Leveraging the Newest Tools to Prevent and Minimize the Impact of Outages in WebSphere Commerce Sites

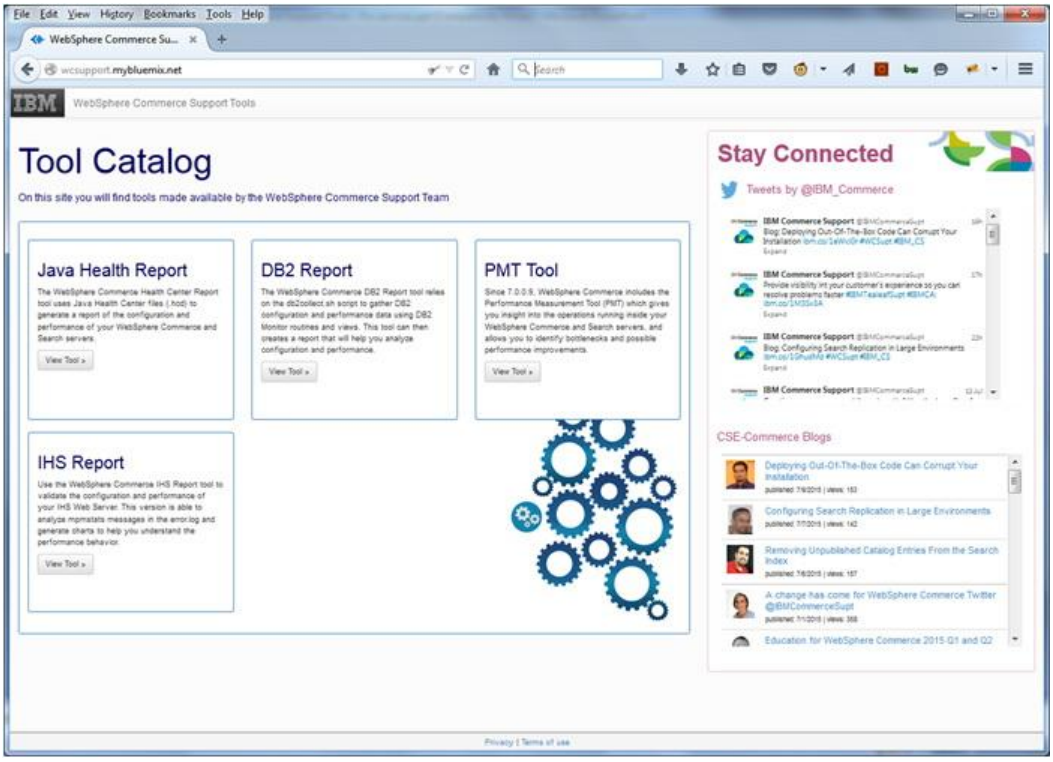
Lab Instructions

Authors:
Andres Voldman, WebSphere Commerce Support, voldman@ca.ibm.com
Charek Chen, Commerce Performance, charekc@ca.ibm.com
Steve McDuff, B2B Performance Architect, mcduffs@ca.ibm.com

1 Introduction

This lab demonstrates how to troubleshoot a production incident using on-line tools made available by the WebSphere Commerce team:

https://wcsupport.mybluemix.net



Instead of following step by step instructions, this lab presents a scenario and different reports generated during the incident.

The lab will highlight different findings in the reports. You are welcome to navigate the reports, and to come up with your own findings and questions.

1.1 Scenario

Shoppers report that the site is unresponsive and they are unable to checkout.

The following data is available for analysis:

a) IHS mpmstats data (automatically collected)
b) Java Health Center data (automatically collected)
c) Performance logger trace for the Performance Measurement tool (collected for 10 minutes during the issue)
d) DB2 data collected with db2collect.sh during the event for use with the WebSphere Commerce DB2 Report tool

1.2 Requirements

To complete this lab you need a desktop computer with internet access, a recent version of Firefox or Chrome, and Excel or equivalent software that can open .xls files.

The Excel viewer for Windows is available for download from the microsoft.com website: https://www.microsoft.com/en-ca/download/details.aspx?id=10.

1.3 Preparation

Download the lab files and uncompress into a local directory:

ftp://public.dhe.ibm.com/software/is/commerce/education/labperftools.zip

The contents of labperftools.zip are as follows:

Baseline – Samples collected during healthy operation	
db2collect.2015-09-19-09.25.46.zip	Output of db2collect.sh for input to the WebSphere Commerce DB2 Report tool
healthcenter180915_200709_15401098_63.hcd	Health Center file for input to the WebSphere Commerce Health Center Report tool
javacore.20150919.085150.15401098.0084.txt	Javacore file
performanceReport (directory)	Report generated by the Performance Measurement Tool (PMT)
Error – Samples collected during the problem time	
db2collect.2015-09-19-13.28.46.zip	Output of db2collect.sh for input to the WebSphere Commerce DB2 Report tool
healthcenter180915_200709_15401098_69.hcd	Health Center file for input to the WebSphere Commerce Health Center Report tool
javacore.20150919.133054.15401098.0168.txt	Javacore file
performanceReport (directory)	Report generated by the Performance Measurement Tool (PMT)
error_log	IHS error_log with mpmstats output for the WebSphere Commerce IHS Report Tool * This file covers both periods, before (baseline) and during the problem

2 Analyzing the web server layer

As the entry point to the site, the web servers offer excellent statistics to understand the load, performance, and general health of the site.

Next we'll discuss how to use mpmstats statistics with the WebSphere Commerce IHS Report tool to get insight into the timeline and severity of the production incident we are investigating.

Mpmstats statistics

Mpmstats is a non-intrusive monitoring module that periodically logs the state of the web server threads. It allows you to quickly get an idea of the load and health of the site.

In the site being investigated, the default mpmstats configuration was updated to get more frequent reports and logging of slow requests.

The configuration in httpd.conf is as follows:

```
<IfModule mod_mpmstats.c>
ReportInterval 60
TrackModules On
SlowThreshold 5
</IfModule>
```

ReportInterval: Every ReportInterval number of seconds, mpmstats reports thread statistics if the server is non-idle. This was updated from 600 (10 minutes) to 60 (every minute).

Sample log entry:

```
[notice] mpmstats: rdy 43 bsy 7 rd 1 wr 6 ka 0 log 0 dns 0 cls 0
```

TrackModules: Enables monitoring of the number of threads active in the WebSphere plug-in module.

Sample log entry:

```
[notice] mpmstats: bsy: 6 in mod_was_ap22_http.c
```

SlowThreshold (IHS Fix Pack 7.0.0.23+): Logs the number of requests that have been active for longer than the time specified as threshold. Value is in seconds.

Sample log entry:

```
[notice] mpmstats (long-running only): bsy: 363 in mod_was_ap22_http.c
```

WebSphere Commerce IHS Report

The WebSphere Commerce IHS Report (https://wcsupport.mybluemix.net/wcihs/) can be used to facilitate the analysis of mpmstats data.

2.1 Generating a new report

Use the WebSphere Commerce IHS Report tool to graph mpmstats statistics generated prior (baseline) and during the problem time:

1. Access the WebSphere Commerce IHS Report tool:

   http://wcsupport.mybluemix.net/wcihs/

2. Use this file to generate a report:

   error\error_log

2.2 Reviewing the report

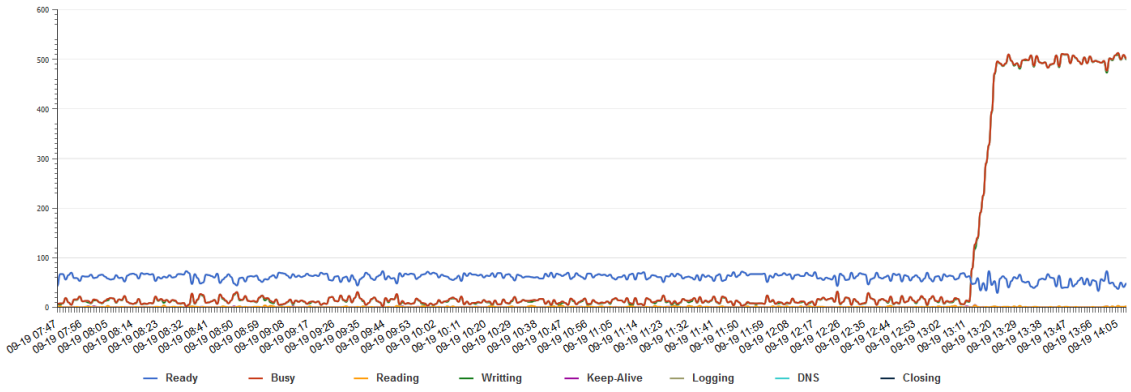
Open the report and navigate the following menu items:

1. mpmstats > Connections Chart
2. mpmstats > WebSphere Chart
3. mpmstats > WebSphere Chart (long running)

2.2.1 mpmstats - Connections by State

The Connections by State graph shows the state of the connections by time. The frequency of the measurements is determined by *ReportInterval* in httpd.conf, currently configured to 60 seconds.

