[OpenVMS] How to Troubleshoot a Process in RWMBX

Last Technical Review: 24-OCT-2000

```
PRODUCT:    DIGITAL OpenVMS VAX, All Versions
            DIGITAL OpenVMS Alpha, All Versions

COMPONENT:  Scheduler

SOURCE:     Compaq Computer Corporation
```

OVERVIEW:

The output from the DCL command "SHOW SYSTEM" displays a process in the RWMBX state (Resource Wait Mailbox Full). A continuous display of one of theses processes may show that the process is accumulating some CPU time, however the majority of the time will be spent in the RWMBX state.

Note:
To produce a continuous display of a process, use the following DCL command;

```
$ SHOW PROCESS/CONTINUOUS/IDENTIFICATION=<pid-number>
```

BACKGROUND:

OpenVMS incorporates a mechanism whereby interprocess communication may occur called a mailbox. A mailbox is the virtual device (MBAn) defined for this use. A process creates mailboxes using the Create Mailbox ($CREMBX) system service and may accessed by multiple processes, typically 1 reader and multiple writers. The RWMBX state is caused by a resource shortage associated with the writing process, mailbox, or system.

Process resources:
- Buffered I/O Limit (BIOLM)
- Buffered I/O Byte Limit (BYTLM)

Mailbox resources:
- Message Limit
- Buffer Limit

System resources:
- Pool memory
- System performance capabilities

PREPARATION:

Use the System Dump Analyzer (SDA) to determine if a process, mailbox, or system resource is responsible for the RWMBX state. This can be accomplished on the live system or dump file.

```
$ SET PROCESS/PRIVILEGES=CMKRNL
$ ANALYZE/SYSTEM
      - or -
$ ANALYZE/CRASH SYS$SYSTEM:SYSDUMP.DMP
VAX/VMS System analyzer
SDA> READ SYS$SYSTEM:SYSDEF          <--- On OpenVMS VAX
SDA> READ SYS$LOADABLE_IMAGES:SYSDEF <--- On OpenVMS Alpha
```

PROCEDURE:

The following procedures outline steps to identify which process, mailbox, or system resources are responsible for causing processes to hang in the RWMBX state. Included in the procedures are steps required to preclude the recurrence of this problem.

DETERMINING PROCESS RESOURCES:

Define the context of the process and determine its available BIOLM and BYTLM resources.

1. Locate the process/es in question.

```
   SDA> SHOW SUMMARY
   Current process summary
   -----------------------
    Extended Indx Process name    Username    State   ....
   -- PID -- ---- --------------- ----------- ------- ....
    30A00201 0001 SWAPPER                      HIB    ....
               :                                :
    30A01034 0034 _VTA495:        J_DOE        RWMBX  ....
                |
                +-------------------------------------------+
                                                            |
2. Use the "Indx" value to view the process resources.      |
                                                            |
   SDA> SHOW PROCESS/INDEX=34  <----------------------------+
   Process index: 0034   Name: _VTA495:   Extended PID: 30A01034
   -------------------------------------------------------------
    Process status:  02040001   RES,PHDRES
             :
    Mutex count                 0  Buffered I/O count/limit   0/18 --+
    Waiting EF cluster          0  Direct I/O count/limit    18/18   |
    Starting wait time  1B001B1B  BUFIO byte count/limit  6/35376 --+
             :                                                       |
                                                                     |
     +---------------------------------------------------------------+
     v
   Note:
       "Buffered I/O count/limit" = BIOLM
       "BUFIO byte count/limit"   = BYTLM
```

BIOLM and BYTLM are resource definitions for the process as specified in the systems User Authorization File (UAF), PQL_D/MBIOLM and PQL_D/MBYTLM parameters in SYSGEN, or /IO_BUFFERED and /BUFFER_LIMIT qualifiers for the RUN command.

The "Limit" is the value originally defined for the process via one of the above methods and "Count" is the value of the unused portion of the original.

The process was granted a limit of 18 buffered I/Os and currently has none remaining.

```
         "Buffered I/O count/limit" = "0/18"
```

If this is the cause of the RWMBX state at least one of the I/Os must complete before the process can request another.

To preclude this resource of being the future cause of this problem, the defined "Limit" can be raised by increasing the parameter or qualifier used to set the original value.

The process was granted a limit of 35,376 bytes of buffer space to process buffered I/Os and currently has 6 remaining.

```
       "BUFIO byte count/limit" = "6/35376"
```

If this process had any "Buffered I/O" left, it would only be able to buffer a six-byte message. If any attempt is made to process a mailbox message of larger than six bytes, the process would go into RWMBX. When any currently outstanding I/Os complete the bytes allocated to buffer the IO will be credited back to the process. If the "Count" value were equal to, or greater than the size of the mailbox request, the process would be able to complete the request.

