From: Spitzer, Andras ([redacted])
Sent: Friday, May 06, 2011 9:32 AM
To: [redacted] ([redacted])
Cc: [redacted] ([redacted]) [redacted] [redacted] ([redacted]); [redacted] ([redacted]); [redacted] ([redacted])
Subject: RE: cluster resource issue #6651166 cmfloukqsbgw1/cmfloukqsbgw2

Gergo,

Please find the RCA for the Siebel VCS issue discussed in Kintana #6651166.

Symptom :

This is a recurring issue which dates back even months, but as you'll see in this RCA most likely even further. The case here is about a **2 node VCS cluster** hosting Siebel application, and the business is having a daily recycle of the application initiated by crond. The two nodes are cmfloukqsbgw1 and cmfloukqsbgw2. From time to time even though the shutdown/startup process seems to be clean, during the startup **the application fails over the other node**. The daily shutdown process starts at 2am, while the startup process at 3am. The recycle process consist of two parts, an application server (Siebel SRVR) and an application gateway (Siebel GTWY) component.

Root Cause Analysis:

I started with the most obvious by reading through all the basic VCS log files, trying to understand what is going on. I found two different type of issues. One of them was a Siebel application failure which affects Siebel SRVR, the other was a mount failure affects Siebel GTWY. Depending on which failure occurs, the corresponding component will fail over to the other node.

The Siebel application failure :

This was the easier one, as I was able to tell by the VCS log files that in some cases after the Siebel agent starts up the application, the application process simply disappears, leaving all its child processes as orphans. As a consequence, VCS Siebel monitor shuts down the Siebel SRVR component, and fails over to the other node.

I collected the relevant lines to prove that :

```
2011/03/31 04:26:44 VCS ERROR V-16-2-13067 (cmfloukqsbgw2) Agent is calling clean for
resource(cmfloukqsbsb_app_siebserver) because the resource became OFFLINE
unexpectedly, on its own.
```

This line **indicates that VCS Siebel monitoring detected the application to go offline**, and initiates a fail over. What does that mean exactly? Let's continue with more details, only relevant lines are shown here :

```
2011/04/10 03:13:49 VCS NOTICE V-16-55013-20037 (cmfloukqsbgw1)
Siebel:cmfloukqsbsb_app_siebserver:online:<Siebel::Start> Confirmed startup of Siebel
[SRVR] with PID [4903]
...
ffdc:20178:<Siebel::FFLMStatus> PID [4903] does not exist]
```

These lines shows you that the Siebel VCS agent stats the Siebel SRVR component, and the started process has the Process ID of 4903. Many lines later :

```
ffdc:20178:<Siebel::FFLMStatus> PID [4903] does not exist]
```

At this phase the Siebel monitor realize that the process is missing. Whenever the Siebel VCS agent detects a failure, it dumps a lot of information into the VCS logs. Further confirmation comes from the lines listing all the processes :

```
…
ffdc:10061:Proc:GetProcessListHash:UID:[60321] PID:[6993] PPID[1] COMMAND[siebprocmw 48
2 /cmfloukqsbsb/siebel/siebsrvr/admin/EEF_QA.eefqsbgw.shm 20860 e]
ffdc:10061:Proc:GetProcessListHash:UID:[60321] PID:[5340] PPID[1] COMMAND[siebmtshmw
/cmfloukqsbsb/siebel/siebsrvr/admin/EEF_QA.eefqsbgw.shm 74 20770 1 5]
ffdc:10061:Proc:GetProcessListHash:UID:[60321] PID:[5406] PPID[1] COMMAND[siebprocmw 33
2 /cmfloukqsbsb/siebel/siebsrvr/admin/EEF_QA.eefqsbgw.shm 20783 e]
```

```
ffdc:10061:Proc:GetProcessListHash:UID:[60321] PID:[5459] PPID[1] COMMAND[siebprocmw 34
2 /cmfloukqsbsb/siebel/siebsrvr/admin/EEF_QA.eefqsbgw.shm 20790 e]
…
```

As you can see, all of the Siebel processes are having the Parent PID of 1, which means they are orphan processes. Again, this **confirms that the main Siebel process has quit.**

```
ffdc:20208:<main::MonitorEntryPoint> Resource [cmfloukqsbsb_app_siebserver] NOT online
with state [100] after first level check.
```

Again, further confirmation about Siebel agent detects missing process.

```
ffdc:20066:<Siebel::GetResourceState> did not find all required processes, returning
[100]
2011/04/10 03:17:08 VCS ERROR V-16-2-13067 (cmfloukqsbgw1) Agent is calling clean for
resource(cmfloukqsbsb_app_siebserver) because the resource became OFFLINE unexpectedly,
on its own.
2011/04/10 03:18:35 VCS ERROR V-16-1-10205 Group cmfloukqsbsb_grp is faulted on system
cmfloukqsbgw1
```

