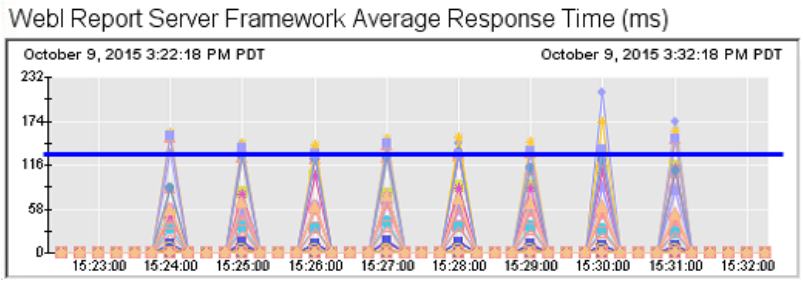
Test 11 – 150 ACU	Test 13 – 300 ACU
	

Comparison Notes: Comparing the CPU usage for our Tomcat boxes shows that it increased between tests but was still well within an acceptable range. This tells us that Tomcat was not likely our bottleneck.

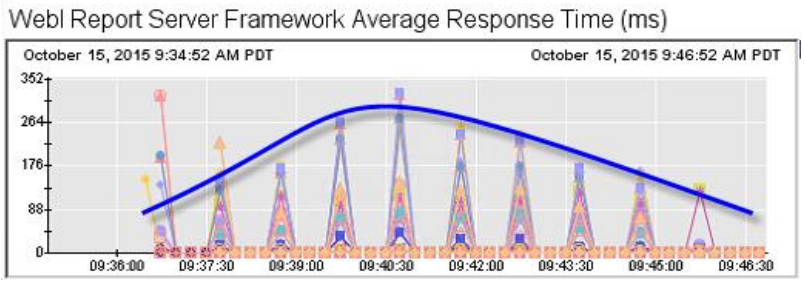
BOE 4.0 Webi Server Details Dashboard

Webi Report Server Framework Average Response Time (ms)

Test 11 – 150 ACU



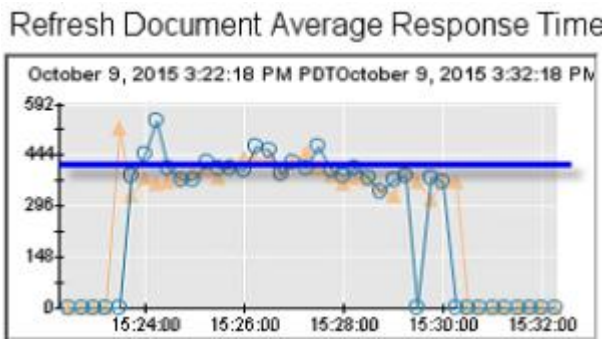
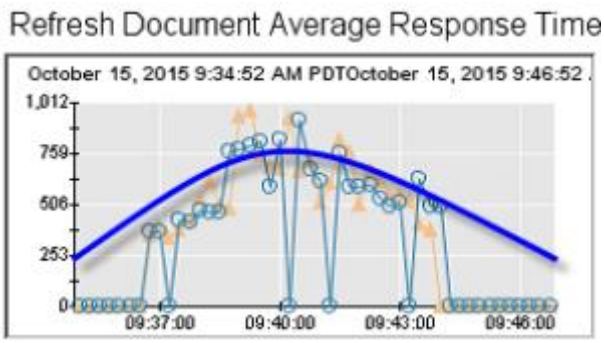
Test 13 – 300 ACU



Comparison Notes: Here is another great example of a load based bottleneck. We can see that for Test 11, the Webi Report Server Framework response time remained consistent across the test. Even at the peak of the load, the averages remained around the 120ms timeframe. For the 300 ACU test, we can see that during the peak of the load, the response time rose to almost triple the starting averages (88ms to 270+ms). This tells us we likely hit a resource limit on the backend server side.

BOE 4.0 Webi WAS Dashboard

Refresh Document Average Response Time (ms)

Test 11 – 150 ACU	Test 13 – 300 ACU
	

Comparison Notes: Another example of how the bottleneck appears in this comparison. We can see that Test 13 spikes up to over 2x the ART during the peak load time. Test 11 was consistent for the most part throughout the whole test. This data suggests a bottleneck at the Processing Tier.

Test 13 Summary

Many of the dashboards supported our theory that we hit a bottleneck when reaching our peak 300 ACU limit. Considering we sized this environment for 100 ACU originally, this is still pretty good. There are a lot of factors to consider though so it's not an exact science. The important take away from the test examples we have provided is that it is up to you to do your own benchmarking and load testing to determine what kind of performance to expect out of your system based on your usage. We were running 2 fairly simple Webi reports in these test plans so it is very likely that the result we will see with actual reports would vary.

Resources for Test 13

JMeter Edit Output

Name: Edit Web Intelligence Report

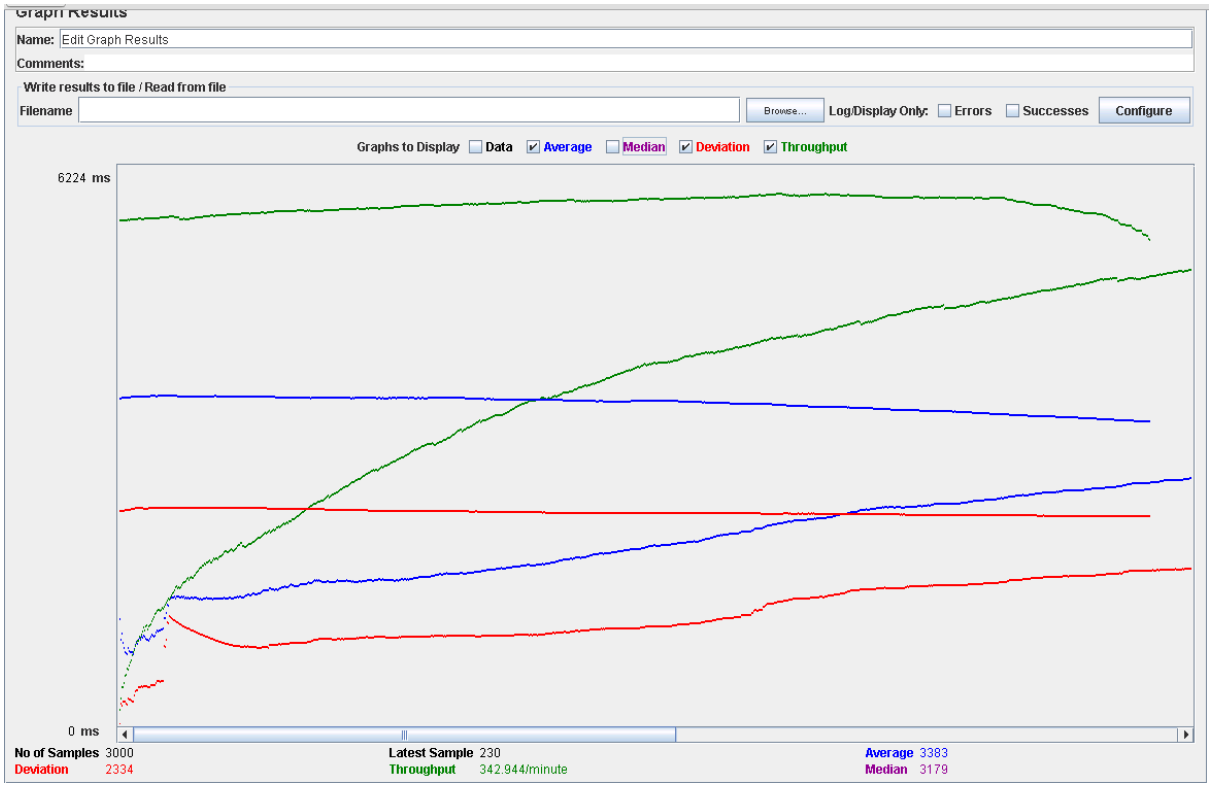
Comments:

Write results to file / Read from file

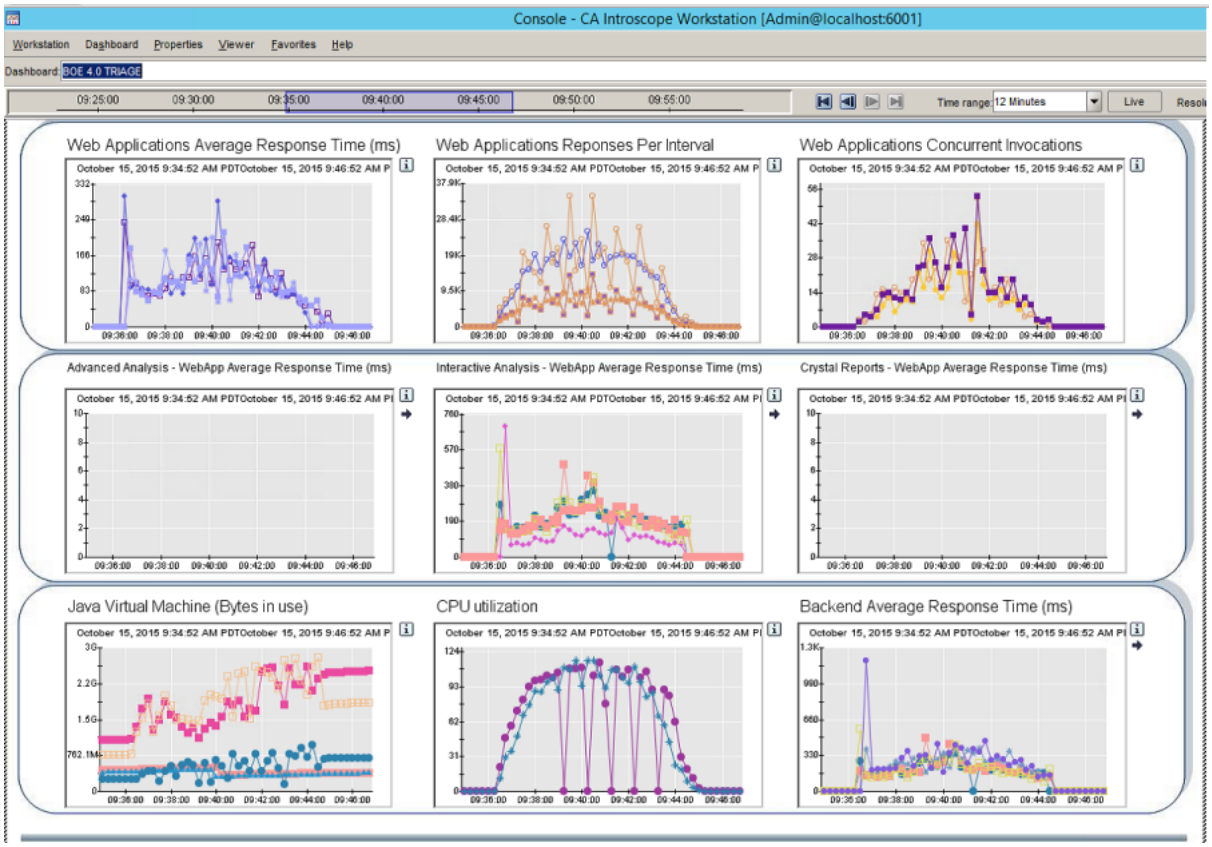
Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	600	2312	496	6724	1190.04	0.00%	1.3/sec	543.55	428992.0
Browse Folders	600	3995	1378	9568	1708.63	0.00%	1.3/sec	489.72	390469.2
View Report	600	4193	1373	11638	1805.95	0.00%	1.3/sec	1172.64	952053.0
Edit and Refresh	600	5802	2562	11890	1941.41	0.00%	1.2/sec	664.41	552422.1
Close and Logout	600	613	109	3144	419.05	0.00%	1.3/sec	38.80	30913.0
TOTAL	3000	3383	109	11890	2334.30	0.00%	5.7/sec	2628.85	470969.9

