[OpenVMS] How to Troubleshoot a Process in RWCSV
Last Technical Review: 5-SEP-1996

PRODUCT: VMScluster Software for OpenVMS AXP
VAXcluster Software for OpenVMS VAX
DIGITAL OpenVMS VAX, All Versions
DIGITAL OpenVMS Alpha, All Versions

COMPONENT: Scheduler

SOURCE: Compaq Computer Corporation

OVERVIEW:

This article describes the RWCSV resource wait state, with some procedures which should help to resolve the issue.

BACKGROUND:

A process enters the RWCSV state when:

a. It's waiting to communicate with the CLUSTER_SERVER process.

b. It's waiting for a cluster wide system service to return completion status from a remote node.

PROCEDURE 1:

1. To determine which node is causing the RWCSV state, issue the following SYSMAN commands:

```
$ MCR SYSMAN
SYSMAN> SET TIMEOUT 00:00:30
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> DO SHOW TIME
```

A node that does not respond to the "SHOW TIME" command, within the 30 second timeout, is probably causing the RWCSV state.

2. Log into the node that does not respond and check the state of the CLUSTER_SERVER process using the DCL command SHOW SYSTEM.

a. If the CLUSTER_SERVER is not running, start it with the following command:

```
$ @SYS$SYSTEM:STARTUP CSP
```

b. If the CLUSTER_SERVER is in a state other than HIB, then the system must be rebooted.

If the problem recurs, ensure that the SYSGEN parameter SCSBUFFCNT is at least 120 on every system in the cluster. If it continues to be a problem, force a crash on that node and analyze the dump file to determine the cause. (See the next PROCEDURE.)

PROCEDURE 2:

The following procedure may be used to determine the node causing the RWCSV state. It is also used to determine if a suspect CLUSTER_SERVER process has exceeded its defined request count, CSP$W_REQCNT.

NOTE: The CSP$W_REQCNT is the lower of 60, or the value of the SYSGEN parameter "SCSBUFFCNT"/2.

The procedure uses the System Dump Analyzer (SDA) to examine a dumpfile, or a running system. Some of the displays in the example have been edited for readability, and knowledge of SDA commands is presumed.

NOTE: Some steps below list different commands for VAX and Alpha systems. If not specified, the commands are the same for both systems.

1. Invoke SDA:

   On the running system:

   ```
   $ ANALYZE/SYSTEM
   ```

   For a crash dump:

   ```
   $ ANALYZE/CRASH filename
   ```

2. Define symbols:

   On VAX:

   ```
   SDA> READ SYS$SYSTEM:SYSDEF
   SDA> READ SYS$SYSTEM:SCSDEF
   SDA> READ/EXECUTIVE
   ```

   On Alpha:

   ```
   SDA> READ SYS$LOADABLE_IMAGES:SYSDEF
   SDA> READ SYS$LOADABLE_IMAGES:SCSDEF
   SDA> READ/EXECUTIVE
   ```

3. Get a list of CSIDs (cluster system identifiers) and cluster node names:

   ```
   SDA> SHOW CLUSTER

                  VAXcluster data structures
                  --------------------------
                  --- VAXcluster Summary ---

   Address   Node    CSID      Votes   State     Status
   -------   ----    ----      -----   -----     ------
   88036DF0  PVDCC   00010025    0     open    member,qf_noaccess
   88009F20  PVJSM   00010020    0     open    member,qf_noaccess
   87FE9E80  PVAMH   00010013    0     open    member,qf_noaccess
   88057C90  MVNS1   00010024    0     open    member,qf_noaccess
   ```

4. Get the process index (Indx) for the process in RWCSV:

```
SDA> SHOW SUMMARY

Current process summary
-----------------------
Extended Indx Process name   Username   State  Pri   PCB      PHD
-- PID -- ---- -------------- ---------- ------ --- -------- ------
 22E00101 0001 SWAPPER                    HIB    16 874E5B98 874E5A00
   ...
 22E0010A 000A CLUSTER_SERVER SYSTEM      HIB     9 87FEA8D0 8B5A7400
 22E0010B 000B OPCOM          SYSTEM      RWCSV   8 87FEA560 8B6F7E00
```

5. Use the Indx number to select the process:

```
SDA> SET PROCESS/INDEX=000B   <--Indx from SHOW SUMMARY command
```

6. Examine the process to be sure of current context and state:

```
SDA> SHOW PROCESS

Process index: 000B   Name: OPCOM   Extended PID: 22E0010B
-------------------------------------------------------
Process status:  00140001   RES,PHDRES,LOGIN

PCB address              87FEA560  JIB address            895CCB90
PHD address              8B6F7E00  Swapfile disk address  00000000
Master internal PID      0001000B  Subprocess count              0
Internal PID             0001000B  Creator internal PID   00000000
Extended PID             22E0010B  Creator extended PID   00000000

State                       RWCSV  Termination mailbox        0000
-->                         ^^^^^
Current priority                8  AST's enabled              KESU
Base priority                   6  AST's active               K
UIC                [00001,000004]  AST's remaining              88
Mutex count                     0  Buffered I/O count/limit     64/64
Waiting EF cluster              0  Direct I/O count/limit       16/16
Starting wait time       19001917  BUFIO byte count/limit    94000/95840
Event flag wait mask     00000010  # open files allowed left    62
Local EF cluster 0       E000203D  Timer entries allowed left   64
Local EF cluster 1       00000000  Active page table count       0
Global cluster 2 pointer 00000000  Process WS page count       365
Global cluster 3 pointer 00000000  Global WS page count          0
```

NOTE: Steps 7 through 9 are not useful on an Alpha system. Go to Step 10.