And finally, the Siebel SRVR component goes into the faulted state, group failover initiated. Just to help our application team I also discovered which process VCS Siebel agent is missing in this case :

```
siebelom  8064     1   0 13:31:37 ?           0:41 siebsvc -s siebsrvr -a -g
3.151.208.47 -e EEF_QA -s eefqsbgw /g 3.215.160.57:23
```

Also an additional information, VCS Siebel agent stores the PID of this process in /tmp/.VRTSSiebel/cmfloukqsbsb_app_siebserver/PID/pidfile.SRVR.

The mount failure :

The 'application failure' issue I described before affects the Siebel SRVR component, the mount issue described affects the SRVR GTWY component. This was a bit more tricky, as the symptom was pretty generic :

```
2011/04/15 03:47:44 VCS INFO V-16-2-13001 (cmfloukqsbgw1)
Resource(cmfloukqsbgw_mnt_siebel78eefqa): Output of the completed operation (online)
UX:vxfs mount: ERROR: V-3-21264: /dev/vx/dsk/siebel78eefqadg/siebel78eefqa is already
mounted, /siebel78eefqa/apps/siebel78eefqa is busy,
                allowable number of mount points exceeded
/siebel78eefqa/apps/siebel78eefqa:
UX:vxfs mount: ERROR: V-3-21264: /dev/vx/dsk/siebel78eefqadg/siebel78eefqa is already
mounted, /siebel78eefqa/apps/siebel78eefqa is busy,
                allowable number of mount points exceeded
file system is clean - log replay is not required
UX:vxfs mount: ERROR: V-3-21264: /dev/vx/dsk/siebel78eefqadg/siebel78eefqa is already
mounted, /siebel78eefqa/apps/siebel78eefqa is busy,
                allowable number of mount points exceeded
file system is clean - log replay is not required
UX:vxfs mount: ERROR: V-3-21264: /dev/vx/dsk/siebel78eefqadg/siebel78eefqa is already
mounted, /siebel78eefqa/apps/siebel78eefqa is busy,
                allowable number of mount points exceeded
2011/04/15 03:47:52 VCS INFO V-16-1-10298 Resource cmfloukqsbsb_mnt_siebel78eefdr
(Owner: unknown, Group: cmfloukqsbsb_grp) is online on cmfloukqsbgw1 (VCS init
iated)
2011/04/15 03:49:48 VCS ERROR V-16-2-13066 (cmfloukqsbgw1) Agent is calling clean for
resource(cmfloukqsbgw_mnt_siebel78eefqa) because the resource is not up even after
online completed.
```

This looks pretty straightforward for the first look, on the other side it simply doesn't make any sense. I tried many things as a start here to find out why VCS could not mount successfully this mountpoint, I had the idea of system overload, or I/O overload. The case looked simple : when VCS Mount agent can't mount the specified filesystem 4 times in a row, it will give up trying and initiates a failover on the depending group. What also made this symptom weird is that this resource group has more than one filesystems to mount, and always this specific one fails only.

The question was **why would fail VCS mounting this specific filesystem**. Again, the mount failure seemed stochastic, so I had to set up various monitoring, looking for any resource shortage which might result this symptom.

Even though I saw some usage spikes during this period, none of them indicated this. My next guess was that a process is holding an open file descriptor on the mount directory, which would block the mount process. This is not obvious as the shutdown process completes without an error at 2am, this process includes forced unmounts and diskgroup deports, this means that if there is any process holding a file descriptor for that directory, it must have started after the shutdown complete, or after 2am in the morning and before the startup starts, or 3am in the morning.

How can you tell if a directory is having an open file descriptor? Fuser or lsof. Lsof didn't work because the filesystem was VxFS, and lsof can't resolve VxFS file handlers by default. I tried fuser, it didn't show anything. The check I did was to run fuser every 15 sec between 2am and 3am. Nothing found. Fuser showed me no PID. Still, I felt something is missing. The problem with both fuser and lsof is that both will show us a snapshot of the system, a point in time state of the open file descriptor table. I wanted something more subtle solution, I didn't want a snapshot, I want a continuous monitoring of the system. Back to DTrace, so I wrote a short D script to monitor any processes opening that directory, or anything below, with a timestamp (script called open.D). I also wrote a short D script to monitor processes executed, I wanted to have a detailed view on the repeated mount commands ran by the VCS mount agent.