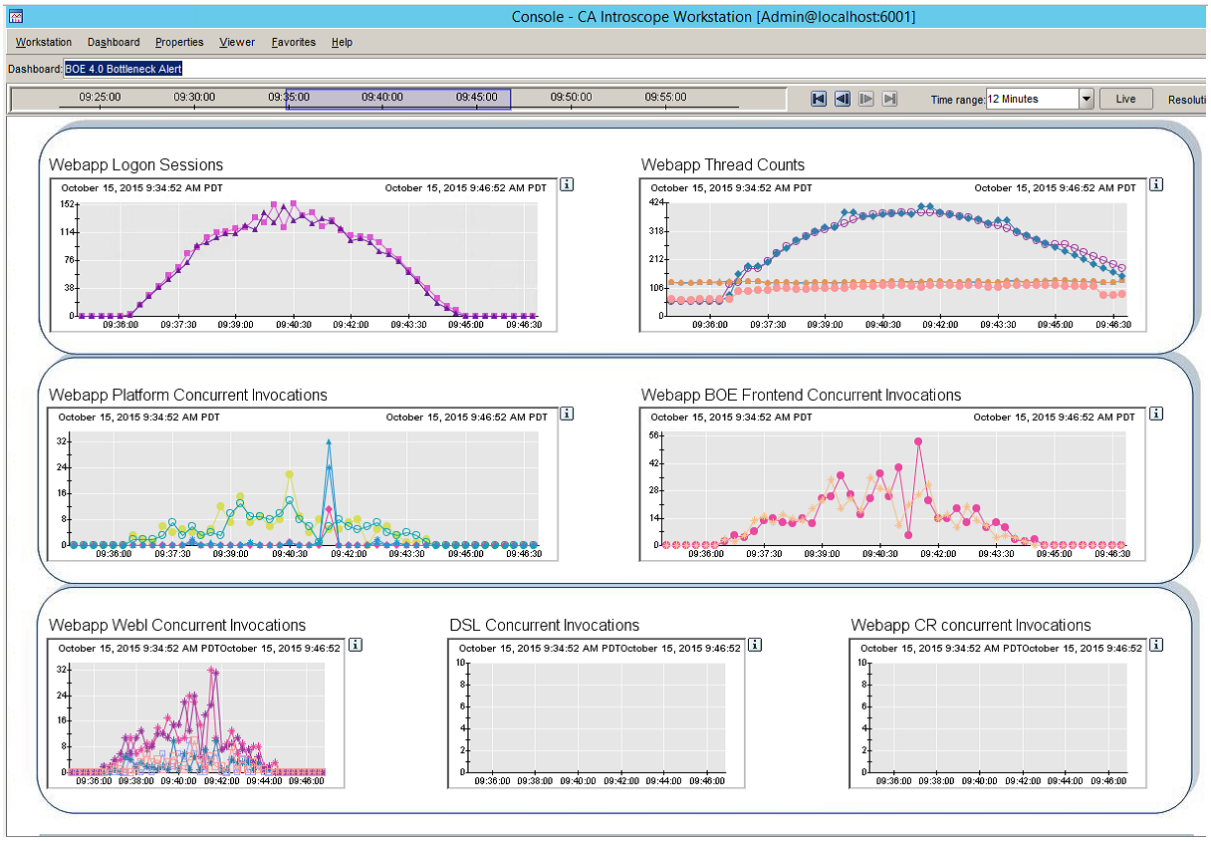
JMeter Edit Output Graph



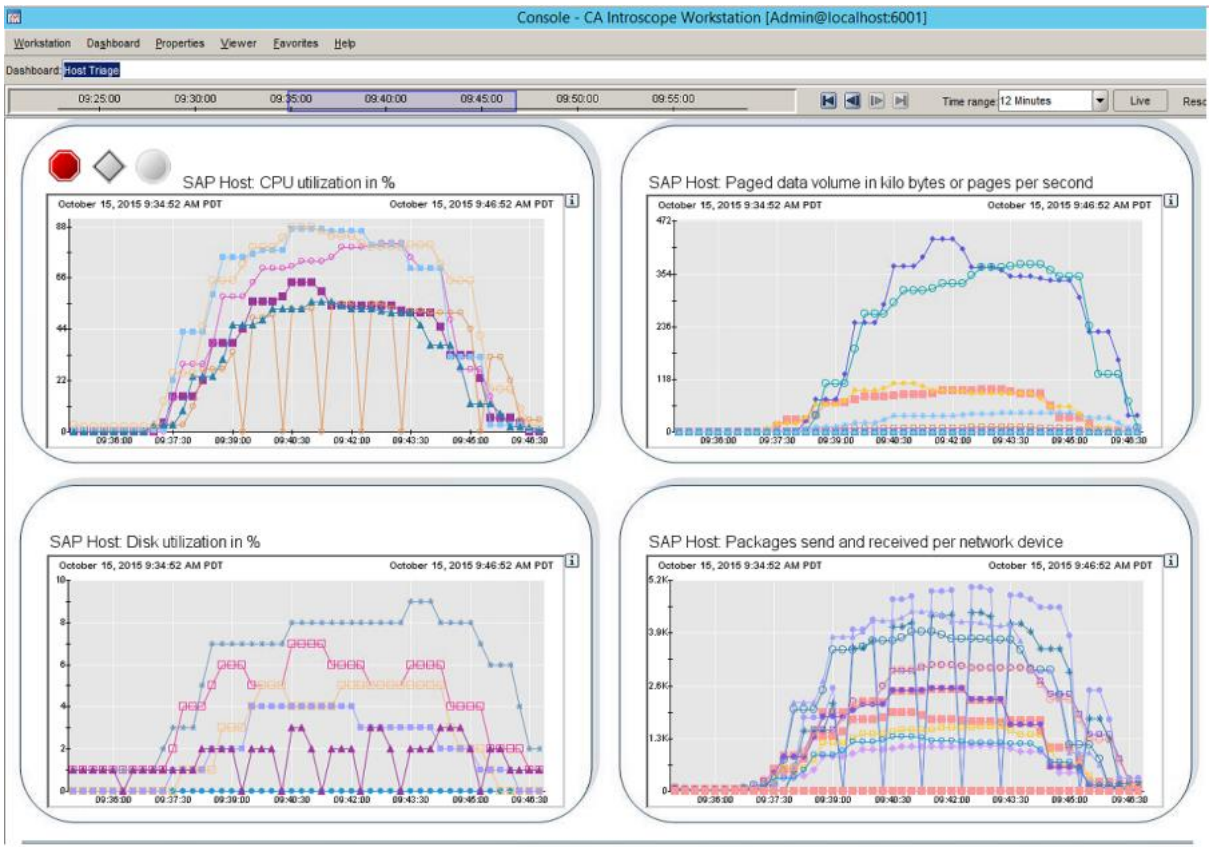
BI Triage - 300 ACU server config-0000



Bottleneck Alert - 300 ACU server config



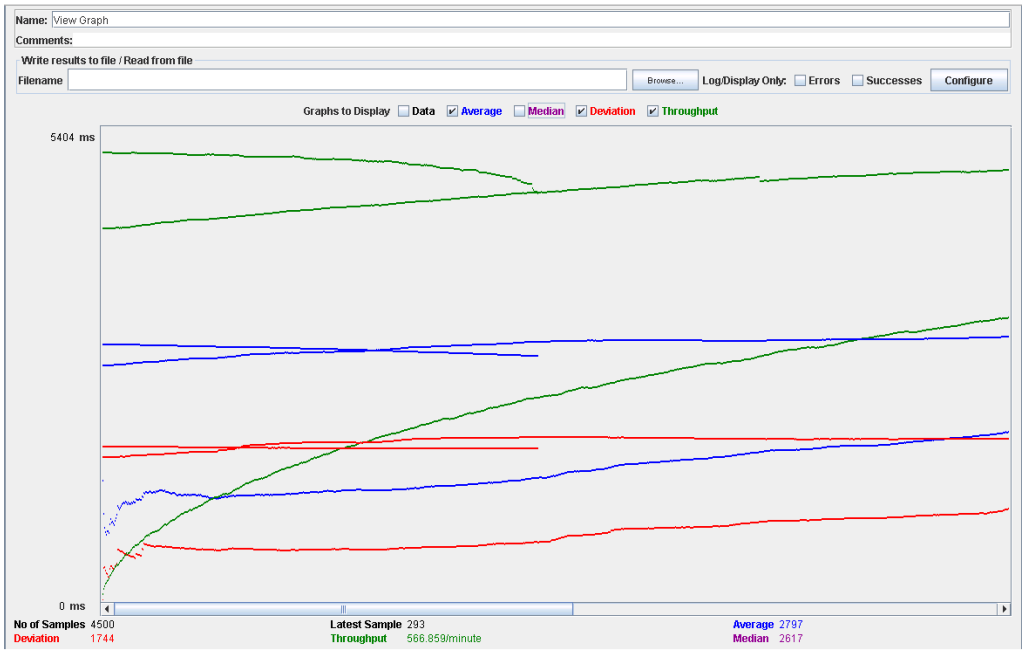
Host Triage - 300 ACU server config-0001