To preclude this resource of being the future cause of this problem, the defined "Limit" can be raised by increasing the parameter or qualifier used to set the original value.

If the "count" of "Buffered I/O count/limit" is greater than 0 and "BUFIO byte count/limit" is greater than 1056, then process resources are probably not the cause of the RWMBX state.

DETERMINING MAILBOX RESOURCES:

If the basic analysis of process resources is deemed not to be the cause of the state, check the limit for the targeted mailbox's message and buffer limit.

1. Maintaining the above process context, locate the mailbox in question.

```
SDA> SHOW PROCESS/CHANNEL
Process index: 0034   Name: _VTA495:   Extended PID: 30A01034
-------------------------------------------------------------

                     Process active channels
                     -----------------------
Channel  Window           Status       Device/file accessed
-------  ------           ------       --------------------
   0010  00000000                       LFILNG$DUS101:
   0020  818D4940                       LFILNG$DUS101:
   0030  00000000                       MBA9092:  <-- Mailbox
   0040  00000000                       VTA495:
   0050  00000000                       VTA495:
```

Note:
  - OpenVMS VAX

    If no MBAn: device, or more than one MBAn: device is displayed, the register R5 should point to the Unit Control Block (UCB) for the mailbox which caused the RWMBX state. The SDA command "SHOW DEVICE/ADDRESS=@R5" can be used to show the device.

- OpenVMS Alpha

  If more than one MBAn: device is displayed, the register R4 should points to the channel in the SHOW PROCESS/CHANNEL display above for the suspect mailbox.

  Note:
  In version 7.2-1, the R17 register will point to the channel.

```
  SDA> SHOW PROCESS/REGISTER
  Process index: 0034   Name: _VTA495:   Extended PID: 30A01034
  --------------------------------------------------------------
  Current process registers
  -------------------------
  R0   = 00000000 7FF91FC0  R1   = 00000000 7F92B898 ............
  R3   = 00000000 0005E110  R4   = 00000000 30300030 ............
                                                 |  |
    The lower word of R4 indicates the ---------+--+
    mailbox in channel 30 is suspect.
```

- If the suspect mailbox is found to be "MBA1,2,3, or 4", see the "HELPFUL HINTS" section below.

  Note:
  Sometimes the channel number is not accessible and the device or channel cannot be determined. If this happens try mapping at the "Return Address" from a SHOW CALL command. It may tell us how this process got to this point. By looking at the executing code we may be able to determine if this is a resource issue or a synchronization issue.

  On a VAX the SHOW CALL would return information including the RETPC. This address must be manually mapped to a P0, P1, or to a system space address using data from a SHOW PROCESS/IMAGE command.

  On OpenVMS ALpha 7.1, the following SDA command can be used to determine which mailbox is full:

```
    SDA> SHOW DEVICE/ADDRESS= -
                  @(@CTL$GA_CCB_TABLE+(((@R17/10)-1)*20))
```

  (R17 points to the channel number for the open mailbox.)

2. Display the mailbox characteristics and format its UCB to determine its available resources.

```
   SDA> SHOW DEVICE MBA9092:
   I/O data structures
   -------------------
   MBA9092                MBX            UCB address:  80BB4A40
      :                    :                              |
      :                    :                              |
                                                          |
   SDA> FORMAT 80BB4A40  <-------------------------------+
   80BB4A40   UCB$L_FQFL              818A8780
              UCB$L_RQFL
              UCB$W_MB_SEED
              UCB$W_UNIT_SEED
   80BB4A44   UCB$L_FQBL              81808A60
              UCB$L_RQBL
                  :
                  :
   80BB4A54   UCB$L_FIRST             00150000
              UCB$L_FR4                  ^
              UCB$W_MSGMAX               |
              UCB$W_MSGCNT    <----------+--------- Number of messages
                                                    outstanding
   80BB4A58   UCB$W_BUFQUO                 0006 <--- Remaining bytes of
                                                    message buffer space
              UCB$W_DSTADDR
   80BB4A5A   UCB$W_INIQUO            0180     <--- Initial size of
              UCB$W_SRCADDR                         message buffer space
   80BB4A5C   UCB$L_ORB               80BB4AF0
                  :
                  :
```

The field UCB$W_INIQUO is the original size of buffer space for mailbox messages. The Create Mailbox system service ($CREMBX) determines this value using the BUFQUO parameter, or the SYSGEN parameter DEFMBXBUFQUO if BUFQUO was not defined. The field UCB$W_BUFQUO tells how much mailbox buffer space is currently available for messages. If the message the process is trying to write is larger than UCB$W_BUFQUO, the process will be placed in the RWMBX state.