Here is what I saw :

```
2011 May  1 03:12:58    4690    4678       0 exec-success  mount
/dev/vx/dsk/cmfloukqsbsbdg/siebsrvr /cmfloukqsbsb
2011 May  1 03:13:01    4690    4678       0 exit          mount
/dev/vx/dsk/cmfloukqsbsbdg/siebsrvr /cmfloukqsbsb

2011 May  1 03:13:03    4967    4954       0 exec-success  mount
/dev/vx/dsk/siebel78eefqa_dg/apps /SIEBEL78EEFDR/apps/SIEBEL78EEFDR
2011 May  1 03:13:08    4967    4954       0 exit          mount
/dev/vx/dsk/siebel78eefqa_dg/apps /SIEBEL78EEFDR/apps/SIEBEL78EEFDR

2011 May  1 03:13:04    5129    5109       0 exec-success  mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef
2011 May  1 03:13:04    5129    5109       0 exit          mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef

2011 May  1 03:13:04    5151    5147       0 exec-success  mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef
2011 May  1 03:13:04    5151    5147       0 exit          mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef

2011 May  1 03:13:06    5221    5109       0 exec-success  mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef
2011 May  1 03:13:06    5221    5109       0 exit          mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef

2011 May  1 03:13:07    5281    5109       0 exec-success  mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef
2011 May  1 03:13:07    5281    5109       0 exit          mount
/dev/vx/dsk/siebel78eefqadg/siebel78eefqa /siebel78eefqa/apps/siebel78eef
```

Exec-success means the process start time, while exit shows us the time process finished. The first column is the timestamp, second is the PID, third is the PPID, fourth is the UID, fifth is the start/stop, and the sixth is the command itself. Ok, nothing special to see here, although you can see that the first two mounts ran only once (obviously because they were successful), while the mount of /siebel78eefqa/apps/siebel78eef is repeated three times. In this case the group didn't fail over as last try, the third succeeded. (It has to fail 4 times before the resource is marked as faulted)

As my next move, I **wanted to see exactly what is going on inside mount**, so I used truss to parse the VCS mount process, and found the following :

```
8442:   7033.3269       mount("/dev/vx/dsk/siebel78eefqadg/siebel78eefqa",
"/siebel78eefqa/apps/siebel78eefqa", MS_DATA|MS_OPTIONSTR, "vxfs", 0x000C9E60, 64,
0x00167C28, 1024) Err#16 EBUSY
```

That's it. This made my theory stronger, as it shows that the mount has failed because the mount point was busy. This means that the theory I set up earlier that a process which starts after 2am but before 3am is holding an open file descriptor on the mount directory, seems to be correct. This is what I tried to check with fuser, without success, but DTrace helped me to strengthen my theory. Now, I had to find which process is causing us headache this time. As fuser couldn't help me, I used my open.D script to see what processes are using that mount directory . Let's see what I found :

```
Started at 2011 May  4 03:07:13
2011 May  4 03:11:12 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:12 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:12 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:12 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:12 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:13 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
2011 May  4 03:11:13 17334  -1   runmqlsr_nd
/siebel78eefqa/apps/siebel78eefqa/data//qmgrs/SIEBEL78EEFQA/errors/AMQERR03.LOG
```

Hm… First colum timestamp, second is the return code of the syscall (-1 means unsuccessful), third column process, fourth filename it opens. Now that's weird. There are two very interesting thing I found here. The process that is continuously put pressure on the mount directory is process runmqlsr which is the MQ Listener process. **This is interesting for two reasons** :

1. At this stage, **no Siebel process should be running**, not even the diskgroups are mounted yet ! (we are at the very beginning of the startup process here after 3am, and we shut down the app already at 2am !)

2. Why would the MQ listener try to open an error log 20+ times a second (!), I guess that relates to point 1. It shouldn't even run

So I found the reason why VCS Siebel agent fails to mount /siebel78eefqa/apps/siebel78eefqa in a random fashion… The mount command initiated by the VCS Siebel agent is competing with runmqlsr process. This is called race condition, and it's an unhealthy scenario. Even though runmqlsr can't open the file under the mount dir, (open system call returns with -1 indicating file does not exist obviously because filesystem is not mounted), so there is no permanent file handler open, but because it tries to open 20+ times a second, it means it will hold the mount dir for short periods of time resulting a chance to conflict with the mount issued by VCS Mount agent.

In short **when this race condition happens, 4 times in a row, the mount resource fails** and the resource group is initiated for a fail over.