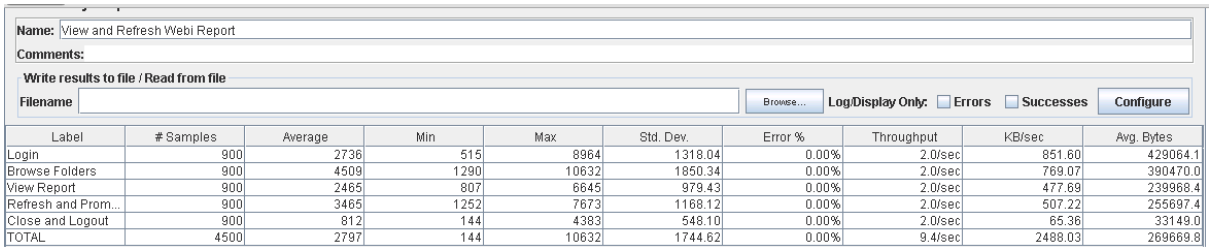
Tomcat GC Details - 300 ACU server config

Tomcat CPU Usage - 300 ACU server config

JMeter View Output Graph



JMeter View Output



Name: View and Refresh Webi Report

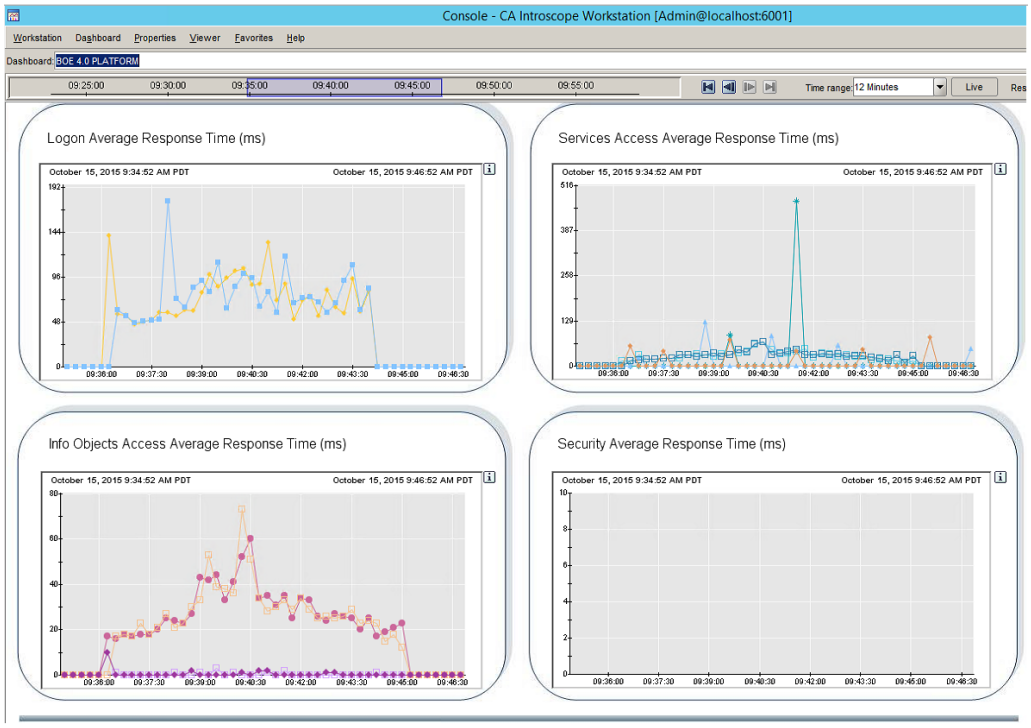
Comments:

Write results to file / Read from file

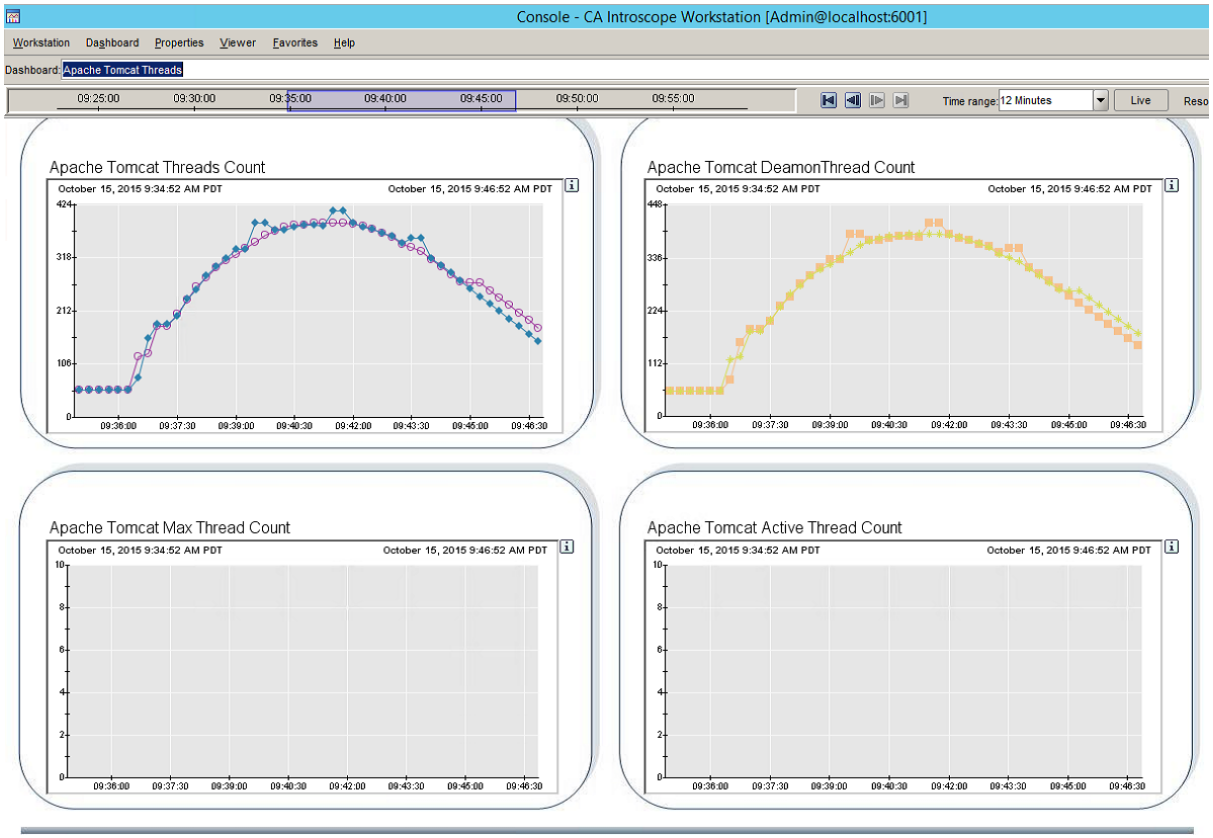
Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	900	2736	515	8964	1318.04	0.00%	2.0/sec	851.60	429064.1
Browse Folders	900	4509	1290	10632	1850.34	0.00%	2.0/sec	769.07	390470.0
View Report	900	2465	807	6645	979.43	0.00%	2.0/sec	477.69	239968.4
Refresh and Prom...	900	3465	1252	7673	1168.12	0.00%	2.0/sec	507.22	255697.4
Close and Logout	900	812	144	4383	548.10	0.00%	2.0/sec	65.36	33149.0
TOTAL	4500	2797	144	10632	1744.62	0.00%	9.4/sec	2488.03	269669.8

