Resource Waits in the OpenVMS Operating System

or

What to do when you R-WASTed by OpenVMS

DECUS Spring ’93 Atlanta Symposia
VS060

David L. Cathey
Montagar Software Concepts
P. O. Box 260772
Plano, TX 75026-0772

(214) 618-2117
davidc@montagar.com

Session Outline

- What are "Resource Waits"
- Resource Waits and MUTEXes
- RWAST causes and descriptions
- Using SDA (System Dump Analyzer) to determine causes of RWAST
- Breaking processes out of RWAST
- Getting tough: Brute force approaches

What is a "Resource Wait"

- A Resource Wait is a type of MUTEX (MUTual EXclusion semaphore) within the OpenVMS Operating System.
- Resource Waits are a set of events that suspends a process until some process or operating system resource becomes available.
- Typically, the resource, or enough of a resource, becomes available and the process is resumed.
- MUTEXes and Resource Waits are not always "evil". They are used to provide a form of flow control, or allow a process to maximize throughput by utilizing all available quota.

Resource Waits and MUTEXes

- A MUTEX is a synchronization mechanism (similar to a Lock, i.e. $ENQ/$DEQ) that allows protecting an operating system resource without blocking all other activity by using elevated IPL.

- A MUTEX is implemented via a longword data cell in the OpenVMS executive. See "VMS Internals and Data Structures V5.2", Chapter 8.5, page 196, or "Alpha Internals and Data Structures", Chapter 9.7, page 9-50.

- Examples of MUTEXes used in OpenVMS V5.5 are:

— LNM$AL_MUTEX	Logical Name Table MUTEX
— IOC$GL_MUTEX	I/O Database
— EXE$GL_CEBMTX	Common Event Block Queue
— SMP$GL_CPU_MUTEX	CPU Database Queue
— EXE$GL_PGDYNMTX	Paged Dynamic Memory
— EXE$GL_GSDMTX	Global Section Descriptor Queue
— CIA$GL_MUTEX	Cumulative Intrusion Analysis Queue
— EXE$GL_BASIMGMTX	Base OpenVMS Image (Loaded Images)

Resource Waits and MUTEXes

- The MUTEX longword is manipulated via the internal routines SCH$LOCKR, SCH$LOCKW, and SCH$UNLOCK.
- The MUTEX longword is divided up into two fields: Status and Owner Count.

31	16	15 0
MBZ	1	Owner Count

Write-in-progress or
Write-pending status bit

- Other related structures:
  - PCB$W_MXTCNT — Count of MUTEXes owned by this process
  - PCB$L_EFWM — Address (0x8nnnnnnn) of pending MUTEX

Example of Accessing a MUTEX

```
; Note that this code must be in Kernel mode,
; in order to access the MUTEX data cell for R/W access.
;
; Grab the Intrusion Queue mutex, so we can
; scan it safely...

       moval  g^CIA$GL_MUTEX,r0
       jsb           g^SCH$LOCKW   ; Lock MUTEX

       movl   g^CIA$GQ_INTRUDER,r3 ; Get first intrusion blk
       moval  g^CIA$GQ_INTRUDER,r4 ; Get listhead address
1$:    cmpl   r3,r4                ; If r3 is listhead, bail
       beql   5$
       ...                         ; Do lots of neat stuff
       movl   CIA$L_FLINK(r3),r3    ; Get next intrusion blk
       brw    1$
5$:
       moval  g^CIA$GL_MUTEX,r0    ; Unlock MUTEX
       jsb           g^SCH$UNLOCK
```

Resource Waits and MUTEXes

- Resource Waits are seen when a process requests an operating system resource or a process resource (quota), and there is not enough. The process will be put into an MWAIT state, and the cause for the wait is placed in the Event Flag Wait Mask (IDSM, Chapter 12, page 283).

- A process can be set to disallow placement into a Resource Wait by setting the Resource Wait Mode using the $SETRWM system service.

  Note: This will not completely disable all Resource Waits. It will prevent most cases. More on this later!

- Processes in a resource wait are identified by "SHOW SYSTEM" with a state starting with "RWxxx".

Resource Waits and MUTEXes

- OpenVMS V5.x defines these and other Resource Waits (defined in $RSNDEF in the macro library SYS$LIBRARY:LIB.MLB):

	State	Reason Code	Value	Meaning
—	RWAST	RSN$_ASTWAIT	1	Wait for AST event
—	RWMBX	RSN$_MAILBOX	2	Mailbox I/O
—	RWNPG	RSN$_NPDYNMEM	3	Nonpaged Dynamic Memory
—	RWPAG	RSN$_PGDYNMEM	5	Paged Dynamic Memory
—	RWMPE	RSN$_MPLEMPTY	11	Waiting for Modified List to empty
—	RWMPB	RSN$_MPWBUSY	12	Modified Page Writer Busy (ReallyWantedMyProcessBack - Pat O.)
—	RWSCS	RSN$_SCS	13	System Communications Services
—	RWCAP	RSN$_CPUCAP	15	CPU Capability (Vectors, etc)
—	RWCSV	RSN$_CLUSRV	16	Cluster Server Process Busy

Resource Waits and MUTEXes

- A Resource Wait is entered by the OpenVMS Executive when an exhaustion of a resource is detected. The routine SCH$RWAIT is called with the RSN$_nnnnnnn symbol as input. The RSN$ code is placed in the PCB$L_EFWM and the process state is set to MWAIT.