That's the first part of the mount issue. **Now we know the reason why VCS Mount fails occasionally**, now I continue with tracing back further about why the Siebel listener process is still running after VCS shut down the application, which lead us toward the solution.

A quick recap : at 2am every day there is a cron initiated shutdown of the Siebel application (GTWY and SRVR), and one hour later, at 3am VCS initiates a startup of the same application on the same nodes. At this point we identified a Siebel process which is running still at 3am and which is causing race condition for the startup process, more precisely to the Mount agent of the startup process.

Questions left :

- What runmqlsr process is about?
- Why this Siebel process, runmqlsr is still running at 3am when it should have quit at 2am with the rest of the Siebel processes ?
- Bonus question : I mentioned that runmqlsr starts it's heavy I/O pressure a few minutes before the 3am, the app start time. Why it's not thrashing errors for the whole duration between 2am and 3am?

The runmqlsr process is part of the MQ WebSphere application. The central piece of the MQ WebSphere is the Queue Manager (qmgr), and connections coming from the network are served through the Siebel listener.

Now I had to find out why this process keep running after the VCS shutdown at 2am. Because the shutdown is controlled via a VCS agent, I had to look at the VCS agent files. After using again the exec.D DTrace program I was able to trace back that this program is controlled by the MQMCustom VCS Agent. First I found a modified version of the original online/offline scripts, then I started to compare them, and I found the following :

```
mfloukqsbgw1# diff online online.orig
136d135
<       # Modified to add script to fire off command server and listener ComFin Siebel
138d136
```

```
<       $res=system("/opt/VRTSvcs/bin/MQMCustom/$qmname.mqstart.sh $userid $qmname");

cmfloukqsbgw1# diff offline offline.orig
237d236
<       system("/opt/VRTSvcs/bin/MQMCustom/$qmname.mqstop.sh $userid $qmname");
246d244
<       system("/opt/VRTSvcs/bin/MQMCustom/$qmname.mqstop.sh $userid $qmname");
260d257
<       system("/opt/VRTSvcs/bin/MQMCustom/$qmname.mqstop.sh $userid $qmname")
```

Now, full of surprises! We have a manually modified version of the online and offline scripts (The lines begin with < are the manually added lines). Someone added a line to the official online/offline script for calling /opt/VRTSvcs/bin/MQMCustom/$qmname.mqstop.sh whenever there is an offline process, and calling /opt/VRTSvcs/bin/MQMCustom/$qmname.mqstart.sh when there is a startup process. What these manually added scripts do :

```
cmfloukqsbgw1% grep lsr SIEBEL78EEFQA.mqstop.sh
su - $APP_USER -c "cd $MQ_HOME; ./endmqlsr -m SIEBEL78EEFQA"
cmfloukqsbgw1% grep lsr SIEBEL78EEFQA.mqstart.sh
su - $APP_USER -c "cd $MQ_HOME; ./runmqlsr -t tcp -p 1414 -m SIEBEL78EEFQA 1>/dev/null
2>&1" &
```

These are the scripts that control the Siebel listener process. So, it's a manual modification to the official MQMCustom agent that whenever there is a shutdown or startup, the MQMCustom script will call the startup/shutdown of the Siebel listener too. The startup works, no doubt, but why the shutdown doesn't work? The application version is MQ WebSphere 5.3, and there is a simple rule that if the queue manager is still running, the listener won't shut down (this changed in version 6). As such, you can only ask the listener to quit, after the queue manager stopped running, if it's running still, it won't quit. Ok, let me check exec.D again, this time I wanted to look for the execution dates for endmqlsr which is responsible for stopping the Siebel listener and endmqm which is responsible for stopping the queue manager :

```
2011 May  3 02:10:02    6844       1   30310 exec-success  endmqm -i SIEBEL78EEFQA
2011 May  3 02:10:21    6844       1   30310 exit          endmqm -i SIEBEL78EEFQA

2011 May  3 02:10:02    6846    6845       0 exec-success  su - mqm -c cd /opt/mqm/bin;
./endmqlsr -m SIEBEL78EEFQA
2011 May  3 02:10:02    6846    6845   30310 exit          -sh -c cd /opt/mqm/bin;
./endmqlsr -m SIEBEL78EEFQA
```

Just look at the dates. Both endmqm and endmqlsr starts at 02:10:02, but endmqm finishes at 02:10:21 while endmqlsr finishes immediately! This is absolutely wrong, because we should execute endmqlsr only **after** endmqm finished! (remember, we can stop the listener only after the queue manager quits)

Let's see that offline script again :

```
/opt/VRTSvcs/bin/MQMCustom/offline :

        system("su - ".$userid." -c \"endmqm -i ".$qmname." \&\" >/dev/null 2>&1");
        system("/opt/VRTSvcs/bin/MQMCustom/$qmname.mqstop.sh $userid $qmname");
```