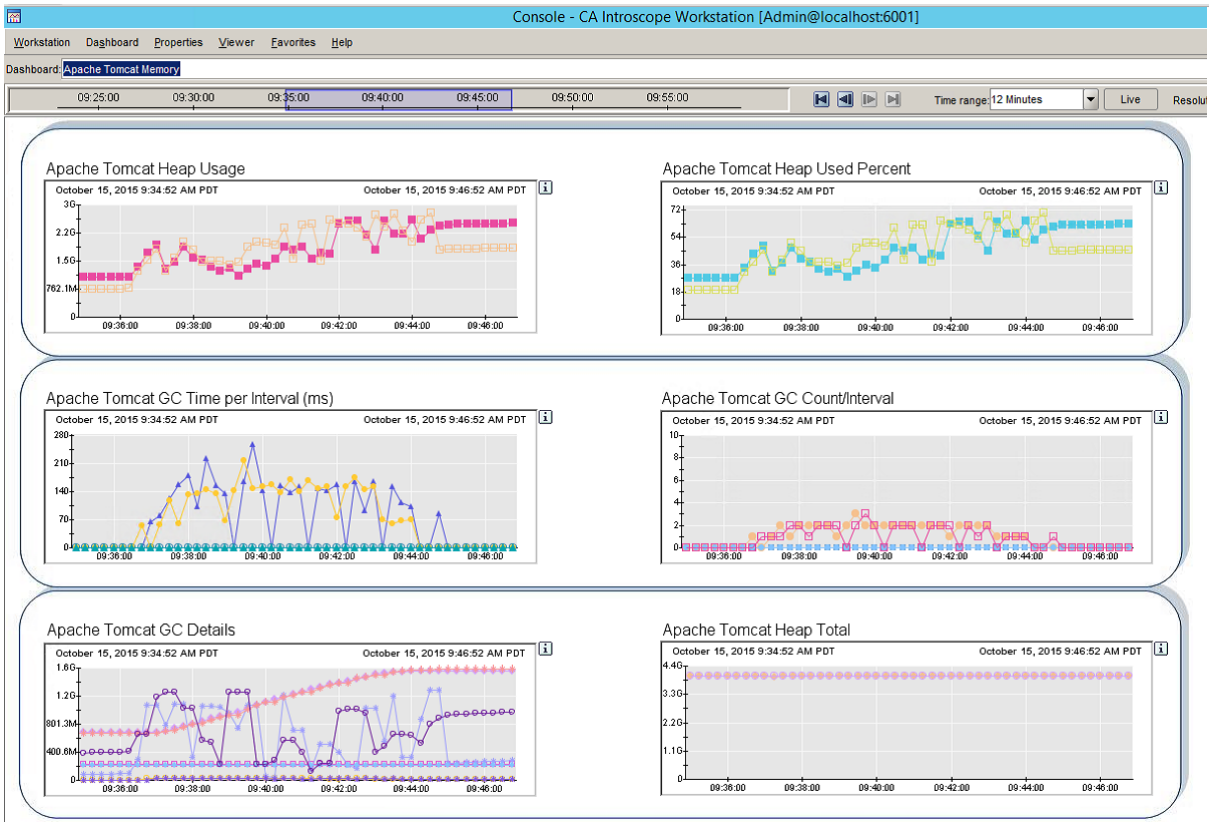
Platform Averages - 300 ACU server config