- The bit corresponding to the RSN$ code is set in the system longword SCH$GL_RESMASK, i.e. RSN$_MPWBUSY = 12.

31	12	0
	1	

- When the OpenVMS Executive determines that a resource has been freed, (via the RSE routine) it check the SCH$GL_RESMASK to determine if any processes are waiting on the resource. If so, the MWAIT process queue is scanned to determine which processes should be rescheduled.

Example of Putting a Process in a Resource Wait

```
;
;       Put self in RWAST if there unable to allocate
;       the required non-paged pool...
;
;       Assume R4 hold value of current PCB
1$:
        movl    #GOOF$C_LENGTH,r1
        jsb     g^EXE$DEBIT_BYTCNT_ALO ; Allocate 1000 bytes
        blbs    r0,5$

        movl    #RSN$_NPDYNMEM,r0     ; Can't do it, wait until
        jsb            SCH$RWAIT      ; the system frees some and
        brb     1$                    ; then try it again...
5$:
        movl    r1,GOOF$W_SIZE(r2)    ; Play with our new buffer
        ...
```

RWAST Causes and Descriptions

- RWAST == Resource Wait for AST-related event

- RWAST generally occurs for the following reasons:

  — Process is waiting for the deletion of a sub process.
  — AST Limit has been exhausted (ASTLM)
  — Direct I/O Limit has been exhausted (DIOLM)
  — Buffered I/O Limit has been exhausted (BIOLM)
  — Process is waiting for outstanding I/O to complete.

- Quota exhaustion typically is transient, as the process is allowed to continue executing once an I/O or other event has completed.

- Waiting for a process deletion is also transient, unless that process is not computable (i.e. **it's** stuck in RWAST, too!).

Using SDA to Determine RWAST Cause

- Process state will be MWAIT or RW???
  - If the state is "MWAIT", the process is in a MUTEX state. Use "EVAL/ADDRESS on the address specified in the Event Flag Wait Mask to determine which MUTEX is locked.
  - If the state is "RW???", then the process is in a resource wait. The Event Flag Wait Mask will be set to the reason code, which should match the state shown by SDA.
- Use "SHOW PROCESS" and "SHOW PROCESS/CHANNEL" to see what it is waiting on.
  - A "0" (BYTLM/BYTCNT may be a small value) a quota mean exhausted quota.
  - A "busy" channel means outstanding I/O
  - Otherwise, check subprocess count for active subprocesses. Analyze them for Resource Wait problems.

Sample SDA Output: SHOW PROCESS/INDEX

```
SDA> SHOW PROCESS/INDEX=0CF

Process index: 000F Name: DAVIDC_1 Extended PID: 000000CF

Process status: 02040001 RES,PHDRES

PCB address               805659E0  JIB address                806D2F80
PHD address               808F9000  Swapfile disk address     00000000
Master internal PID       00020019  Subprocess count                 1
Internal PID              0003000F  Creator internal PID      00020019
Extended PID              000000CF  Creator extended PID      00000099
State                        RWAST  Termination mailbox           002F
Current priority                 6  AST's enabled                 KESU
Base priority                    4  AST's active                  NONE
UIC                [00002,000001]  AST's remaining                148
Mutex count                      0  Buffered I/O count/limit        0/40
Waiting EF cluster               0  Direct I/O count/limit         40/40
Starting wait time        1B001B1B  BUFIO byte count/limit      30800/30800
Event flag wait mask      00000001  # open files allowed left     147
Local EF cluster 0        E0000000  Timer entries allowed left     20
Local EF cluster 1        00000000  Active page table count         0
Global cluster 2 pointer  00000000  Process WS page count         161
Global cluster 3 pointer  00000000  Global WS page count           40
```

Zero remaining quota

Event Flag Mask == 1 == RSN$_ASTWAIT
if 8nnnnnnn, then it would indicate which MUTEX

Sample SDA Output: SHOW PROCESS/CHANNEL

```
 SDA> SHOW PROCESS/INDEX=0CF/CHANNEL

Process index: 000F Name: DAVIDC_1 Extended PID: 000000CF

         Process active channels
         -----------------------

Channel    Window            Status   Device/file accessed
-------    ------            ------   -------------------

 0010     00000000                    DUA0:
 0020     8071C470                    DUA0:[DAVIDC.RWAST]RWAST_BIO.EXE;4
 0030     00000000            Busy    MBA50:
 0040     00000000                    TWA3:
 0050     00000000                    TWA3:
```

Mailbox I/O incomplete, probably needs flushing

Breaking Processes Out of RWAST

- Incomplete I/O make up most of the "undeletable" processes.
- Different device types require different ways to break the I/O.
- Typical types of incomplete I/O cases that are seen:
  - Network devices (NETnn: and RTAn:)
  - Mailboxes (MBAnnn:)
  - Disks
  - Tapes
  - Printers (believe it or not...)