In this graph you can clearly see that a sudden event was triggered at approximately 13:10, and the number of connections jumped.



The number of connections can increase due to a sudden increase in load (e.g. time boxed promotion), or due to a responsiveness problem with the site.

Note that although all states are charted, due to overlaps (e.g. Busy and Writing), some states might not be clear in the graph.

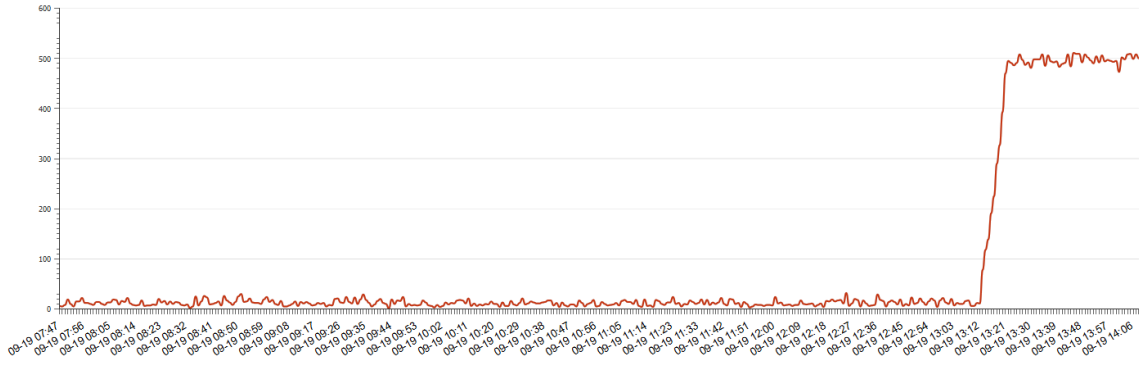
The complete list of states is as follows:

State	Description
Ready (rdy)	Threads started and ready for new connections
Busy (bsy)	Threads already associated to a connection. This number includes threads in keep-alive state
Reading (rd)	Reading a request from the client
Writing (wr)	Either processing the request (e.g. waiting for the Application Server) or writing back to the client. This value typically matches the number of requests waiting for the Commerce servers
Keep-Alive (ka)	Threads waiting for new work (KeepAlive) on the same connection
Logging (log)	Writing to the logs
DNS Lookup (dns)	Doing a DNS lookup
Closing (cls)	Threads that are waiting for the client to acknowledge that the entire response has been received so that the connection can be closed

2.2.2 mpmstats – WebSphere Chart

The WebSphere Chart uses *TrackModules* output to chart the number of threads in the WebSphere plugin-in.

This second chart confirms that the increased threads are doing WebSphere work.



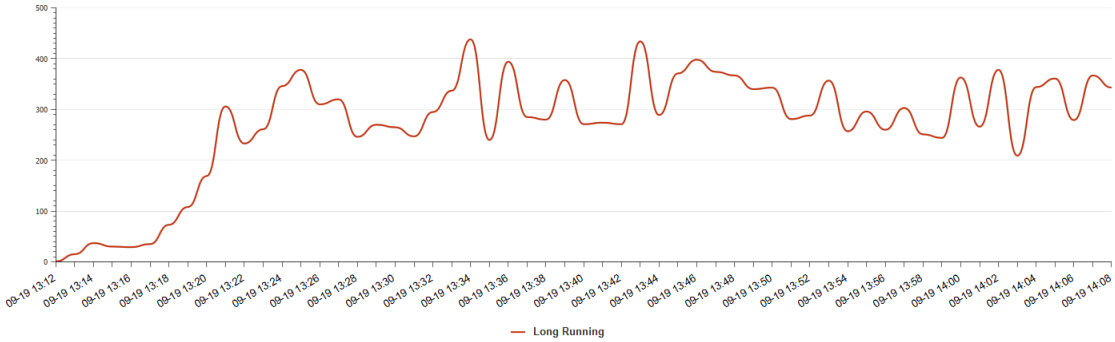
This could still be the result of increased load or slower responses.

2.2.3 mpmstats - Long Running WebSphere Plug-in Connections

When *SlowThreshold* is enabled, the WebSphere Commerce IHS Report also graphs slow responses.

With *SlowThreshold* configured to 5 seconds in httpd.conf, the chart shows the

number of requests that have been active for 5 seconds or longer at every point of measurement (every 60 seconds as configured with *ReportInterval*).



SlowThreshold only logs when slow responses are captured. The timeline of this chart could be a bit misleading. Notice that this chart starts at 13:12 instead of 7:50 like the previous ones. This is because there were no slow responses before 13:12.

This confirms a large number of long running threads in the plug-in, which points to a responsiveness issue with the WebSphere Commerce servers.

2.3 Learning Checkpoint

After reviewing the IHS Report we find the following:

1. The report confirms the event started at 13:10
2. Most connections are waiting on the plugin for response from the Commerce servers
3. At some point all the connections are hung or slow responding

The next step is to see what was happening inside the Commerce servers to understand why they might not be responding.

3 Analyzing the WebSphere Commerce layer

In this section we'll analyze the WebSphere Commerce servers using the WebSphere Commerce Health Center Report tool.

WebSphere Commerce Health Center Report tool

IBM Health Center for Java is a low overhead agent included with the WebSphere Application Server Java SDK that collects Java configuration and performance data. The data collected includes CPU, native memory, method profiling, garbage collection, locks, threads, and others.

Configured in headless mode, the agent continuously collects performance data into Health Center files (.hcd). This data is valuable not only for performance tuning, but also serves as a performance "flight recorder" for root cause analysis of production incidents.

The WebSphere Commerce Health Center Report tool uses the Health Center file (.hcd) to generate a web-based report of the performance data with WebSphere Commerce specific insight.

Configuring the Health Center agent

Although the Health Center agent is included with WebSphere, if you are not running a recent SDK Fix Pack (7.0.0.37+), you might need to update it.

After the Health Center agent is enabled in the generic JVM properties, the server will start to continuously log performance data into .hcd files.

The following link contains detailed instructions to configure a server:

Health Center agent - Installation
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W02f8b2c4804e_40e0_bbd1_32921368b666/page/HC%20-%20Installation

3.1 Generating a new report

Use the WebSphere Commerce Health Center Report tool to analyze the data collected by the Java Health Center agent during the time of the problem:

1. Access the WebSphere Commerce Health Center Report tool:

   http://wcsupport.mybluemix.net/hctd/

2. Use this file to generate a report:

   error\healthcenter180915_200709_15401098_69.hcd

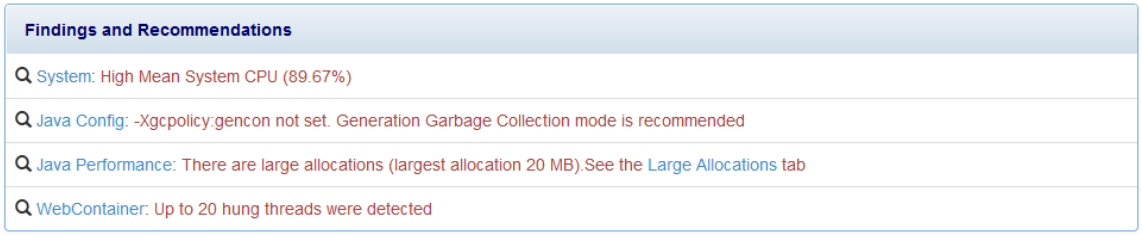
For comparison purposes, you can also generate a report with data collected during healthy operation (baseline) using this file:
baseline\healthcenter180915_200709_15401098_63.hcd

3.2 Analyzing the Health Center report

Open the report generated with data collected during the problem and navigate these menu items:

1. Overview
2. System > Environment
3. System > CPU Usage

The overview panel includes important findings and recommendations for the report:



Some of the findings might not be directly related to the outage, but it is still worthwhile addressing them. The most interesting finding is that up to 20 WebContainer threads were found to be hung.