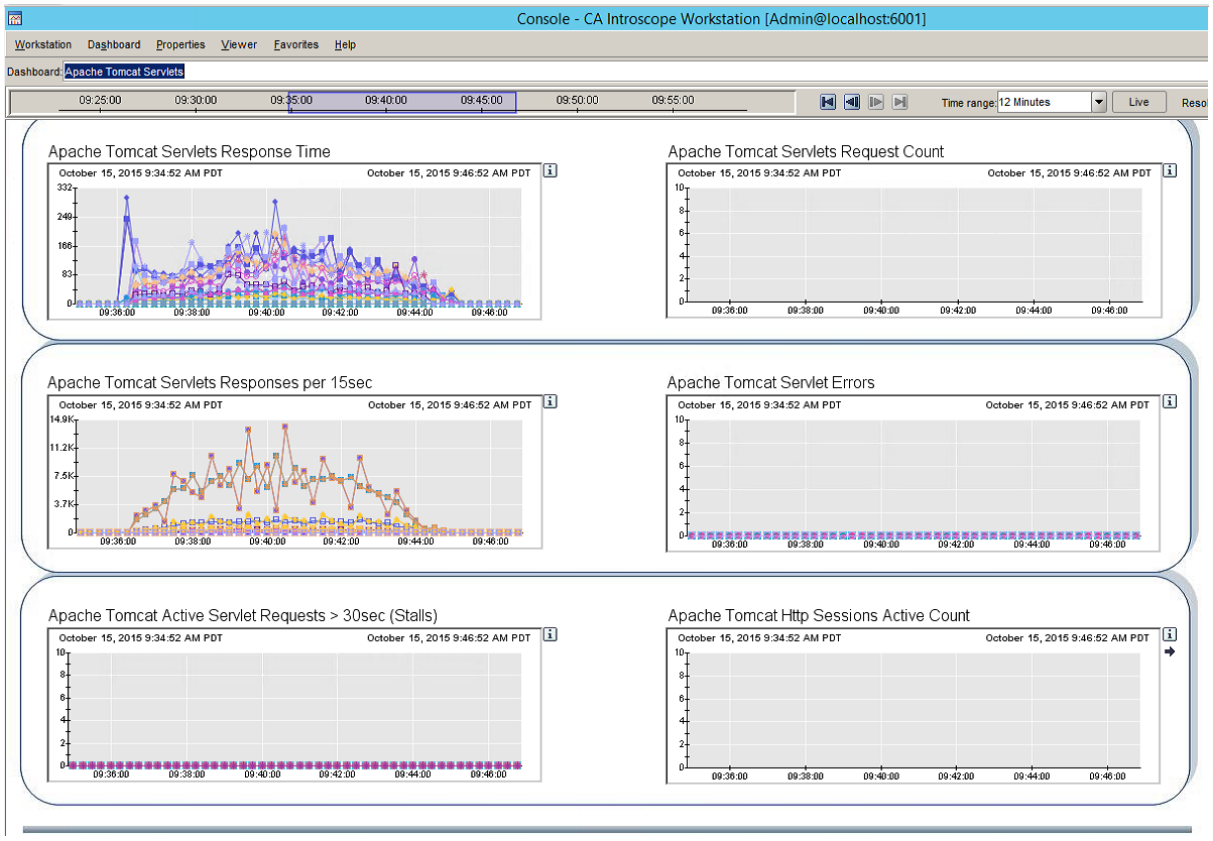
Tomcat Threads - 300 ACU server config



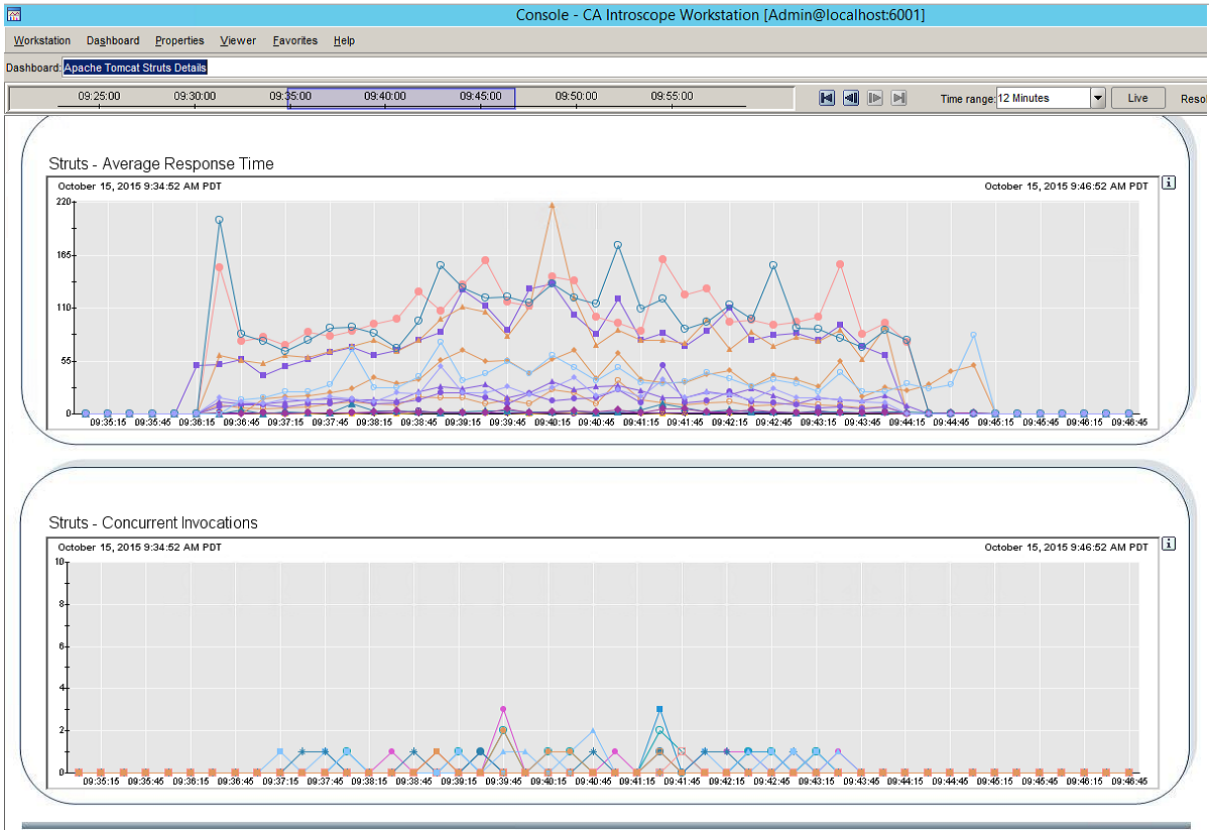
Tomcat Memory - 300 ACU server config



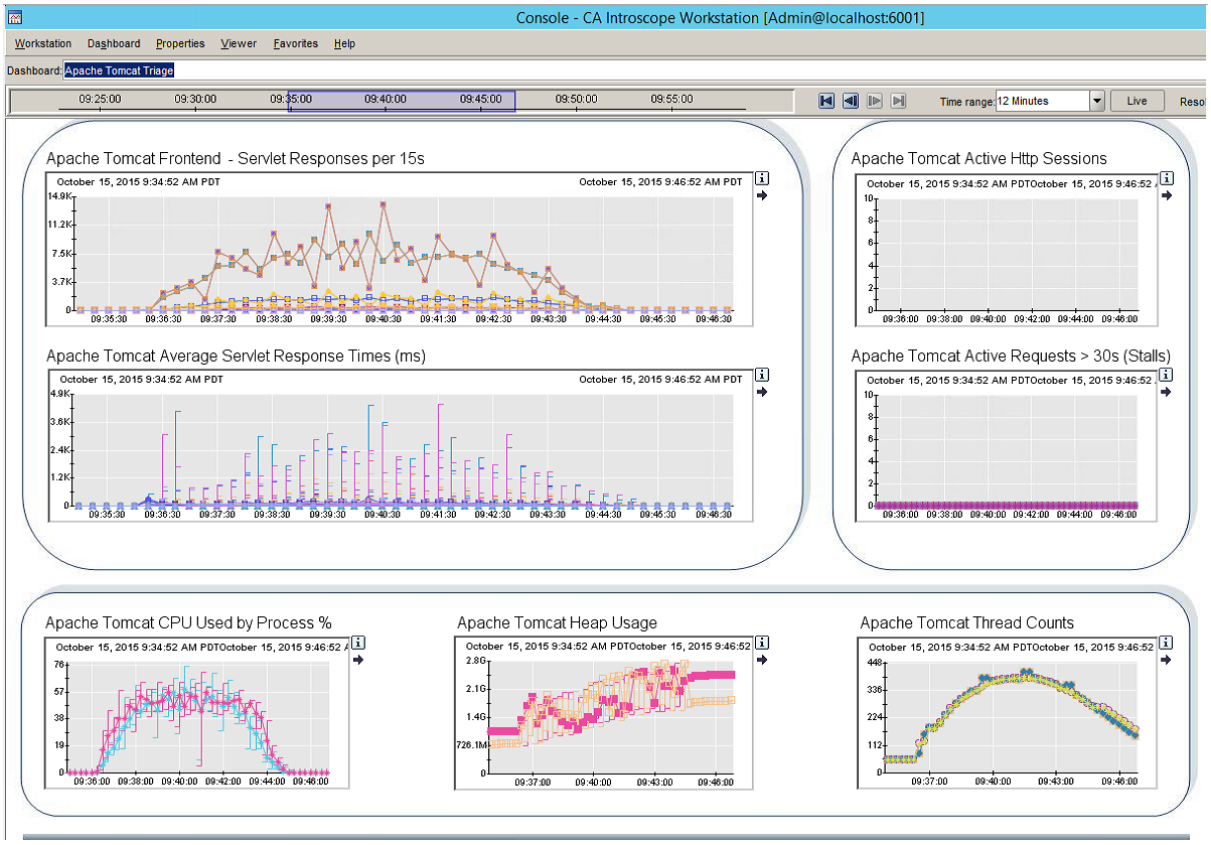
Tomcat Servlets - 300 ACU server config



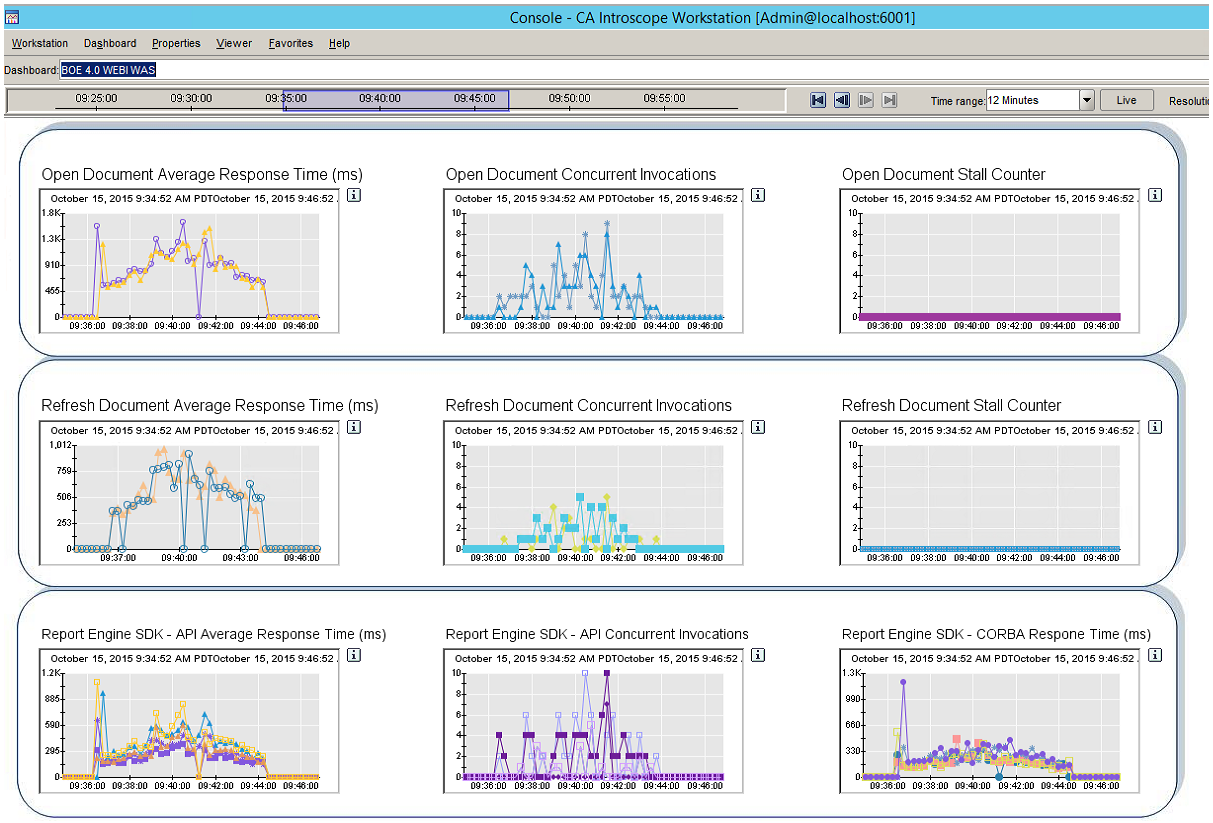
Tomcat Strut - 300 ACU server config



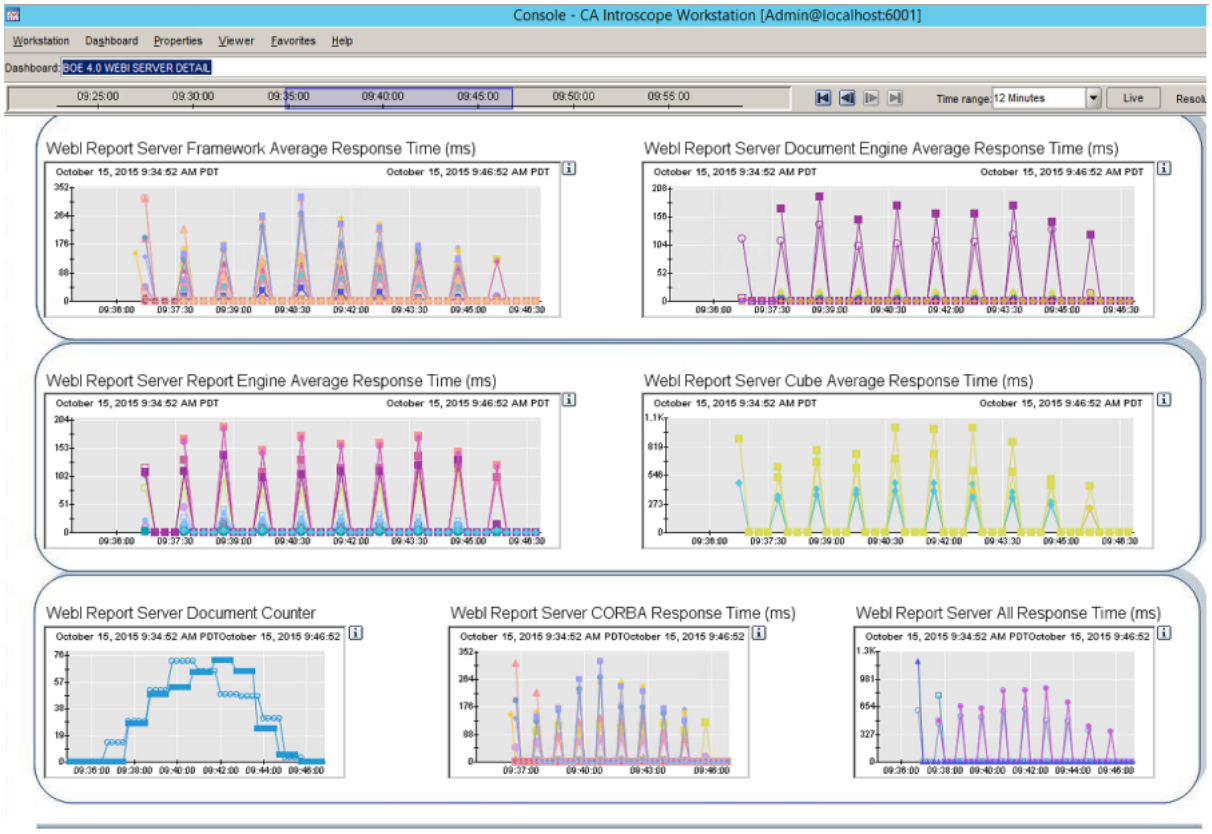
Tomcat Triage - 300 ACU server config



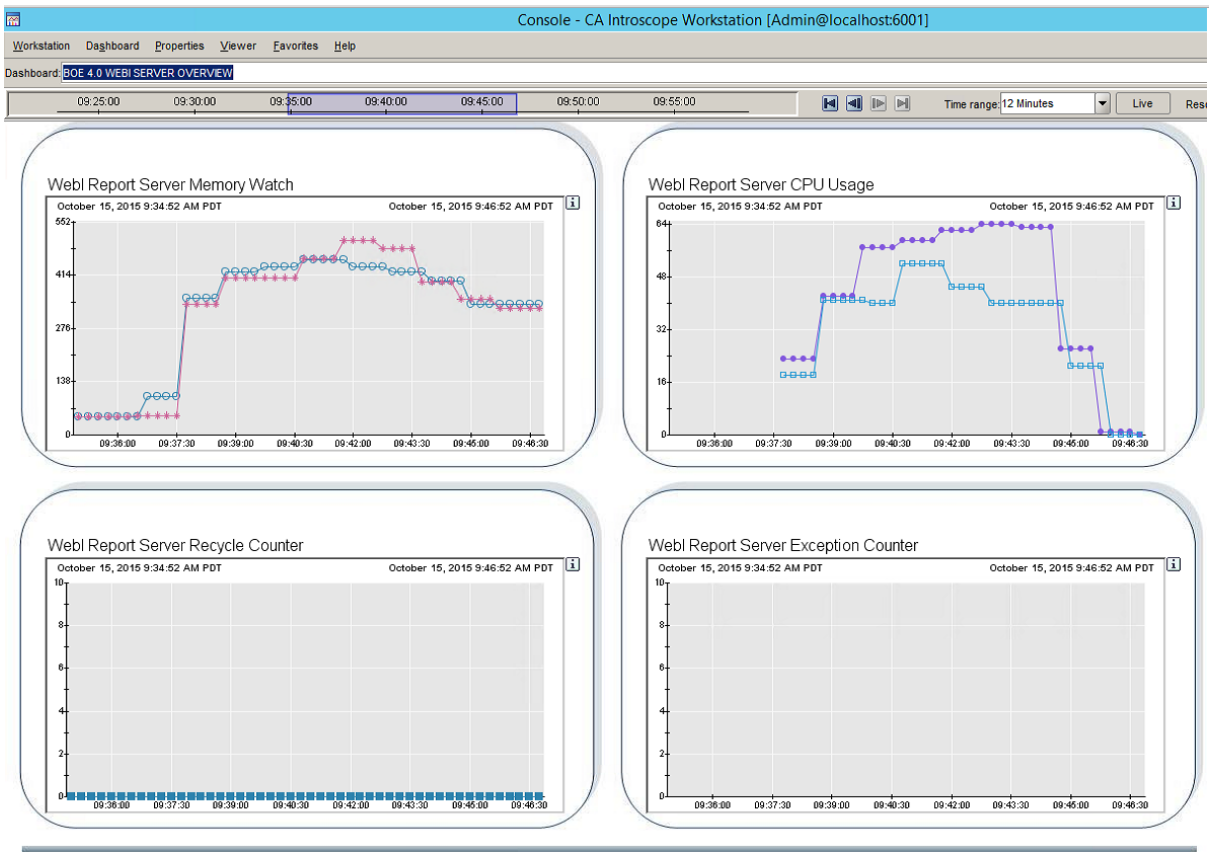
Webi WAS - 300 ACU server config



Webi Server Details - 300 ACU server config-0002



Webi Server Overview - 300 ACU server config



Performance Tuning Resources and Tips

This section will cover some of the resources and tips that can help you performance tune your future or existing BI 4.x deployments.

BI Platform Suite

BI Suite Performance Wiki – A comprehensive listing of Knowledge Base Articles, SCN Content and other resources related to improving the performance of your BI Suite.

Sizing and Deploying SAP BI 4 and SAP Lumira – SCN Document that covers sizing and deployment for the BI 4.x Suite and Lumira.

SAP BI 4 Sizing Guide – The official Sizing Guide for BI 4.x

SAP BI 4 Resource Usage Estimator – A Sizing Estimator that you can use to do a simple sizing exercise for your environment.

Best Practices for SAP BI 4.x Adaptive Processing Servers – An SCN Document that covers some best practices for splitting the Adaptive Processing Server so that it will perform optimally for your deployment.

