From: Spitzer, Andras ([redacted])
Sent: Tuesday, May 18, 2010 1:13 PM
To: [redacted] ([redacted])
Subject: RE: Issues with the Sanatander DB Server at LDC - myukswlndcodb01 /02

Hi,

I spent some time analyzing this case. The **symptom is that one of the nodes is usually gets evacuated** because of heartbeat failure. The node which gets evacuated (and rebooted) is usually node 02, although it can happen on 01 as well. Diagnosing an evacuation in an Oracle RAC environment is one of the most complex issues, especially if we use Oracle Clusterware for the cluster membership management as well, and in this case that's what we do. Oracle Clusterware (CRS or Cluster Ready Services) is using many different type of heartbeats, for example disk based heartbeats (using the shared OCR and Voting disks) and network based heartbeats (using the private interconnects).

One of the problems I found is that **all the three voting disks are corrupt**, I know this has been mentioned before, and although I don't believe that's the case why the nodes gets evacuated this should be fixed as well :

```
[    CSSD]2010-04-29 20:24:24.477 [1136679264] >ERROR:   clssnmvReadFatal: voting
device corrupt (0x00000000/0x00000000/2//CRS/voting_03.dbf)
[    CSSD]2010-04-29 20:24:24.504 [1126189408] >ERROR:   clssnmvReadFatal: voting
device corrupt (0x00000000/0x00000000/1//CRS/voting_02.dbf)
[    CSSD]2010-04-29 20:24:24.545 [1115699552] >ERROR:   clssnmvReadFatal: voting
device corrupt (0x00000000/0x00000000/0//CRS/voting_01.dbf)
```

Hopefully you have a backup of the voting disks (using dd), so you can fix this. If you don't, you have to add new ones. (in Oracle RAC 11g both the OCR and the Voting disks are backed up automatically by Oracle, in 10g only OCR is backed up automatically, Voting disks has to be backed up manually, and this Oracle RAC has version 10.2).

Anyway, after analyzing all the logfiles I found out that **the heartbeat is a network heartbeat failure, although it's not the network's fault**. Please let me present you what I found using the example of May 13th, when node 01 was up and running (and was the only member of the cluster), around ~11am Venu asked the DBAs to join 02 to the cluster, and around ~13:42 node 02 got evacuated again. Venu created a tcpdump to see heartbeat on the links, so we can move on to the next stage diagnosing the case (to narrow down the possible components that might cause this evacuation). I also would like to point out that I'm aware that although these series of issues started on April 29th, I can see in the logs that it has happened before a few times (true though not lately, in 2009).

Checking the OCSSD logs we can see that the heartbeat failure was caused by network related issues :

```
[    CSSD]2010-05-13 13:40:59.627 [1241577824] >WARNING: clssnmPollingThread: node
myukswlndcodb02 (2) at 50 3.119720e-317artbeat fatal, eviction in 28.810 seconds
... ( warning keeps repeated)
[    CSSD]2010-05-13 13:43:23.460 [1262557536] >TRACE:   clssnmDoSyncUpdate:
Terminating node 2, myukswlndcodb02, misstime(60000) state(5)
```

The OCSSD heartbeat is using port 49895 protocol TCP (http://download.oracle.com/docs/cd/B19306_01/install.102/b15660/app_port.htm), the UDP traffic we can see between the nodes is caused by Cache Fusion (which is a daemon shares caches between Oracle RAC nodes). Also want to point out that a normal heartbeat message is 84 bytes long, and it takes place once every second, in both directions. This means that both 01 sends it's 84 to 02 and 02 sends it's 84 bytes to 01. We can confirm the port number by checking the OCSSD logs as well :

```
Listener port : [    CSSD]2010-05-13 17:57:27.130 [1189128544] >TRACE:
clssnmClusterListener: Listening on (ADDRESS=(PROTOCOL=tcp)(HOST=myukswlndcodb01_hb)
(PORT=49895))
```

That's correct, the OCSSD port is default here.