3.2.1 Java menu

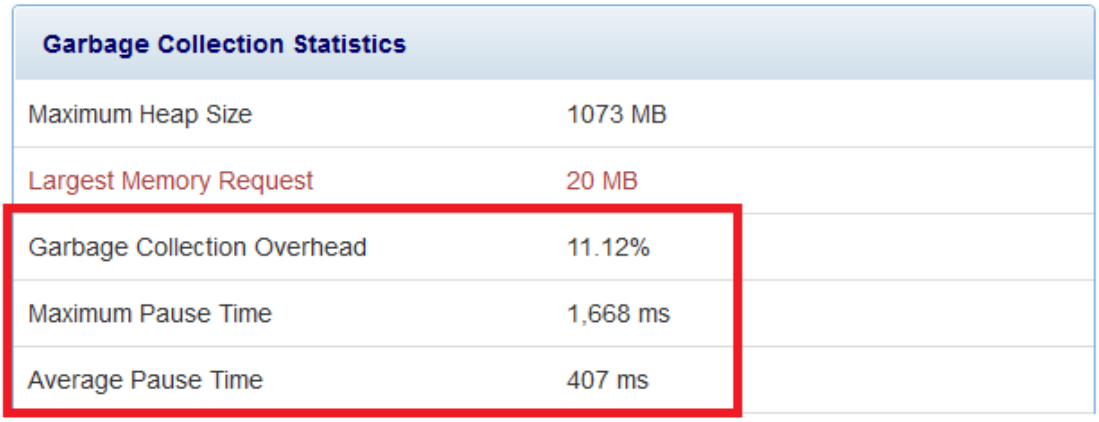
When reviewing a report, it is a good practice to navigate the different panels to identify anomalies.

The Java menu item provides access to Java configuration and garbage collection analysis.

Open the report and navigate the following menu items:

1. Java > Config
2. Java > Statistics
3. Java > Heap Usage
4. Java > Large Allocations

The Java overhead (percentage of time spent doing garbage collection work versus application work) and the pause times, under Statistics, are good indicators as to whether the problem is related to Java garbage collection. In this case the values are not alarming.



Garbage Collection Statistics	
Maximum Heap Size	1073 MB
Largest Memory Request	20 MB
Garbage Collection Overhead	11.12%
Maximum Pause Time	1,668 ms
Average Pause Time	407 ms

When a server is having memory problems, the garbage collection overhead is typically +40%, and pauses are of several seconds.

The "Rate Of Garbage Collection" value can help you understand the level of load the server was receiving, as more load also means more objects are created that need to be garbage collected. This value is most useful when compared against a baseline.

Although Java garbage collection does not seem to be the cause of the outage, you should take note of the report findings, to investigate them and address them at a later time:

1. -Xgcpolicy:gencon not set. Generation Garbage Collection mode is recommended
2. There are large allocations (largest allocation 20 MB)

When using the Java section of the report, keep in mind that to keep the overhead to a minimum, the Java Health Center agent limits the information it collects and this could affect the accuracy of the reports. If you suspect a Heap memory problem, use a tool such as PMAT and garbage collection logging (verbose gc) for deeper analysis.

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)
https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=22d56091-3a7b-4497-b36e-634b51838e11

3.2.2 Thread analysis

Using Health Center for thread analysis has advantages over the use of javacores. The Health Center agent automates the collection of thread stacks. Samples are collected every 30 seconds and saved into the binary file.

Using the WebSphere Commerce Health Center Report tool you can extract thread data into Javacore-like files (from Threads > Thread Dump menu item). The tool also presents the thread data in charts and table format. The summarized views greatly reduce the troubleshooting time.

The tool works by discovering activities in the stack and creating a list. The most recent activity is called the "Base Activity". It is graphed in the "WebContainer Base Activity chart". This chart is very helpful to identify bottlenecks. Commonly seen

base activities include: waiting on the database, waiting for an STMP server, waiting for the Search server, and others.

The complete list of activities for a thread is called the activity chain. The activity chain is read the same way you read Java stacks, in that the most recent activity is to the left, and was initiated by the activity to its right. The activity chain helps identify the code that executed prior to the base activity. You could find, for example, that all the threads waiting for the database used the same EJB.

The following example summarizes a stack that is over 200 lines long:

```
Network(DB2):DBQuery:EJB(OrderItemAdjustmentBean):RESTTag:JSP(OrderShip
pingBillingConfirmationPage):Get:Runtime:Web
```

In this example the OrderShippingBillingConfirmationPage JSP, used the REST Tag to access the database with the OrderItemAdjustment EJB. The thread is currently waiting on the database to respond (Network(DB2)).

The tool also uses the concept of "Hung Threads", which are threads whose stack hasn't changed from the previous sample (30 seconds ago). Highlighting hung threads helps to quickly identify which threads are not moving.

Next we'll review the charts the tool makes available with this data.

Open the report and navigate these menu items:

1. Threads > Thread Dumps
2. Threads > WebContainer Base Activity
3. Threads > WebContainer Detailed Activity

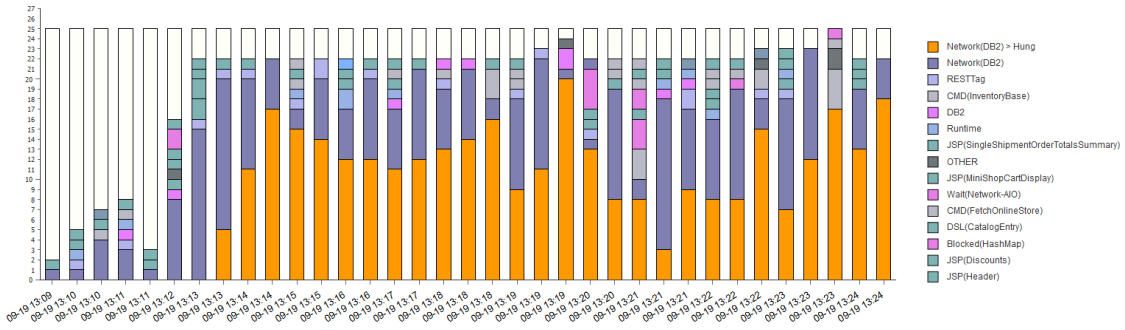
3.2.2.1 Threads > Thread Dumps

From this menu you can download thread data into Javacore-like files. It's useful for additional analysis or analysis of non-WebContainer threads.

3.2.2.2 Threads > WebContainer Base Activity

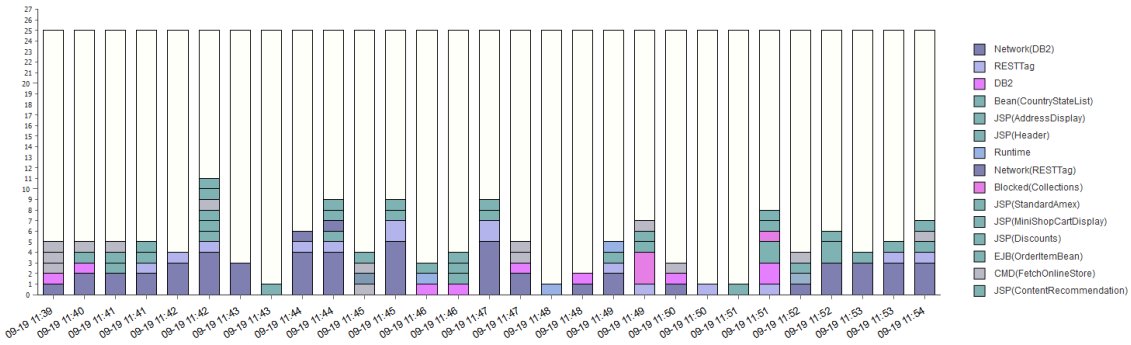
From the initial findings in the Overview panel we know that up to 20 threads were detected as being hung, so the WebContainer Base Activity chart should show what the threads are doing.

In this case, we find that most WebContainer threads are in use, and they are waiting for DB2 to respond.



To facilitate analysis, the tool makes a distinction of hung threads. You can see that the blue and orange lines (top 2), are both for threads waiting on DB2.