Large-Scale BI 4 Platform Deployment White Paper - This white paper describes the outcome of a project enabled by SAP Co-Innovation Lab to validate the performance and scalability of a large-scale deployment of the SAP® BusinessObjects™ Business Intelligence (BI) platform, release 4, capable of sustaining ten thousand plus concurrent users. The SAP BusinessObjects BI platform and database technologies were configured to run over a standard enterprise-ready open source operating system from Red Hat (RHEL 6), using enterprise data center–ready Intel® Xeon® processor-based hardware from Supermicro (TwinBlades and SuperServers) and an application delivery control system from F5.

Improving BI 4.x Performance & Reliability using Java Parallel Garbage Collector Video – YouTube video covering how to use Parallel Garbage Collection with the SAP BI 4.x Java Processes to help improve performance and reliability.

SAP BI 4.x Pattern Books – Other SAP Pattern Books for BI 4.x

Technical Whitepaper: Best Practices for Virtualizing SAP BusinessObjects BI 4.x on VMware ESXi 5 - Whitepaper that details a co-innovation project that SAP, VMware and Supermicro performed. The whitepaper focuses on validating SAP's BI Platform on VMware ESXi 5 Virtualized hardware

Best Practices for running BI 4 in virtualized environments - guidances from COIL projects - A blog about best practices and resources for setting up BI 4 in a virtualized environment.

Evaluating Selected Java Best Practices For SAP BusinessObjects Business Intelligence 4 on vSphere - SCN Document that focuses on evaluating a subset of Java-specific best practices for BI 4 and VMware vSphere 5.

Web Intelligence