7. Examine the current stack for the process:

```
SDA> SHOW STACK

Process stacks
--------------
Current operating stack (KERNEL):

SP =>  7FFE770C  881304D0
       7FFE7710  88875A60
       7FFE7714  8887A824
       7FFE7718  00000002
       7FFE771C  888758D8
       7FFE7720  0008C9CA
       7FFE7724  88130508  <-- 7 longwords from the top of the stack
       7FFE7728  88130868      is the CSD (Cluster Server Data block)
         ...
```

8. Get the CSID of the node this system is trying to transfer to.
   This value is in the CSD at offset E (hexadecimal):

```
SDA> EXAMINE 88130508+E

88130516:  00010013
```

9. Look at the instruction stream executed by the process:

```
SDA> EXAMINE/INSTRUCTION @PC-50;50

88875D08:  TSTW    8887A90E  <-- This is the address of
88875D0C:  BLSS    88875D1E      the CSP$W_REQCNT
88875D0E:  BISB2   #80,35(R4)
88875D13:  DECW    8887A90E
88875D17:  MOVL    #03,R1
88875D1A:  MOVL    #01,R0
88875D1D:  RSB
88875D1E:  INCL    8887A914
88875D22:  PUSHL   R4
88875D24:  MOVL    @#CTL$GL_PCB,R4
88875D2B:  ASHL    #10,#02,-(SP)
88875D2F:  BLBC    @#SMP$GL_FLAGS,88875D3F
88875D36:  MOVZBL  #2F,R0
88875D39:  JSB     @#V_SMP$ACQUIRE
88875D3F:  BLBC    @#SMP$GL_FLAGS,88875D4F
88875D46:  MOVZBL  #34,R0
88875D49:  JSB     @#V_SMP$RELEASE
88875D4F:  MOVL    #10,R0
88875D52:  JSB     @#V_SCH$RWAIT
88875D58:  BLBC    @#SMP$GL_FLAGS,88875D6A
```

10. Examine the address for CSP$W_REQCNT:

    On VAX:

    ```
    Examine the value for CSP$W_REQCNT using the address
    determined in the prior step:

    SDA> EXAMINE 8887A90E

    8887A90E:  F02FFFFF
                   ^^^^ this is -1, which indicates that the
                        transfer quota has been exhausted

    NOTE:  This is a word length field.
    ```

    On Alpha:

    ```
    Determine the address for CSP$L_REQCNT as shown in the
    following steps:

    SDA> EXAMINE SYS$CLUSTER_NPRW+840+1A4
    SYS$CLUSTER_NPRW+009E4:  FFFFFFFF   "...."
                             ^^^^^^^^ this is -1, which indicates
                                      that the transfer quota has
                                      been exhausted

    NOTE:  Displacement 840+1A4 is valid for OpenVMS Alpha, V1.5.
           Other versions may require an adjustment to these offsets.
    ```

11. Get the address of the transfer queue 'CSP$Q_ACB_XFER':

    On VAX:

    ```
    Subtract 16 from the CSP$W_REQCNT address:

    SDA> EVALUATE 8887A90E-16
    Hex = 8887A8F8   Decimal = -2004375304
    ```

    On Alpha:

    ```
    SDA> EXAMINE SYS$CLUSTER_NPRW+840+190
    SYS$CLUSTER_NPRW+009D0:  81214940 81524F80   ".OR.@I!."

    NOTE:  As in the step above, the offsets may need to be
           adjusted in versions other than OpenVMS Alpha, V1.5.
    ```

12. Use the CSP$Q_ACB_XFER address to determine the number of outstanding transfers:

    On VAX:

    ```
    SDA> VALIDATE QUEUE 8887A8F8
    Queue is complete, total of 61 elements in the queue
    ```

    On Alpha:

    ```
    SDA> VALIDATE QUEUE SYS$CLUSTER_NPRW+840+190
    Queue is complete, total of 61 elements in the queue
    ```

13. Examine the queue header 'CSP$Q_ACB_XFER' to establish context:

    On VAX:

```
SDA> EXAMINE 8887A8F8
8887A8F8:  88091300
```

    On Alpha:

```
SDA> EXAMINE SYS$CLUSTER_NPRW+840+190
SYS$CLUSTER_NPRW+009D0:  81214940 81524F80   ".OR.@I!."
```

14. Move to the first entry:

```
SDA> EXAMINE @.
88091300:  880941F0
```

15. Get the CSID of the first AQB (ACP Queue Block) in the transfer queue:

```
SDA> EVAL @(@(.+14)+E)
Hex = 00010024   Decimal = 65572  UCB$M_DELETEUCB+00024
```

    Usually the first CSID in the queue is the culprit. The rest of this procedure shows how to traverse the remaining elements in the queue. Normally, this would only be done if the node pointed to by the first entry is not the source of the problem.

16. Look at the next ACB in the transfer queue:

```
SDA> EXAMINE @.
880941F0:  880941C0
```

17. Get the CSID of the second AQB in the transfer queue:

```
SDA> EVALUATE @(@(.+14)+E)
Hex = 00010024   Decimal = 65572  UCB$M_DELETEUCB+00024
```

Repeating the instructions in the previous two steps gets the next entry in the transfer queue and its CSID. Repeat the instructions until a pattern of the same CSID is established. Again, if the same CSID shows up time after time in the transfer queue, that system is the most likely source of the problem.