From: Spitzer, Andras ([redacted])
Sent: Monday, May 30, 2011 5:25 PM
To: [redacted] ([redacted]) [redacted]
Cc: [redacted] ([redacted]); [redacted] ([redacted]); [redacted] ([redacted]); [redacted] ([redacted]); [redacted] ([redacted])
Subject: Lease Source application "slow", Account Service Timeouts, vfsciohplsrc01 & vfsciohplsrc02

Mark,

Symptom :

Please let me show you what I found in regard to the Lease Source application issue, which is known lately for being "slow" sometimes resulting account timeouts. There have been many performance related troubleshootings before on these two systems, both by GE and the vendors, and even though many recommendations were made as far as I know these are mostly generic recommendations, and the issue is still around as I can see it in the log files.

My first and foremost action was to collect all the background information available in this case, including case numbers (Kintana and Oracle) and I also spoke to one of the app admins, Luis to send me the error message we face related to this timeout issue. That's what I got from Luis :

```
18:19:03,059 ERROR [CSCAdminService] CSCAdminServiceAdminService :: eaSignIn():Exception in logging
Transactionnull
18:20:21,454 ERROR [STDERR] com.[redacted]framework.service.ServiceException: Framework Exception:
Service Exception: Error in getBillingInformation() while calling getBillingInformationThread() for BI
method : Timeout occured
18:20:21,455 ERROR [STDERR]             at
com.[redacted]custconn.acct.service.impl.CSCAcctService.getBillingInformation(CSCAcctService.java:2185)
```

So, the direction of my investigation headed to find out why this timeout happens. I really don't want to go deep describing the architecture, I just started to map what is connected to where a few days ago, so even to me there are unmapped areas, so I really wanted to focus on this error message (marked in red), and the surrounding environment. This error message is coming from a JBoss instance the logfile location is /netnasa001a/lsrc/j2ee/prod/custconn_prod2/log for node02 (custconn_prod1 for node01, it's a shared storage area). As I wrote earlier everyone's guess was that this message is a resource shortage related issue, an intermittent resource shortage makes the application to time out.

Root cause analysis :

A timeout warning can mean many things, especially if it's not a system message, and you can tell mostly by the format of the warning message that it's a debug message by the application. (system warning messages usually have more formal format) So to me, the first step was to find out exactly what this timeout means. Based on the reversed namespace notation (com.[redacted]framework.etc), I had a feeling that the code which generates this message will be found in a Java .class file, also the namespace comfin.[redacted] tells me that most likely an in-house developed Java class I have to look for. I also looked for the error message in the log file (this is just an example) :

```
2011-05-25 18:20:21,206 INFO  [CSCAcctService] CSCAcctService:Error in getBillingInformation() while
calling getBillingInformationThread() for BI method : Timeout occurred
```

This is from /netnasa001a/lsrc/j2ee/prod/custconn_prod2/log/server.log.2011-05-25, which tells us that the symptom was still with us on May 25th. I also found this in the stdout.log :

```
03:25:20,204 INFO  [WebService] Using RMI server codebase: http://3.131.91.88:8083/
```

This line declares a remote repository for Java files, which means I may have difficulties finding the Java file I need. Or at least not in the regular directories, I knew that even though this means the Java files may not be stored locally, it's obvious that most likely I can find a cached copy of it. After running a simply 'find' under /netnasa001a/lsrc/j2ee/prod/custconn_prod2, I found the Java file I was looking for :

```
./tmp/deploy/tmp61629CustomerConnection-1.0.0.ear-contents/CSCWeb-1.0.0-exp.war/WEB-
INF/classes/com/ge/comfin/custconn/acct/client/CSCAcctServiceContext.class
```

You can tell by the naming of the Java .class and the error message that most likely these two are related. (Hopefully this .class file generated this timeout error message) Because I wanted to have a bit broader view on the code which generates this error message, I actually started to look for the .jar file (a .jar file is a structured container for many Java files, basically a Java package) which contains this Java .class file, and I found it really quick :

./tmp/deploy/tmp61629CustomerConnection-1.0.0.ear-contents/CSCWeb-1.0.0-exp.war/WEB-INF/lib/CSCAcct-1.0.0.jar

I grabbed this .jar file to my laptop, and I started to decompile it using JD, which is a very popular Java decompiler. After an hour I had a view of the .class file I was looking for, and I also found the procedure what I needed :

```
public CSCAcctServiceContext getBillingInformation(CSCAcctServiceContext reqServiceContext)
    throws ServiceException, FrameworkException
```

I wish I had the original source code so I can use the comments to understand the code, but during the weekend that was not really an option, so I had to understand what it does only based on the code. I will really try to keep it to the minimum here from now on, I will only focus on the necessary parts to understand what is happening.