Tips for Optimizing the Performance of Web Intelligence Documents – SCN Doc that covers many tips and tricks you can use to improve the performance of your Webi documents. This covers everything from Process Workflow to Design tips.

Tips for Fine Tuning Performance of the Webi Applet Interface Wiki – Tips on how you can improve the performance/load time of the Applet Interface (aka Java Report Panel) for Web Intelligence.

Tuning Web Intelligence: Query Stripping and Server Settings Video – A video covering some key features related to fast performing Webi documents.

Performance Optimizing SAP BOBJ Reports Based on SAP BW using BICS Connectivity – A comprehensive guide covering performance optimizations and best practices for Web Intelligence reports using BICS based connectivity.

Performance Tips for Webi Reports using BICS – blog post by SCN member giving some pointers on how to optimize Webi and BW to work optimally together.

BI 4.x Web Intelligence Performance Test with JMeter – SCN Document covering some Performance Testing scenarios with JMeter and Webi.

Web Intelligence BI 4.x Tricks – A pretty complete listing of helpful resources that will help design better Webi docs.

Semantic Layer

Universe Optimization Techniques – Blog post by an SCN community member with a bunch of valuable tips.

BI 4.x Performance Optimizations for Multisource Universes (BW JCO) – A blog talking about MSU optimizations.

Web Server / Application Server Tier

Improving the User Experience in SAP BI 4.x with Apache and WDeploy – SCN Doc covering how to utilize Apache Web Server in conjunction with WDeploy to improve performance for static content on your Web Tier.