Now, let's **see the dump what happened on May 13th** :

First, let me give you a little example how a **healthy network heartbeat** looks like :

```
12:51:11.837875 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
302316:302400(84) ack 302317 win 166 <nop,nop,timestamp 64
12:51:12.079530 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
```

```
302317:302401(84) ack 302400 win 173 <nop,nop,timestamp 18
12:51:12.840007 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
302400:302484(84) ack 302401 win 166 <nop,nop,timestamp 64
12:51:13.081632 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
302401:302485(84) ack 302484 win 173 <nop,nop,timestamp 18
12:51:13.842162 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
302484:302568(84) ack 302485 win 166 <nop,nop,timestamp 64
```

As you can see 84 bytes are coming and going between node 01 and node 02 once in a second. Now, let's see **what happened before the evacuation** :

```
13:39:53.421795 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
547680:547764(84) ack 547681 win 203 <nop,nop,timestamp 67
13:39:53.617106 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547681:547765(84) ack 547764 win 346 <nop,nop,timestamp 18
13:39:53.617142 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547681:547765(84) ack 547764 win 346 <nop,nop,timestamp 18
13:39:54.620549 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547765:547849(84) ack 547848 win 346 <nop,nop,timestamp 18
13:39:55.231348 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547765:547849(84) ack 547848 win 346 <nop,nop,timestamp 18
13:39:56.047772 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547765:547849(84) ack 547848 win 346 <nop,nop,timestamp 18
13:39:57.680768 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
547765:547849(84) ack 547848 win 346 <nop,nop,timestamp 18
13:39:59.827371 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548185:548269(84) ack 547848 win 346 <nop,nop,timestamp 18
13:40:00.622994 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548269:548353(84) ack 547848 win 346 <nop,nop,timestamp 18
13:40:01.243403 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548269:548353(84) ack 547848 win 346 <nop,nop,timestamp 18
13:40:01.697556 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548353:548437(84) ack 547932 win 346 <nop,nop,timestamp 18
13:40:02.615987 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548437:548521(84) ack 548436 win 391 <nop,nop,timestamp 18
13:40:02.616023 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548437:548521(84) ack 548436 win 391 <nop,nop,timestamp 18
13:40:02.639146 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
548436:548520(84) ack 548521 win 203 <nop,nop,timestamp 67
13:40:06.902475 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548521:548605(84) ack 548604 win 391 <nop,nop,timestamp 18
13:40:07.627505 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548857:548941(84) ack 548688 win 391 <nop,nop,timestamp 18
13:40:08.079966 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548857:548941(84) ack 548688 win 391 <nop,nop,timestamp 18
13:40:08.984701 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
548941:549025(84) ack 548688 win 391 <nop,nop,timestamp 18
13:40:10.031172 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
549025:549109(84) ack 549024 win 436 <nop,nop,timestamp 18
13:40:19.623489 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
549865:549949(84) ack 549948 win 482 <nop,nop,timestamp 18
13:40:19.878772 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
549865:549949(84) ack 549948 win 482 <nop,nop,timestamp 18
13:40:21.628918 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
550033:550117(84) ack 550116 win 482 <nop,nop,timestamp 18
13:40:28.622232 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
550621:550705(84) ack 550704 win 527 <nop,nop,timestamp 18
13:40:30.626546 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
550789:550873(84) ack 550704 win 527 <nop,nop,timestamp 18
13:40:32.630388 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
550957:551041(84) ack 550704 win 527 <nop,nop,timestamp 18
13:40:33.622579 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
551041:551125(84) ack 550704 win 527 <nop,nop,timestamp 18
13:40:46.246284 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
550957:551125(168) ack 550704 win 527 <nop,nop,timestamp 1
13:41:23.749311 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
```

```
550704:550788(84) ack 551125 win 240 <nop,nop,timestamp 67
13:41:23.749870 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: .
551125:552573(1448) ack 550788 win 527 <nop,nop,timestamp
13:41:23.749882 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: .
550788:552236(1448) ack 552573 win 285 <nop,nop,timestamp
13:41:23.749892 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: .
552236:553684(1448) ack 552573 win 285 <nop,nop,timestamp
13:41:23.749908 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: .
552573:554021(1448) ack 550788 win 527 <nop,nop,timestamp
13:41:23.750434 IP myukswlndcodb01_hb.49895 > myukswlndcodb02_hb.41575: P
554021:555325(1304) ack 552236 win 572 <nop,nop,timestamp
13:41:23.750444 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: .
553684:555132(1448) ack 555325 win 375 <nop,nop,timestamp
13:41:23.750450 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
555132:555324(192) ack 555325 win 375 <nop,nop,timestamp 6
13:41:35.591941 IP myukswlndcodb02_hb.41575 > myukswlndcodb01_hb.49895: P
555996:556080(84) ack 555325 win 375 <nop,nop,timestamp 67
```
As you can see the **heartbeat stopped indeed** before the evacuation. More precisely node 01 kept sending the heartbeats, but node 02 didn't respond for more than a minute. This proves that the problem is not on the network, as the **packets never left the box in time**. Also would like you to check the numbers in parenthesis, that's the amount of bytes the IP datagram is having. This number is 84 bytes for a standard heartbeat, just look at that after 02 starts replying to the heartbeats it sends packets with the size of 1448 bytes, which explained with that the OCSSD process kept sending the heartbeats to node 01, it **got queued up somewhere inside Linux**, and something has stopped/blocked that amount of data to be sent out to the network.