For reference, the baseline chart looks as follows. In healthy servers it is common to find that only a few threads executing.

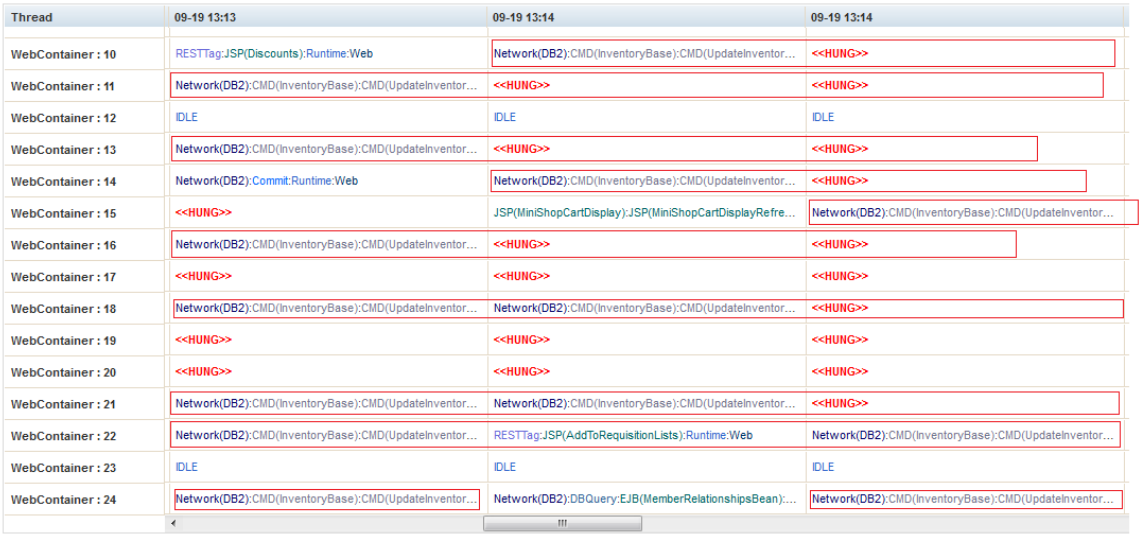


3.2.2.3 Threads > WebContainer Detailed Activity

The WebContainer Base Activity chart shows the bottleneck is the database. Next we'll review the WebContainer Detailed Activity chart which shows the complete activity chains in table format. This table can be used to identify if all threads waiting on the database are using the same code (e.g. same EJB), or if the delay appears to be with all the queries that are executed.

By clicking on an activity chain, the tool reveals a popup with the original Java stack from which the chain was created. This can be helpful if more details about the stack are needed.

The review of the WebContainer Base Activity table shows that many of the DB2 queries hanging are for the INVENTORY table from the OrderProcess command:



Thread	09-19 13:13	09-19 13:14	09-19 13:14
WebContainer : 10	RESTTag:JSP(Discounts):Runtime:Web	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>
WebContainer : 11	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>	<<HUNG>>
WebContainer : 12	IDLE	IDLE	IDLE
WebContainer : 13	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>	<<HUNG>>
WebContainer : 14	Network(DB2):Commit:Runtime:Web	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>
WebContainer : 15	<<HUNG>>	JSP(MiniShopCartDisplay):JSP(MiniShopCartDisplayRefre...	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...
WebContainer : 16	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>	<<HUNG>>
WebContainer : 17	<<HUNG>>	<<HUNG>>	<<HUNG>>
WebContainer : 18	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>
WebContainer : 19	<<HUNG>>	<<HUNG>>	<<HUNG>>
WebContainer : 20	<<HUNG>>	<<HUNG>>	<<HUNG>>
WebContainer : 21	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	<<HUNG>>
WebContainer : 22	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	RESTTag:JSP(AddToRequisitionLists):Runtime:Web	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...
WebContainer : 23	IDLE	IDLE	IDLE
WebContainer : 24	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...	Network(DB2):DBQuery:EJB(MemberRelationshipsBean):...	Network(DB2):CMD(InventoryBase):CMD(UpdateInventor...

The complete activity chain from the image above is as follows:

Network(DB2):CMD(InventoryBase):**CMD(UpdateInventory):**CMD(DoInventoryAction):CMD(ProcessOrder):CMD(OrderProcessPostApproval):CMD(BusinessFlowUrlEvent):**CMD(OrderProcess)**

This matches the report that many shoppers are unable to check out.

3.3 Learning Checkpoint

After reviewing the WebSphere Commerce layer we learned the following:

1. Java Health Center confirms a bottleneck in the Commerce servers
2. The report shows threads waiting on the database
3. Most threads are executing an inventory query from the OrderProcess command

The next step is to analyze DB2 data to understand why these queries are unresponsive.

4 Reviewing the database layer

Given that the WebSphere Commerce analysis points to threads waiting for DB2 to respond, in this section we'll analyze database activity.

Using the WebSphere Commerce DB2 Report tool

The WebSphere Commerce DB2 Report tool relies on the db2collect.sh script to gather DB2 configuration and performance data using DB2 Monitor routines and views. The collection is lightweight and provides a high level overview of database performance.

db2collect.sh script
https://wcsupport.mybluemix.net/static/1/db2report/scripts/db2collect.sh

4.1 Generating a new report

Use the WebSphere Commerce DB2 Report tool to generate a report with data collected during the problem time:

1. Access the WebSphere Commerce DB2 Report tool:

   http://wcsupport.mybluemix.net/wcdb2/

2. Use this file to generate a report:

   error\ db2collect.2015-09-19-13.28.46.zip

For comparison purposes, you can also generate a report with data collected during healthy operation (baseline) using this file: baseline\ db2collect.2015-09-19-09.25.46.zip

4.2 Analyzing the report

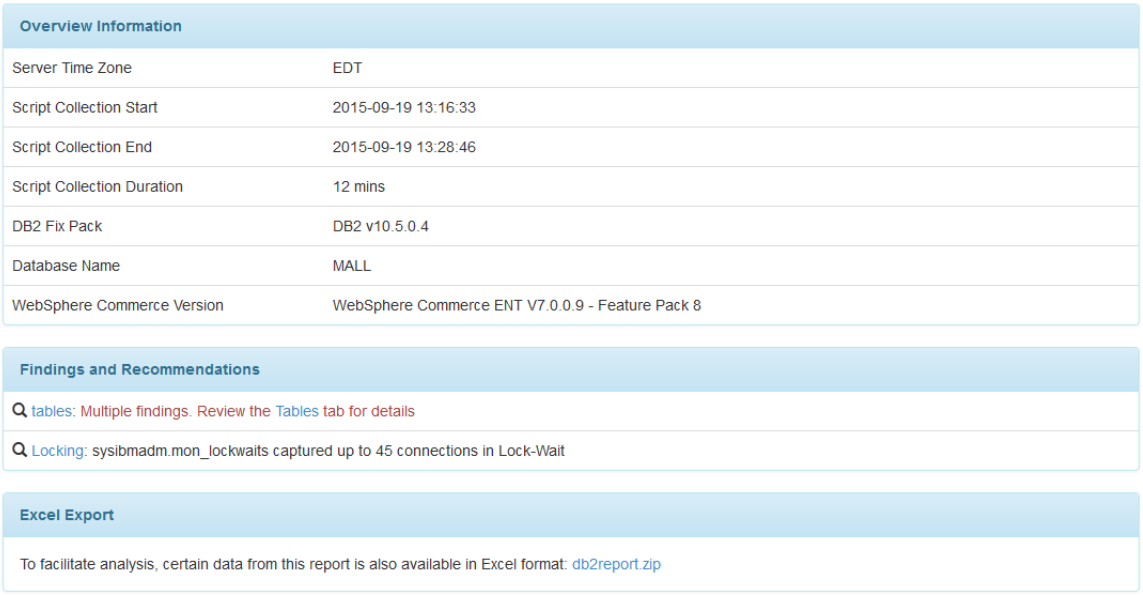
The following analysis is based on the report collected during the problem time. The baseline is provided for comparison purposes.

Open the report generated with data collected during the problem and navigate these menu items:

1. Overview
2. System
3. Config > Database
4. Config > Instance
5. Config > Profile Variables

4.2.1 Overview

Same as with the previous tools, the Overview tab presents high level data, findings and recommendations.