Network Devices

- Network devices can generally be freed by telling NCP to disconnect the link between the processes:

```
$ MCR NCP SHOW KNOW LINKS
$! kill the link that seems to be connected to the RWAST'd process
$ MCR NCP DISCONNECT LINK
```

- Example:

```
$ mcr ncp show know link

Known Link Volatile Summary as of 6-APR-1993 20:17:47

    Link    Node          PID       Process     Remote link   Remote user
    8193 1.42 (AVATAR)   21600033  REMACP             8445   DAVIDC

$ mcr ncp disconnect link 8193
```

Mailbox Devices

- Once the proper mailbox is discovered, use a utility such as COPY to dump data into, or out of, the mailbox:

- For example, if the mailbox name was MBA1284, one of the two following commands should clear the condition:

  $ COPY MBA1284: NLA0:

  or

  $ COPY LOGIN.COM MBA1284:

  It's probably a better practice to copy **from** before copying to...

Tape Devices

- Tape devices occasionally fall off-line, and are not handled correctly. If this happens, it can usually be fixed by:
  - — Reload the tape and place back on-line
  - — DISMOUNT/ABORT [tape_drive]
  - — Force a "pack acknowledge":

```
devnam: .ascid      /MUA0:/
chan:   .word       0

        .entry      packack,0
        $ASSIGN_S             chan=chan,-
                              devnam=devnam
        $QIOW_S               chan=chan,-
                              func=#IO$_PACKACK
        ret

        .end        packack
```

Disk Devices

- Disk devices fall off-line as well. Follow the same guidelines used for tape devices:
  - DISMOUNT/ABORT/OVER=CHECK [disk_name]
  - Toggle drive off/on-line
  - Also, the "pack acknowledge" routine can sometimes be used to recover the disk drive.

Line Printer Devices (believe it or not!)

- In really perverse cases, a line printer device (typically LP11's) may get a partial buffer out, but for some reason be unable to complete the current I/O. Of course, just simply:

  - Close printer doors
  - Clear jams
  - Add paper
  - Other printer stuff...

- Case history:

A printer got stuck at the same time the SYMBIONT had a lock on a RIGHTSLIST entry... and was stopped. The SYMBIONT was RWASTed, had a blocking lock on the RIGHTSLIST, that ended up locking up everyone on the system (600+ angry users) in LOGINOUT, DIR/OWNER, etc.

The solution? Close the door to the 15-year-old-washing-machine-sized LP27 printer :-(

Brute Force Approaches to Getting out of RWAST

- REBOOT THE SYSTEM!!!

- But really folks, "simply" disable the resource wait for the process and let it die off:

  — Get the Process Control Block (PCB) address of the target process.

  — Set the PCB$M_SSRWAIT bit in PCB$L_STS.

  — Clear the PCB$M_DELPEN bit in PCB$L_STS.

  — Issue another $DELPRC to the process...

For those that need the code example...

```
          .title    DISABLE_RW
;++ ;     DISABLE_RW          -- Disable Resource Wait of another process
; ; Author:         David L. Cathey
;                   Montagar Software Concepts
;                   P. O. Box 260772
;                   Plano, TX 75026-0772
;                   davidc@montagar.com
;
          .link               "SYS$SYSTEM:SYS.STB"/SE
          .library  /SYS$LIBRARY:LIB/
          $PCBDEF             ; Process Control Block definitions


asc_pid:  .ascid    "xxxxxxxx"                  ; Save space for PID
bin_pid:  .long     0
prompt:             .ascid    "Process ID: "    ; Prompt string

          .entry    Main,0


          pushaw    asc_pid
          pushaq    prompt
          pushaq    asc_pid
          calls     #3,g^LIB$GET_FOREIGN        ; Get PID from user
          blbc      r0,999$

          pushal    bin_pid
```

```
        pushaq   asc_pid                        ; Convert ascii hex to binary
        calls    #2,g^OTS$CVT_TZ_L
        blbc     r0,999$

        $CMKRNL_S routin=do_it                  ; Play with the process...
999$:   ret

        .entry   Do_It,^M<>

        movl     bin_pid,r0
        jsb      g^EXE$EPID_TO_PCB              ; Get PCB from EPID
        tstl     r0                             ; Did we???
        beql     99$                            ; Nope, bail out
        bisl2    #PCB$M_SSRWAIT,PCB$L_STS(r0)   ; Set SSRWAIT disable
        bicl2    #PCB$M_DELPEN,PCB$L_STS(r0)    ; Clear delete pending
        $DELPRC_S pidadr=bin_pid                ; And delete again.
        ret                                     ; Bye...
99$:    movl     #SS$_NONEXPR,r0                ; Non-existent process!
        ret

        .end     Main
```

Questions and Answers