Yes, here it is. You can see, the first command endmqm (and which is there by factory default) is ran by & which means the offline script won't wait till it's finish but will continue with the program flow with the second line (and which is added manually to control the listener!), as such it will immediately fire the external script which calls endmqlsr! This is wrong, as it's almost 100% sure that the queue manager is still running when the agent calls the endmqlsr! Btw the & is correct from the Agent standpoint, because even though the endmqm –l command would return only after the queue manager stopped, Veritas intentionally calls endmqm with & because this way it can use its internal timeouts, instead of relying whether endmqm will return or not, and if so, when.

So the problem is not the & sign, but the way this external script calling was put there, it simply doesn't work this way. I needed a final confirmation that endmqlsr is having a problem shutting down the listener, I couldn't find any error message in the logs, so I had to trace the MQMCustom agent, and I saw the following :

```
6593/1:         13133.4728      write(2, " A M Q 9 5 9 6 :   Q u e".., 53)      = 53
```

The process when its run by VCS writes to the standard error (and it's not logged by VCS unfortunately) the error code AMQ9596, which is :

```
"AMQ9596 Queue Manager <insert_3> still running
Severity:

30 : Severe error
Explanation:

The requested operation can not complete because queue manager <insert_3> is still
running.
Response:

End the queue manager and retry the operation."
```

My theory has proved, the endmqlsr command won't even ask the listener to shut down, because the queue manager still running, that's how it can happen that all the Siebel processes gets shut down cleanly at 2am, except the listener, which will keep running and which will create a race condition during the next startup at 3am. Btw if the race condition doesn't occur during the startup, the MQMCustom agent kills the orphan listener and creates a new one.

Also, just want to mention that because of this behavior the /var/mqm filesystems are full on both nodes, so I also would recommend cleaning those up too. The majority of the disk space are error logs, the listener complaining about what I describe here.

Solution :

We identified two issues here :

1. **Application failure issue which will cause the Siebel SRVR component to fail** over randomly during the daily recycle

   Application team have to find out why the process 'siebsvc -s siebsrvr -a -g 3.151.208.47 -e EEF_QA -s eefqsbgw /g 3.215.160.57:23' quits randomly during startups

2. **Mount issue which will cause the Siebel GTWY component to fail** over randomly during the daily recycle

   Application team have to find a way how to fix the MQMCustom online/offline agent so the Siebel listener gets shut down cleanly next to the rest of the Siebel processes. This will stop the stochastic race conditions during a Siebel GTWY component startup.

Additionally a few comments, I recommend to clean up the /var/mqm/errors directories on both nodes to the application team, as the /var/mqm partition is full. Also would like to recommend to check these mentioned above on their other MQ/VCS servers which may have configured similarly, especially with the improper MQMCustom Agent online/offline modification.

Summary:

The root cause of this case goes back to two different issues.

First, whenever a Siebel application startup occurs, sometimes the main Siebel process (siebsvc -s siebsrvr -a -g 3.151.208.47 -e EEF_QA -s eefqsbgw) after a few seconds/minutes disappears leaving all of its child processes as orphans (with Parent PID of 1) and the VCS Siebel monitor puts the resource into a faulted state and initiate a failover. Application team should investigate the reason of this sudden quit.

Second, the mount point issue is caused by the Siebel listener process which doesn't stop during the shutdown process. The shutdown process is scheduled to run every day automatically at 2am as part of a daily app recycle, while the startup process scheduled to run 1 hour later, at 3am every day. In our case the listener process keeps running as the only Siebel process left after a clean shutdown initiated by VCS and as can't communicate with the queue manager it tries to flood the error log files (20+ times a second) with error messages. Unfortunately as the error log file's location is under one of the 3 mount points (/siebel78eefqa/apps/siebel78eefqa) whenever VCS is trying to bring online/mount that filesystem during the startup, a race condition occur between the constantly error logging listener process and the mount command, and in some cases it will result the mount to fail. If it fails 4 times in a row, the resource is moved to the faulted state, and a fail over is initiated. The reason why the listener doesn't stop (it should normally) is because it has been added manually to the MQMCustom Agent's online/offline files, immediately after the queue manager's shutdown. As of MQ WebSphere 5.3 (it's difference since version 6) the Siebel listener can only be shut down after the queue manager has quit.

In this case, as the queue manager shut down and the listener shut down occurs simultaneously, and because the listener requires the queue manager to shut down already, the listener doesn’t shut down.

Regards,
sendai