Overview Information

Server Time Zone	EDT
Script Collection Start	2015-09-19 13:16:33
Script Collection End	2015-09-19 13:28:46
Script Collection Duration	12 mins
DB2 Fix Pack	DB2 v10.5.0.4
Database Name	MALL
WebSphere Commerce Version	WebSphere Commerce ENT V7.0.0.9 - Feature Pack 8

Findings and Recommendations

tables: Multiple findings. Review the Tables tab for details

Locking: sysibmadm.mon_lockwaits captured up to 45 connections in Lock-Wait

Excel Export

To facilitate analysis, certain data from this report is also available in Excel format: db2report.zip

An important finding is that up to 45 connections we found to be in lock wait.

4.2.2 Connections

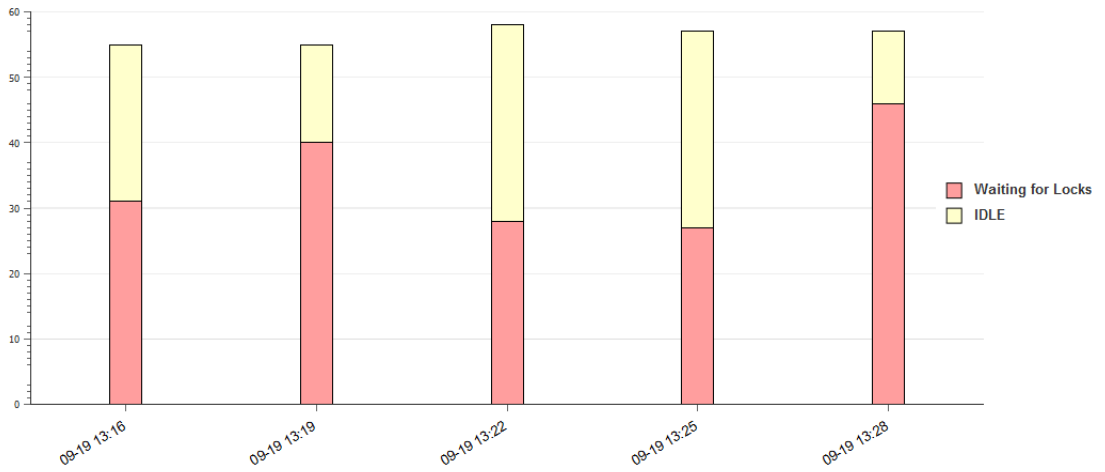
Continue analyzing the report by navigating the following menu items:

1. Connections > Agent State
2. Connections > Timeline
3. Connections > Active SQLs

4.2.2.1 Connections > Agent State

The Agent State chart shows the number of connections and their state at each snapshot time. States include IDLE, waiting for locks, executing, and others. In a way, you can think of this as the equivalent to mpmstats in the web server tier.

On a healthy database, you typically find only a few non-idle connections. In this case, we find a very large number of connections waiting for locks, and that's a problem.



4.2.2.2 Connections > Active Statements

The Active Statements menu displays the statements that were actively executing at the different snapshot times:

Summary of Active Statements

2015-09-19 13:16:36 (33) | 2015-09-19 13:19:38 (41) | 2015-09-19 13:22:40 (29) | 2015-09-19 13:25:42 (27) | 2015-09-19 13:28:44 (47)

STMT_TEXT	AGENT_STATE	NUM_AGENTS	ELAPSED_TIME_SEC
SELECT T1.QUANTITY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATENTRY_ID, T1.QUANTITYMEASURE, T1.STORE_ID, T1.OPTCOUNTER FROM INVENTORY T1 WHERE (T1.CATENTRY_ID = ? and T1.FFMCENTER_ID = ? and T1.STORE_ID = ?) FOR UPDATE WITH RS	LOCK:ACQUIRE	46	1,116
SELECT CATENTRY_ID, FFMCENTER_ID FROM INVENTORY WHERE STORE_ID = ? FOR UPDATE WITH RS	REQUEST:WAIT	1	969

As expected, there are many connections (46) waiting for locks (LOCK:ACQUIRE) while executing an inventory query. This matches the finding in the WebSphere Commerce layer that showed many threads waiting for the database while executing inventory logic from the OrderProcess command.

Interestedly, there is also a select for inventory data that is fetching all data for a particular store. This can point to a batch job running.

4.2.3 Locking

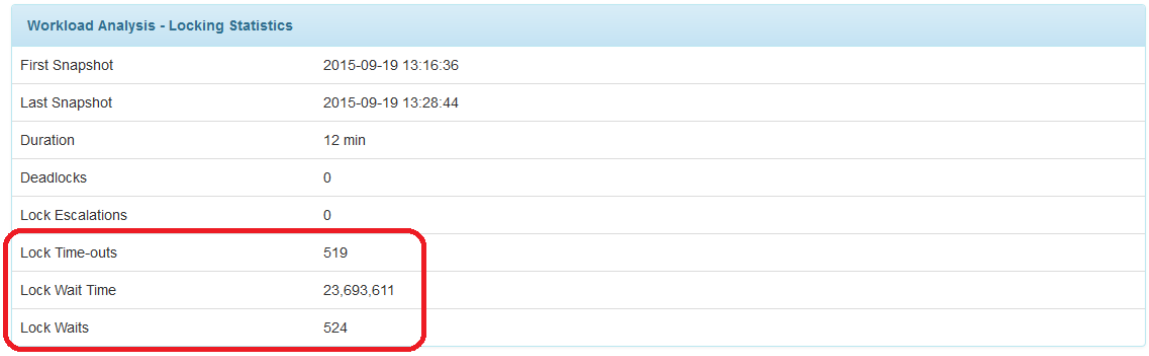
Continue analyzing the report by navigating to the following menu item:

1. Locking

As we know the connections are in lock-wait state, next we'll review the locking menu item.

Under this option you find various locking data, such as statistics, configuration, active lock waits, and queries involved in locking.

Locking statistics show there were 519 timeouts while the dbcollect.sh script was running (13:16 to 13:28), and also significant wait time (in milliseconds). Compare these values with the baseline report.

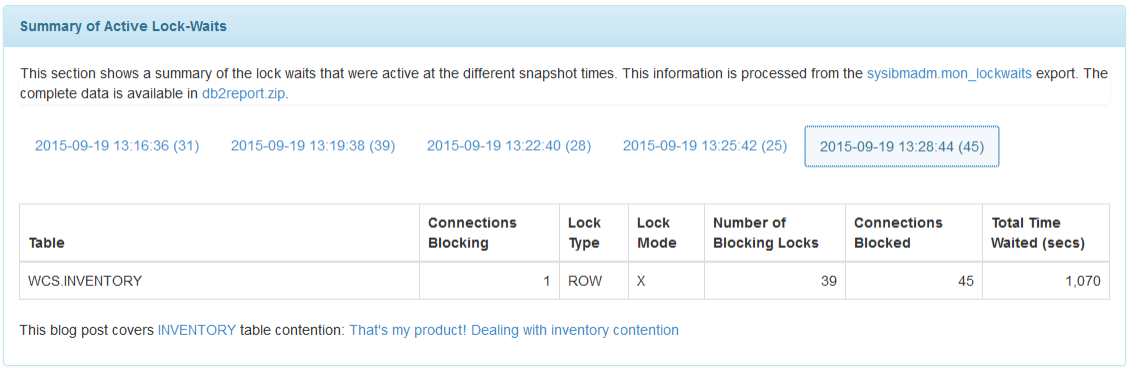


Workload Analysis - Locking Statistics

First Snapshot	2015-09-19 13:16:36
Last Snapshot	2015-09-19 13:28:44
Duration	12 min
Deadlocks	0
Lock Escalations	0
Lock Time-outs	519
Lock Wait Time	23,693,611
Lock Waits	524

4.2.3.1 Summary of Active Lock Waits

The Summary of Active Lock-Waits table lists summarized information for the active lock waits at the time of each snapshot.



Summary of Active Lock-Waits

This section shows a summary of the lock waits that were active at the different snapshot times. This information is processed from the sysibmadm.mon_lockwaits export. The complete data is available in db2report.zip.