We have this function called getBillingInformation and here is the code part which generates the timeout error message :

```
        key1 = "BILLINFO_FINISHED" + userId;
          keyErr = "BILLINFO_ERR" + userId;
          key2 = "ACCOUNTPROFILEBO" + userId;

          if (this.billInfoMap != null)
          {
            this.cscLogger.debug("", "keys to match : " + key1);
            long totWait = 0L;

            while (this.billInfoMap.get(key1) == null) {
              Thread.sleep(this.waitTime.longValue());
              totWait += this.waitTime.longValue();
              if (totWait == this.maxWait.longValue()) {
                this.cscLogger.info("", "Waiting for the other BI call to end" +
cscUserProfile.getUserId());
                hsmOutputMap.put("CSC_ERROR_CODE",
this.cscErrorConstants.getCscErrorCode().getProperty("ERR_BI_FAILURE"));
                hsmOutputMap.put("CSC_ERROR_MESSAGE",
this.cscErrorConstants.getCscErrorMessage().getProperty("ERR_BI_FAILURE"));
                hsmOutputMap.put("SRVCTX_RESPONSE_STATUS", "BI FAILURE");
                this.cscLogger.info("", "Error in getBillingInformation() while calling
getBillingInformationThread() for BI method : Timeout occured");
                throw new ServiceException("Error in getBillingInformation() while calling
getBillingInformationThread() for BI method : Timeout occured");
              }
            }
```

Let me explain the important parts. The important part is that it waits for a condition to happen, if the condition won't happen, it will log the "Timeout occurred" error message we want to resolve. What is the condition? It looks for a global variable called this.billInfoMap (this. refers to the current object we are running in) which is an array and looks for a member called "BILLINFO_FINISHED" + userid. If any member of the billInfoMap array is equal to this string, the code can continue, if not, it will throw an exception, and will send an email with monitoring alert. Basically it's used as a semaphore. Let's see the definition of the billInfoMap variable :

```
 private HashMap billInfoMap = new HashMap();
```

Ok, so it's a HashMap. HashMap is an associative array, normal array members have a number position to identify the value (0,1,2,3, etc), while associative arrays can have anything, strings for example. In short standard arrays have numbers as unique keys, while associative arrays can have not just numbers but anything as unique keys, of course each key can have its own value. So, now we know the condition which will trigger the "Timeout occurred" error message.

Not too far before this condition this function calls another function, called getBillingInformationThread :

```
private void getBillingInformationThread(String userId, String[] billingIds, CSCUserBillingInfo[]
cscUserBillingInfoArr, int threadlen)
```

Here comes the interesting part. This part of the code is multi-threaded, it will get a parameter from its parent function (you can see the last parameter, called thrdlen), and will create a pool of executing threads (or workers as Java calls them), and it starts to process the billInfoMap variable. How many threads a particular processing requires? From our issue standpoint that's irrelevant, in short it can be somewhere between 1 – 30 as I saw in the log files. Going one level deeper, each of these workers are calling this particular function (that's the last level I promise) :

```
private void getBillingInformationThreadImpl(String user, String[] billingIdsForBIInput,
CSCUserBillingInfo[] cscUserBillingInfo, String msg, int thrdlen)
```

This is the level where the code actually decides to "sign off" the condition, to switch the "semaphore", which is checked later on 2 levels above as I described before. Here is the code :

```
        if ((this.billInfoMap.get(key1) != null) && (this.billInfoMap.get(key2) != null)) {
          String[] splited = this.billInfoMap.get(key2).toString().split(";");
          if ((splited.length == cscUserBillingInfo.length) &&
            (this.billInfoMap.get(key1).toString().equals(splited[0]))) {
```

```
            this.billInfoMap.put(keyFlag, "YES");
          }
        }
```

If this condition-jungle appears to be true (I already figured out what it does but I won't explain it here, again, only the parts that are relevant) in any of the executed threads (you remember? This function is called on many separate threads in parallel one level above), the continuing code 2 levels above won't complain and will continue. If this for any reason won't get executed, the code will bail out with "Timeout occurred". We are almost at the end, let me continue. The interesting condition in this condition-jungle is this one : splited.length == cscUserBillingInfo.length

The way it works that each thread appends the userid to key2 of the HashMap (key2 is the unique key used here to access the HashMap) using ; as the separator, and this part of the code will double check if all the threads are complete, by simply checking if the number of threads we started earlier is equal to the number userids in key2. Let me show you how it works normally, from the log files (/netnasa001a/lsrc/j2ee/prod/custconn_prod2/log/server.log.2011-05-18) :

```
2011-05-18 13:49:13,437 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :1
2011-05-18 13:49:13,437 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
omer23L omer230
2011-05-18 13:49:13,437 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null omer23
2011-05-18 13:49:13,443 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :5
2011-05-18 13:49:13,443 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
omer23L omer230
2011-05-18 13:49:13,444 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null omer23;omer23
2011-05-18 13:49:13,450 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :6
2011-05-18 13:49:13,451 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
omer23L omer230
2011-05-18 13:49:13,451 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null omer23;omer23;omer23
2011-05-18 13:49:13,485 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :2
2011-05-18 13:49:13,485 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
omer23L omer230
2011-05-18 13:49:13,485 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null omer23;omer23;omer23;omer23
```