Now, the next question is what happened here, what blocked the Linux, so the heartbeat packets got hold back for more than a minute? We had OSWatcher running before the evacuation happened, and I was analyzing what might have caused this. This is super difficult now to troubleshoot this further, as we are getting into the Linux kernel and scheduler here. Anway, the system was 98% idle that time, I found only one component which behaves weirdly. It was device "/dev/sdes". iostat shows 100% utilization, even though it almost has no I/O traffic on it (or just a minimal).

/dev/sdes is the boot disk, and it's a logical disk using DELL MegaRaid Perc 5/i controller with two physical disks beneath. I also want to point out that the average service time on /dev/sdes is way out of the normal range. I feel like something is not good there, and this is really important because if the boot disk has issues, the whole system has issues. This can freeze up the system for seconds, and that's what it happens in this case. It's not fully freezing up (because I can see Cache Fusion traffic happens still, although it's UDP which is way more lightweight than TCP), but it can cause a glitch in the TCP/IP stack which holds back heartbeat packets so the system gets evacuated...

To test this, I wrote a small C program. My test was to simulate the heartbeat. The program listens on node 01, the other part of the program connects to node 02, and they just send 84 bytes of data to each other, even second. I wanted to see if that simple test will fail as well (if it does, the heartbeat would fail also!). **I ran my heartbeat test yesterday, and I did experience similar issues, the "heartbeat" got hold back for +30 seconds.**

I'm in the phase of reproducing the issue, if I can reproduce it it means we are one step ahead of finding a proper solution. My current test was to run my "heartbeat simulator" programs on both node 01 and 02, and because I assumed this Kernel lockup is related to the system disks, I started to run bonnie++ on the boot disks to see whether it affects the TCP/IP stack, please find my latest result (I started to run bonnie++ at 10:59) :

```
1274177169 / 1          WRITE      Tue May 18 11:06:09 2010
1274177170 / 1          WRITE      Tue May 18 11:06:10 2010
1274177171 / 1          WRITE      Tue May 18 11:06:11 2010
1274177172 / 1          WRITE      Tue May 18 11:06:12 2010
1274177173 / 1          WRITE      Tue May 18 11:06:13 2010
1274177301 / 128        WRITE      Tue May 18 11:08:21 2010
1274177302 / 1          WRITE      Tue May 18 11:08:22 2010
1274177303 / 1          WRITE      Tue May 18 11:08:23 2010
1274177304 / 1          WRITE      Tue May 18 11:08:24 2010
1274177305 / 1          WRITE      Tue May 18 11:08:25 2010
1274177306 / 1          WRITE      Tue May 18 11:08:26 2010
1274177307 / 1          WRITE      Tue May 18 11:08:27 2010
1274177308 / 1          WRITE      Tue May 18 11:08:28 2010
```

The number after the slash indicates the delta seconds elapsed between heartbeat sent. Normally this should be 1 always, as my program keeps sending heartbeat packets every 1 second. Although when I ran bonnie++ to stress out the boot disk (I stressed out especially the /opt partition, because that's where the application is kept), the **TCP stream got "blocked" for 128 seconds**, which is even double the time Oracle RAC would tolerate. I also want to mention that I was running this heartbeat simulator during all night long (more than 10 hours), and it had no latency at all, and I didn't stress the boot disk either (I was sleeping).. is it a coincidence?