2015-09-19 13:16:36 (31) | 2015-09-19 13:19:38 (39) | 2015-09-19 13:22:40 (28) | 2015-09-19 13:25:42 (25) | 2015-09-19 13:28:44 (45)

Table	Connections Blocking	Lock Type	Lock Mode	Number of Blocking Locks	Connections Blocked	Total Time Waited (secs)
WCS.INVENTORY	1	ROW	X	39	45	1,070

This blog post covers INVENTORY table contention: That's my product! Dealing with inventory contention

Using the last snapshot (13:28) as an example, the table is interpreted as follows:

There is a single connection (Connections Blocking), holding at least 39 write row locks on the INVENTORY table. These locks are blocking 45 other connections (Connections blocked).

4.3 Interpreting the database data

The DB2 Report tool is meant as a high level, non-intrusive report of the health of the database. As such, when you find a problem, you might need to do further troubleshooting to complete root cause analysis.

In this case we know the problem is related to inventory. Inventory problems can arise when:

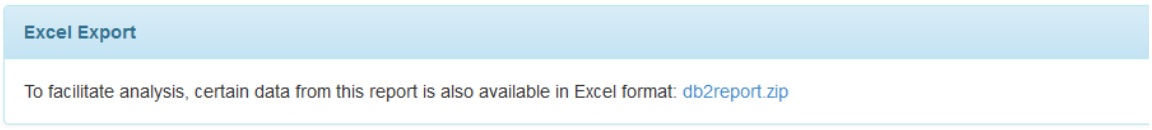
a) All shoppers are buying the same products (e.g. free gift promotion)
b) A backend job is updating and locking the table

The fact that in the Summary of Active Lock-Waits table there is always a single locker, and that the Active Statements captured a select statement for all inventory records for a store, seems to indicate that a batch job is responsible for the locking.

4.3.1 Using the Excel data

As the data in the web report is aggregated and summarized, to get more specific data you can use the Excel report which contains all the raw data as extracted from the db2collect zip file.

A link to the Excel report can be found in the Overview panel:



Each report of the active lock waits shows a single locker. The question is: is it always the same locker across snapshots?

Complete these steps:

1. From the Overview panel, download db2repot.zip
2. Extract the Excel file within the zip - db2report_MALL_2015-09-19-13.28.46.xls
3. Find the lockwait tab. The data in this tab was extracted using this view:

   MON_LOCKWAITS administrative view
   http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.sql.rtn.doc/doc/r0056601.html

4. The lockwait sheet contains multiple snapshots (SNAPSHOT_TIME) column. Review the locks captured.
5. The HLD_APPLICATION_HANDLE column contains the handle of the connection holding the locks. Is it always the same one across snapshots?

Your review should show you that handle 2607 is the single holder of the locks.

REQ_STMT_TEXT	HLD_APPLICATION_HANDLE	HLD_MEMBE	HLD_APPLICATI
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application
TY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATEN	2,607	0	db2jcc_application

With the handle of the connection holding the locks (2607), we can use the connections tab to find more details about the connection, such as the server is coming from and the database user associated to the connection.

Complete these steps:

1. Open the connections tab
2. Locate application_handle 2607
3. Review the information available for the connection

APPLICATION_HANDL	APPLICATION_NAM	APPLICATION_ID	MEMBE	CLIENT_WRKSTNNAM	CLIENT_ACCTN	CLIENT_USER
2,607	db2jcc_application	1.1.1.2.52163.150919171237	0	***1.1.1.2***		
2,620	db2jcc_application	1.1.1.1.57894.150919171501	0	1.1.1.1		
2,041	db2jcc_application	1.1.1.1.41470.150919114728	0	1.1.1.1		

The connection was initiated from server with IP 1.1.1.2, which is used exclusively for batch jobs.

5 Final analysis and conclusion

Analysis of web server data helped us understand the timeline and severity of the production issue.

Analysis of the WebSphere Commerce servers showed that most threads were hung waiting on the database to respond, in particular for an inventory query that is executed as part of checkout (OrderProcess command).

DB2 analysis confirmed the findings in the other layers and showed a large number of connections in Lock-Wait while executing a select for update for inventory data.

The database report also showed that a single connection (2607) was holding all the locks. This connection was coming from the server with IP 1.1.1.2 which is known to be used exclusively for batch jobs.

Further analysis confirmed that a brand new inventory update job was started at 1 PM. This job selected and updated all the inventory records for a store. This was done using a single database transaction that could last several minutes.

5.1 Resolving the problem

After working with the developers, the code of the custom inventory job was updated to implement multiple transactions. This allows freeing up the locks more quickly and minimizes the impact to shoppers.

6 Optional activities

The following are optional activities:

6.1 Analyzing Java performance with the Performance Measurement Tool (PMT)

Shows how to analyze the WebSphere Commerce servers during the production incident using the Performance Measurement Tool (PMT).

6.2 DB2 Top 10 SQL performance analysis

Shows how the Top 10 SQL reports can help you find misbehaving queries or queries that could be cached.

6.3 Analysis: The inventory table is also queried during add-to-cart. Why was locking only happening during order process?

Discusses how the inventory table is queried from the storefront and the use of the currently committed DB2 setting.

6.1 Analyzing Java performance with the Performance Measurement Tool (PMT)

Another way to analyze performance in the WebSphere Commerce layer is by using the Performance Measurement Tool (PMT).

WebSphere Commerce Fix Pack 9 includes a new series of performance loggers. The Performance Measurement Tool can then be used to parse the logs (trace.log) and generate performance reports.

Performance measurement loggers
http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/refs/rlsperfmeasureservicelog.htm

Although the logger can be left enabled, it can be verbose in production. To troubleshoot this problem the logger was only enabled for a few minutes.

The trace specification used is as follows:
`com.ibm.commerce.foundation.logging.service.*=FINE`

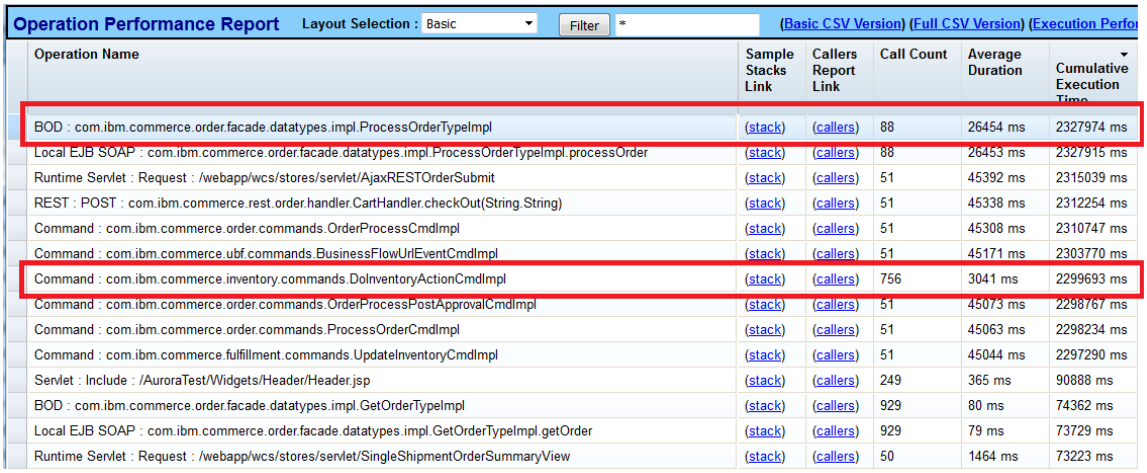
For convenience, the PMT reports were already generated using generatePerformanceReport.bat

Using the Performance Measurement tool
http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/tasks/tdccachemeasuring.htm

Complete these steps:

1. Pre-generated PMT reports are available under the *performanceReport* folder for both, the baseline and error time. Select the error folder
2. Open report-operations.html inside the performanceReport folder
3. Inside the Operation Performance Report, sort by Cumulative Execution Time in descending order. This highlights the operations taking the most time.