In the example above, the mailbox's initial allocation of 384 bytes (180 hexadecimal), UCB$W_INIQUO, has been depleted leaving only 6 bytes, UCB$W_BUFQUO, for future messages. It also shows that the 378 bytes of buffer space is consumed by 21 (15 hexadecimal), unread messages, UCB$W_MSGCNT. The reading process is the most likely cause of the message back up and the subsequent hang in RWMBX of the writing process.

Follow these steps in an attempt to determine which process should be reading the mailbox.

```
  SDA> SET OUTPUT CHAN.TXT
  SDA> SHOW PROCESS/CHANNEL ALL
  SDA> SET OUTPUT TT:
  SDA> SPAWN
  $ EDIT CHAN.TXT
```

Note:
The above commands produce the CHAN.TXT file which will include the assigned channels of every active process. The channels

associated to a specific process are individually listed under the process.

You should expect to see warning messages regarding processes whose P1-space channel control blocks are currently paged out. These channels are not available via this method.

If you receive an ACCVIO, (Access Violation), using the procedure above, please see the following article:

[OpenVMS] How To Use Programmable EDT To View Process
Information In SDA

If the reader process is located it can be analyzed to determine why the mailbox isn't being read.

Some reasons may include the following:

1. The priority of the reading process has been lowered or originally set very low and the process is being pre-empted by higher priority processes. Raise the priority of the reading process. Its suggested that the reader's priority be at least as high as the writer's.

2. The initial mailbox buffer size (UCB$W_INIQUO) is not large enough to preclude writing processes to be forced to wait in RWMBX before messages can be read. Defining a larger BUFQUO value during mailbox creation, or DEFMBXBUFQUO in SYSGEN should lower the occurrences of the RWMBX state.

   Note:
   The SYSGEN parameter, DEFMBXBUFQUO, is only used to size mailboxes that don't specify a size value when the $CREMBX (Create Mailbox) system service was called.

3. System performance is precluding timely scheduling of the reader process. The cause of the performance degradation must be determined.

If no reading process is located and the process hung in RWMBX must be cleared, the following command may be used in an attempt to flush the mailbox buffer and allow the process to complete. However, if multiple attempts fail to clear the condition, a reboot or restart of the reading process will be required.

```
SDA> SPAWN
$ COPY MBAn: NL:
(Wait for 15-30 seconds then ^y)
$ LOGOFF
SDA>
(Recheck the process and repeat if required.)
```

DETERMINING SYSTEM RESOURCES:

If the analysis of processes provides an indication the system resources are in question, a systems evaluation is required.

1. In some cases, system pool expansion may cause the RWMBX state if OpenVMS is unable to allocate buffer space for the message. Use the following steps to determine pool expansion or pool allocation failures have occurred on the system.

```
  $ RUN SYS$SYSTEM:AGEN$FEEDBACK

On OpenVMS VAX 6.0 and above, and OpenVMS Alpha
-----------------------------------------------
  $ SEARCH SYS$SYSTEM:AGEN$FEEDBACK.DAT NPAGEDYN
  NPAGEDYN_CUR = 749568
  NPAGEDYN_PEAK = 814080
  NPAGEDYN_ALLOCFAIL = 0
         :
         :

On OpenVMS VAX 5.0 through 5.5-2
--------------------------------
  $ SEARCH SYS$SYSTEM:AGEN$FEEDBACK.DAT SRP
  SRPCOUNT_CUR = 1024
  SRPCOUNT_PEAK = 1024
  SRP_ALLOCFAIL = 0
```

   Note:
   The "_CUR" field represents the current SYSGEN value for the indicated parameter.

   If the "_CUR" value is greater than the "_PEAK" value, regardless of which operating system is being evaluated, then pool expansion has occurred and is the possible cause for the RWMBX state. This would be further supported by a non-zero value for the "_ALLOCFAIL" field.

2. Use the MONITOR utility to check the occurrence of certain process states. The occurrence of certain other process states may indicate that the system may not be able to schedule the reader process frequently enough.

   Note:
   the following MONITOR display has been modified for this article to include only those fields deemed pertinent to the article.

```
$ MONITOR/NODISPLAY/SUMMARY=MON.SUM/END="+00:10" STATES
$ TYPE MON.SUM

                        VAX/VMS Monitor Utility
                             PROCESS STATES
                             on node nnnnnnn
                                SUMMARY

                               CUR    AVE    MIN    MAX
  Mutex & Misc Resource Wait  0.00   0.00   0.00   0.00
  Page Fault Wait             0.00   0.00   0.00   0.00
  Local Evt Flg (Outswapped)  0.00   0.00   0.00   0.00
  Hibernate (Outswapped)      0.00   0.00   0.00   0.00
  Suspended (Outswapped)      0.00   0.00   0.00   0.00
```

```
     Free Page Wait              0.00   0.00   0.00   0.00
     Compute                     0.00   0.00   0.00   0.00
     Compute (Outswapped)        0.00   0.00   0.00   0.00
```

If any of the above fields show an integer in the "AVE" (Average) column, then the systems overall performance may be effecting the readers ability to process mailbox messages.