I have to keep testing to prove my idea, but 3 times out of 3 I experience this latency on the network when I put I/O stress on the boot disk, and what is really scary that it affect especially TCP, UDP and ICMP packets can come and go eventually while the TCP stream is blocked randomly. One of the problems I see here is that the application itself sits on the boot disk :

```
[root@myukswlndcodb02 ~]# cd /opt/app/oracle/product/10.2.0/DB
[root@myukswlndcodb02 DB]# df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdes2             50G   20G   28G  43% /opt
```

Obviously the tablespaces are sitting on SAN, but as the application is sitting on the boot disk, it can generate I/O which can cause glitches in the heartbeat. Even though it's not a recommended design to have the application on the boot disk (the boot disk should be kept only for OS related files, all the application should reside on SAN), I'm afraid that even if we move the app to the SAN, we just created a workaround, as any process can generate I/O on the boot disks.

Also, the next step will be to diagnose whether the boot disk (and the surrounding components like RAID Controller) behaves incorrectly, or that's just how the Linux/PC architecture works. Btw, while I was running bonnie++ to generate I/O load on the boot disk, I saw the following (please look at the serve columns, second and third from the right) (iostat -x 60 | grep sdes) :

```
sdes         0.05 10922.64  0.98 158.70    22.01 88574.59     11.00 44287.30    554.82
3094.94 19447.09   6.27 100.05
sdes         0.00 20666.57  0.03 161.76    0.67 167603.27     0.33 83801.63     1035.89
60458.52 27999.18   6.18 100.07
sdes         0.02 4590.70   0.68 166.82    12.27 37237.35     6.14 18618.67     222.38
3350.79 59022.15   5.97 100.05
sdes         0.03 32621.11  0.13 151.65    5.47 264945.83     2.74 132472.92    1745.58
22105.52 25620.91   6.59 100.09
sdes         0.00   0.00    0.00 157.55    0.00    0.00       0.00     0.00     0.00
17120.44 81806.97   6.35 100.05
sdes         0.00  38.68    0.00 176.83    4.67   352.90       2.33   176.45     2.02
7754.93 1739766487874603.50   5.66 100.08
sdes         0.07 9626.95   1.80 153.33    76.28 78075.81    38.14 39037.91
  503.80   570.56 34958.00   6.45 100.03
sdes         0.00 11176.71  0.10 171.58    0.93 90768.73      0.47 45384.36
  528.72   782.94 4569.92   5.83 100.03
sdes         0.00   9.12    0.00 18.22     0.00  117.51       0.00    58.75     6.45
39.85 2182.19  54.90 100.05
sdes         0.00   5.97    0.25  3.22     4.93   73.48       2.47    36.74
22.62    12.03    6.40 288.50 100.03
```

As you can see these numbers are unrealistic (even the realistic ones are way too high!), so it may happen that we have some sort of hardware issue. After I installed the DELL diagnostic tools I found out that controller might have some issues :

```
May 18 09:55:40 myukswlndcodb02 MR_MONITOR[10807]: <MRMON113> Controller ID: 0
Unexpected sense: PD= 1:255, CDB =  0x1c  0x01  0x80   0x04  0x00  0x00 , Sense =
0x70  0x00  0x05  0x00  0x00  0x00  0x00  0x0a  0x00  0x00  0x00  0x00  0x24  0x00
0x00  0x00  0x00  0x00
```

I do recomend to ask DELL support for a health checkup around the boot disks, including both boot disk mirrors plus the RAID controller, because all these symptoms suggests us that something wrong here and it affects RAC's reliability.

Regards,
sendai