Your screen should look as follows:



Operation Performance Report Layout Selection : Basic Filter * (Basic CSV Version) (Full CSV Version) (Execution Perfo

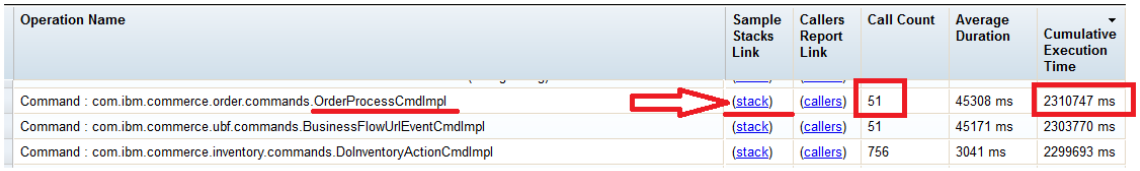
Operation Name	Sample Stacks Link	Callers Report Link	Call Count	Average Duration	Cumulative Execution Time
BOD : com.ibm.commerce.order.facade.datatypes.impl.ProcessOrderTypeImpl	(stack)	(callers)	88	26454 ms	2327974 ms
Local EJB SOAP : com.ibm.commerce.order.facade.datatypes.impl.ProcessOrderTypeImpl.processOrder	(stack)	(callers)	88	26453 ms	2327915 ms
Runtime Servlet : Request : /webapp/wcs/stores/servlet/AjaxRESTOrderSubmit	(stack)	(callers)	51	45392 ms	2315039 ms
REST : POST : com.ibm.commerce.rest.order.handler.CartHandler.checkOut(String,String)	(stack)	(callers)	51	45338 ms	2312254 ms
Command : com.ibm.commerce.order.commands.OrderProcessCmdImpl	(stack)	(callers)	51	45308 ms	2310747 ms
Command : com.ibm.commerce.ubf.commands.BusinessFlowUrlEventCmdImpl	(stack)	(callers)	51	45171 ms	2303770 ms
Command : com.ibm.commerce.inventory.commands.DoInventoryActionCmdImpl	(stack)	(callers)	756	3041 ms	2299693 ms
Command : com.ibm.commerce.order.commands.OrderProcessPostApprovalCmdImpl	(stack)	(callers)	51	45073 ms	2298767 ms
Command : com.ibm.commerce.order.commands.ProcessOrderCmdImpl	(stack)	(callers)	51	45063 ms	2298234 ms
Command : com.ibm.commerce.fulfillment.commands.UpdateInventoryCmdImpl	(stack)	(callers)	51	45044 ms	2297290 ms
Servlet : Include : /AuroraTest/Widgets/Header/Header.jsp	(stack)	(callers)	249	365 ms	90888 ms
BOD : com.ibm.commerce.order.facade.datatypes.impl.GetOrderTypeImpl	(stack)	(callers)	929	80 ms	74362 ms
Local EJB SOAP : com.ibm.commerce.order.facade.datatypes.impl.GetOrderTypeImpl.getOrder	(stack)	(callers)	929	79 ms	73729 ms
Runtime Servlet : Request : /webapp/wcs/stores/servlet/SingleShipmentOrderSummaryView	(stack)	(callers)	50	1464 ms	73223 ms

The report shows Order Process and Inventory commands as the most time consuming, confirming the findings from the Health Center report.

The Performance Measurement Tool is also able to link operations to their callers, and to display invocation trees.

Complete these steps:

1. Select the stack link for the OrderProcess command, to find the operations inside OrderProcess that take the most time:



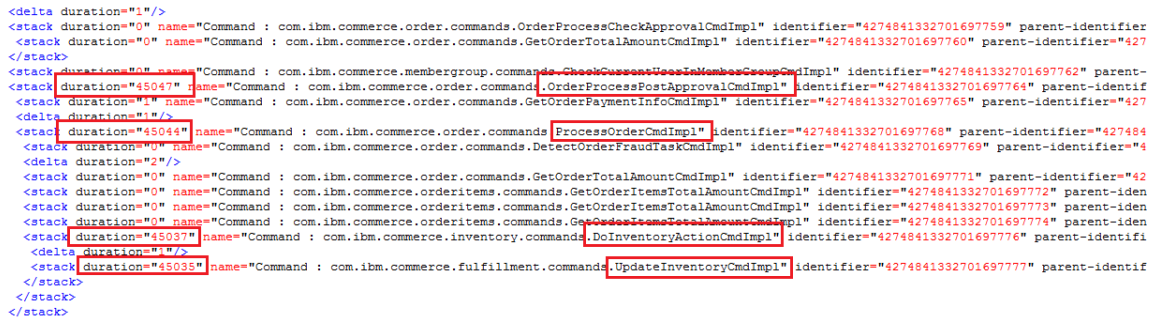
Operation Name	Sample Stacks Link	Callers Report Link	Call Count	Average Duration	Cumulative Execution Time
Command : com.ibm.commerce.order.commands.OrderProcessCmdImpl	(stack)	(callers)	51	45308 ms	2310747 ms
Command : com.ibm.commerce.ubf.commands.BusinessFlowUrlEventCmdImpl	(stack)	(callers)	51	45171 ms	2303770 ms
Command : com.ibm.commerce.inventory.commands.DoInventoryActionCmdImpl	(stack)	(callers)	756	3041 ms	2299693 ms

2. The stack data is presented in xml format. Three sample stacks are taken for this operation: The fastest, the average, and the slowest. Review the xml file.

   To facilitate analysis, if available, read the xml file using an editor such as Notepad++ with no word wrapping.

3. Using the duration attribute, locate the operations inside OrderProcess that are consuming the most time.

You should find that the UpdateInventoryCmdImpl accounts for most of the OrderProcess time:



This links the OrderProcess to the Inventory command. The finding matches what was found with the WebSphere Commerce Java Health Center Report tool.

6.2 DB2 Top-10 SQL performance analysis

In the WebSphere Commerce DB2 Report tool, under the Performance menu, you can find Top-10 SQL Analysis.

This section of the report shows the most expensive queries by different criteria, such as execution time, number of executions, rows reads and others. It can very useful to highlight problematic queries.

6.2.1 Generating a new report

Use the WebSphere Commerce DB2 Report tool to generate a report with data collected during the problem time:

1. Access the WebSphere Commerce DB2 Report tool:

   http://wcsupport.mybluemix.net/wcdb2/

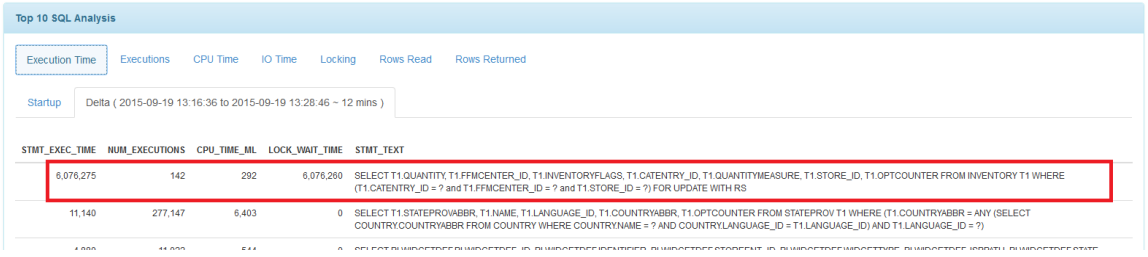
2. Use this file to generate a report:

   error\ db2collect.2015-09-19-13.28.46.zip

6.2.2 Reviewing the most expensive queries

When a query is problematic, you often see that its cost is orders of magnitude higher than any other.

Reviewing the top queries by execution time shows the inventory select. You can see that the lock_wait_time value is almost the same as the stmt_exec_time.



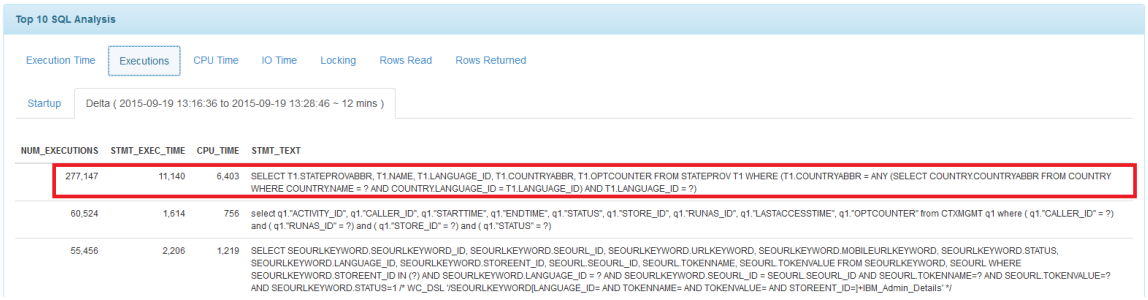
If there were other slow queries, such as due to a missing index, they would also show in this report.

Reviewing the top queries by number of executions shows something interesting, that although is not related to the outage, is something that can be fixed.