You see, "Ends Thread No:<number>" indicates that a thread had been finished, and you can see in the "Value : " line that another userid (omer23 which is the userid and some digits, won't go into that now, take it as the userid) is appended to the "Value: ". You see, after Thread 1 finished, it's "omer23" (the value after null), after Thread 5 finished it's "omer23;omer23", after Thread 6 finished, it's "omer23;omer23;omer23", etc.

And all the threads are checking how many "omer23"s we have (each thread appends another one), and if it's equal to the number of threads, we are done.

Hope it's clear until now. Now, let me show you what happens when something goes wrong :

```
…
2011-05-25 18:16:01,851 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :1
2011-05-25 18:16:01,851 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
dy135994L dy1359940
2011-05-25 18:16:01,851 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null dy135994;dy135994;dy135994;dy135994
2011-05-25 18:16:02,328 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :2
2011-05-25 18:16:02,328 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
dy135994L dy1359940
2011-05-25 18:16:02,328 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null dy135994;dy135994;dy135994;dy135994;dy135994
2011-05-25 18:16:02,365 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :7
2011-05-25 18:16:02,366 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :9
2011-05-25 18:16:02,366 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
dy135994L dy1359940
2011-05-25 18:16:02,366 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
dy135994L dy1359940
2011-05-25 18:16:02,366 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null dy135994;dy135994;dy135994;dy135994;dy135994;dy135994
```

```
2011-05-25 18:16:02,366 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null dy135994;dy135994;dy135994;dy135994;dy135994;dy135994
```

You see everything looks good until Thread 7. Whenever a thread ends, it appends another username to "Value :". What happens when Thread 7 and 9 ends? It appends the username only once! There you go. This means, at the end the equation of whether the "number of threads" equal to "number of usernames" will be FALSE, as such it will trigger the "Timeout occurred" error message !

And why is this happening? The answer is clear, you may recall we defined the variable which holds these infos as HashMap, from the HashMap's doc :

http://download.oracle.com/javase/1.4.2/docs/api/java/util/HashMap.html :

"**Note that this implementation is not synchronized.** If multiple threads access this map concurrently, and at least one of the threads modifies the map structurally, it *must* be synchronized externally."

This variable type is not thread safe, your code has to make sure there won't be any race condition! (which is not handled here) Not thread safe means that you have to make sure that you access this variable by only one thread at a time, otherwise you'll see issues like this.. Now, please take a look at the date when Thread 7 and Thread 9 ends, they ends pretty much in the same time down to the microseconds… ! Which is exactly when you have a problem with a thread-unsafe implementation.

Another example (you have many, many more in the log files) :

```
2011-05-18 10:36:26,599 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :4
2011-05-18 10:36:26,599 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
gmerkelg2L gmerkelg20
2011-05-18 10:36:26,599 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null gmerkelg2;gmerkelg2
2011-05-18 10:36:26,612 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :1
2011-05-18 10:36:26,612 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
gmerkelg2L gmerkelg20
2011-05-18 10:36:26,612 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null gmerkelg2;gmerkelg2;gmerkelg2
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :7
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Ends
Thread No :19
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
gmerkelg2L gmerkelg20
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Key :
gmerkelg2L gmerkelg20
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null gmerkelg2;gmerkelg2;gmerkelg2;gmerkelg2
2011-05-18 10:36:26,677 DEBUG [CSCAcctService] CSCAcctServicegetBillingInformationThreadImpl():Value :
null gmerkelg2;gmerkelg2;gmerkelg2;gmerkelg2
```

You see, when two threads ends the same time, the code will end up one less username, which will make the condition fail at the end and which will trigger the monitoring with "Timeout occurred", and which will seem to be slowness on the user side as the app had been aborted, and most likely some retransmit/retry will happen after a while.

Summary :

The "Error in getBillingInformation() while calling getBillingInformationThread() for BI method : Timeout occurred" Error message is caused by a bug in the Java application because of using "HashMap" variable in a threaded environment, when "HashMap" is not thread-safe to use. Whenever two parallel threads are trying to modify the billInfoMap variable in the same time, only one of the changes will get stored, resulting a necessarily false condition later on which leads to the error message quoted before.

Regards,
sendai