HELPFUL HINTS:

Certain mailboxs are typically associated with specific processes in OpenVMS.

```
  MBA1: = JOB_CONTROL
  MBA2: = OPCOM, AUDIT_SERVER
  MBA3: = AUDIT_SERVER, OPCOM
  MBA4: = CONFIGURE
```

Few problems have been reported with MBA1, 3, or 4.

- There are some known RWMBX issues with MBA2: as the suspect mailbox. These issues occur when the AUDIT SERVER has been enabled to monitor for many different conditions that produce security alarm. If many alarms are generated, the OPCOM process may become overloaded, causing it to back up as it processes the abundance of data and eventually fail.

  This issue was corrected in OpenVMS VAX v6.0. However, for systems running OpenVMS VAX versions prior to v6.0, increasing the WSQUOTA and PGFLQUOTA parameters for OPCOM should preclude it from failing. The following values are suggested.

  ```
             NEW_WSQUOTA = CUR_WSQUOTA x 4
                WSEXTENT = NEW_WSQUOTA x 2
               PGFLQUOTA = 40960
  ```

  Note:
  The increase in WSEXTENT is included due to the increase in WSQUOTA.

  Parameters for the OPCOM process are defined in its startup file, SYS$STARTUP:VMS$CONFIG-050_OPCOM.COM. This file should be modified with the new parameter settings. The following list equates the process parameters defined above to the definition of the parameters in the file.

  ```
                 WSQUOTA = /WORKING_SET
                WSEXTENT = /MAXIMUM_WORKING_SET
               PGFLQUOTA = /PAGE_FILE
  ```

  Once the file has been modified the OPCOM process should be restarted with the following command.

  ```
             $ @SYS$SYSTEM:STARTUP OPCOM
  ```

- Processes may hang in the RWMBX state when spawning subprocesses. When a process spawns a subprocess, SPAWN creates the subprocess and sends it information from the creating process through a system mailbox device (MBAx). The transfer of information through the mailbox can be the most time consuming part of the SPAWN operation. It is normal for the parent processes doing SPAWN operations to go into the RWMBX state. Typically this condition only lasts for short periods of time. This issue is addressed in another database article.

- If the SMISERVER process, on any node of a VMScluster hangs, the CUSTER_SERVER process enters a RWMBX state. When this occurs, other facilities that use the CLUSTER_SERVER process may not work correctly. For example, OPCOM messages won't be displayed and attempts to access the node via the SYSMAN utility will produce the SYSMAN-I-NODERR and SYSMAN-F-UNREACHABLE error messages. This issue is addressed in another database article.

- The analysis of a forced crash may show many "<LOGIN>" processes hung in RWMBX with a busy channel to a mailbox associated with the SECURITY_SERVER process. This issue is addressed in another database article.

- The following are some helpful programming tips when using mailboxes.

  1. Make sure the "reader" process is reading all the mailbox messages as soon as possible. Some "readers" check for mailbox messages on a "timed" interval. This is okay if the mailbox traffic remains low enough that the mailbox buffer never fills. If, however, the "writer" process becomes busy writing multiple messages, the "reader" may fall behind. If the "writer(s)" remains busy, the "reader" may not be able to read messages fast enough, and the mailbox fills up. The "writer(s)" would then go into RWMBX until the "reader" process reads another message.

     Some ways to try to keep the "reader" caught up are:

     o After processing one mailbox message, immediately request another read to make sure no messages were queued while the first message was being processed.

     o The "reader" process could always have an outstanding I/O request on the mailbox (without the qualifier IO$M_NOW).

     o The "reader" process could issue a SETMODE QIO request to ask for AST notification of unsolicited messages placed into the mailbox. When a message appears in the mailbox, the AST will be delivered, and the AST routine could read the mailbox and place the message into a work buffer.

  2. It is possible for the "writer" to get an error status back when the mailbox is full instead of going into the RWMBX state. The Set Resource Wait Mode system service, $SETRWM, can be used to tell OpenVMS you would rather receive an error indication. When such an error is returned, the "writer" could check the system for the "reader" process.

  3. If you know the number and sizes of the pending messages, you can estimate the size of the buffer area needed to hold the mailbox messages, and use this value as the BUFQUO parameter in the Create Mailbox system service, $CREMBX. Also, verify the processes doing the reading and writing have enough BIOLM and BYTLM to handle all the messages.