The top query by number of executions executes 4 times more than the second highest. The query is to retrieve State/Province and Country data:

SELECT T1.STATEPROVABBR, T1.NAME, T1.LANGUAGE_ID, T1.COUNTRYABBR, T1.OPTCOUNTER FROM STATEPROV T1 WHERE (T1.COUNTRYABBR = ANY (SELECT

COUNTRY.COUNTRYABBR FROM COUNTRY WHERE COUNTRY.NAME = ? AND COUNTRY.LANGUAGE_ID = T1.LANGUAGE_ID) AND T1.LANGUAGE_ID = ?)



State/Province and Country data is ideal for caching:

1. The data set is very small
2. Data almost never changes
3. Very high hit ratio expected (few cache entries prevent a high number of database queries)

WebSphere Commerce does offer caching for most of these objects. Review the list of beans that can be cached with the Data Cache:

Logical cache names and the DistributedMaps they use by default
http://www-01.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/refs/rdclogcachnameDM.htm

The Data Cache defines caches with name CountryCache and StateProvinceCache inside the WCSystemDistributedMapCache distributed map:

WCSystemDistributedMapCache DistributedMap

com.ibm.commerce.user.objsrc.OrganizationCache
com.ibm.commerce.taxation.objsrc.CountryCache
com.ibm.commerce.taxation.objsrc.StateProvinceCache
com.ibm.commerce.user.objsrc.MemberGroupCache
com.ibm.commerce.user.objsrc.RoleCache
com.ibm.commerce.fulfillment.objsrc.ShippingModeCache

When these caches are enabled, queries for the STATEPROV/COUNTRY tables are only expected during cache warm up.

Next Steps:

As WebSphere Commerce provides caching for these objects, validate that the Data Cache is enabled and sized correctly, in particular the WCSystemDistributedMapCache distributed map that contains the state/province and country caches.

Enabling WebSphere Commerce data cache

http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/tasks/tdcenabcommdatacache.htm

When reviewing other queries, either because they run too frequently or too slow, the following articles can be of use:

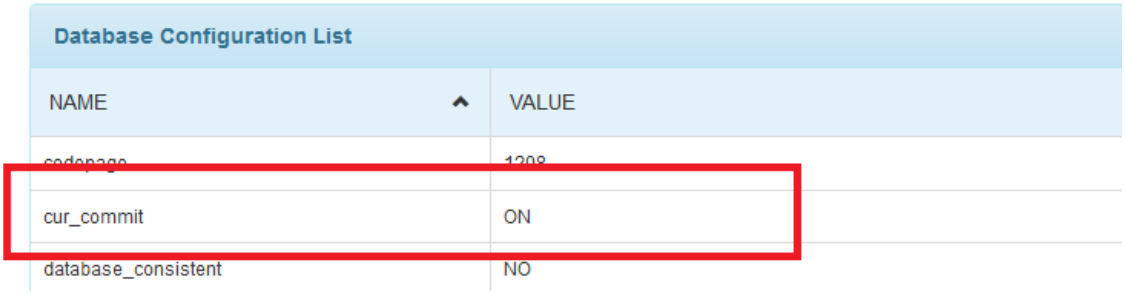
1. Performance: Why is this SQL running so frequently?
   https://www.ibm.com/connections/blogs/wcs/entry/performance_why_is_this_sql_running_so_frequently

2. Tips for troubleshooting slow queries (DB2)
   https://www.ibm.com/connections/blogs/wcs/entry/tips_for_troubleshooting_slow_queries_db2

6.3 Analysis: The inventory table is also queried during add-to-cart. Why was locking only happening during order process?

The inventory table is queried from different pages in the storefront, including add-to-cart. Why was only OrderProcess failing?

The database uses the cur_commit setting, which is recommended:



With the currently committed (cur_commit) setting, instead of locking, scans using the cursor stability (CS) isolation level return data that is currently committed, meaning values before the write operation.

Currently committed semantics improve concurrency
http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.admin.perf.doc/doc/c0053760.html

During add-to-cart, the inventory query is executed with cursor stability (CS) isolation level, which is the default:

```
SELECT T1.QUANTITY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATENTRY_ID,
T1.QUANTITYMEASURE, T1.STORE_ID, T1.OPTCOUNTER
  FROM INVENTORY  T1
 WHERE  (T1.CATENTRY_ID = ? and T1.FFMCENTER_ID = ? and T1.STORE_ID = ?)
    or  (T1.CATENTRY_ID = ? and T1.FFMCENTER_ID = ? and T1.STORE_ID = ?)
    or  (T1.CATENTRY_ID = ? and T1.FFMCENTER_ID = ? and T1.STORE_ID = ?)
order by T1.CATENTRY_ID
```

Then, even if the row being read is currently being updated by another connection, the currently committed setting comes into play and instead of locking the query, it returns the inventory value that was committed to the database prior to the update operation.

During OrderProcess inventory is updated. The code uses this query to find the inventory rows prior to issuing the update statement:

```
SELECT T1.QUANTITY, T1.FFMCENTER_ID, T1.INVENTORYFLAGS, T1.CATENTRY_ID,
T1.QUANTITYMEASURE, T1.STORE_ID, T1.OPTCOUNTER
  FROM INVENTORY  T1
 WHERE (T1.CATENTRY_ID = ? and T1.FFMCENTER_ID = ? and T1.STORE_ID = ?)
   FOR UPDATE
  WITH RS
```

The select for update explicitly specifies Read Stability (RS) isolation level instead of CS. With RS, the currently committed setting does not come into play and the connection needs to wait for the other connection to release the lock by either committing or rolling back its transaction.

If the cur_commit setting had been off, the impact of the inventory job would have been greater, as also pages reading inventory would have locked.

7 Link Reference

7.1 General

1. WebSphere Commerce Support Tools
   https://wcsupport.mybluemix.net/

2. CSE-WebSphere Commerce blogs
   https://www.ibm.com/connections/blogs/wcs

7.2 WebSphere Commerce IHS Report

3. WebSphere Commerce IHS Report
   https://wcsupport.mybluemix.net/wcihs/

4. Module mod_mpmstats
   http://publib.boulder.ibm.com/httpserv/manual70/mod/mod_mpmstats.html

5. Using IBM HTTP Server 7.0.0.23/8.0.0.4 and later extended module timing diagnostics
   http://publib.boulder.ibm.com/httpserv/ihsdiag/mpmstats_module_timing.html

6. Blog article: mpmstats: The eye to the site
   https://www-304.ibm.com/connections/blogs/wcs/entry/mpmstats_the_eye_to_the_site

7. WebSphere Commerce IHS Report – Community
   https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W02f8b2c4804e_40e0_bbd1_32921368b666/page/WebSphere%20Commerce%20IHS%20Report

7.3 WebSphere Commerce Health Center Report

8. WebSphere Commerce Health Center Report
   https://wcsupport.mybluemix.net/hctd/

9. Health Center agent - Installation
   https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W02f8b2c4804e_40e0_bbd1_32921368b666/page/HC%20-%20Installation

10. IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
    https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=22d56091-3a7b-4497-b36e-634b51838e11

7.4 WebSphere Commerce DB2 Report

11. WebSphere Commerce DB2 Report
    https://wcsupport.mybluemix.net/wcdb2/

12. Performance: Why is this SQL running so frequently?
    https://www.ibm.com/connections/blogs/wcs/entry/performance_why_is_this_sql_running_so_frequently

13. Tips for troubleshooting slow queries (DB2)
    https://www.ibm.com/connections/blogs/wcs/entry/tips_for_troubleshooting_slow_queries_db2

7.5 Inventory

14. Blog: That's my product! Dealing with inventory contention
    https://www.ibm.com/connections/blogs/wcs/entry/that_s_my_product_dealing_with_inventory_contention

7.6 Performance Measurement Tool

15. Blog: Find your Performance Bottlenecks Using Statistics Gathered at Every Layer
    https://www-304.ibm.com/connections/blogs/wcs/entry/find_your_performance_bottlenecks_using_statistics_gathered_at_every_layer

16. CSE PMT Blogs
    https://www-304.ibm.com/connections/blogs/wcs/tags/pmt

17. Performance measurement loggers
    http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/refs/rlsperfmeasureservicelog.htm

18. Using the Performance Measurement tool
    http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/tasks/tdccachemeasuring.htm