Tracing Tools on OpenVMS

Quick Usage Guide

FLT Trace: 2
PCS Trace: 3
PRF Trace: 4
EXC Trace: 5
SPL Trace: 5
IO Trace: 6
LNM Trace: 7
LCK Trace: 7
MTX Trace: 8
TQE Trace: 8
System Configuration Details: 8

FLT Trace:

Based on the architecture, a processor requires its data or variables to reside at particular offsets in the system's memory. For example, a 32-bit processor requires a 4-byte integer to reside at a memory address that is evenly divisible by 4. This requirement is called "*memory alignment*." Thus, a 4-byte integer can be located at memory address 0x2000 or 0x2004, but not at 0x2001. This makes alignment the aspect of a data item that refers to its placement in memory.

The mixing of byte, word, longword, and quadword data types can lead to data that is not aligned on natural boundaries. When accessing these unaligned data we end up generating a fault. Normally, for each of the alignment faults, a fault handler is called for retrieving the data from the unaligned address, thereby using more CPU cycles.

Note: A fault normally occurs when the instruction cannot complete and the Program Counter is left pointing at the instruction, and we have to rectify this fault to resume execution

Thus, an alignment fault is a concern for performance, not for program correctness. When a customer observes that an alignment fault is causing a performance issue, the FLT trace can be collected from the customer site using these commands sequentially:

```
SDA> READ SYSDEF
SDA> READ/EXEC
SDA> FLT LOAD
SDA> FLT START TRACE
SDA> SPAWN WAIT 00:03:0.0 (wait for 3 minutes)
SDA> FLT STOP TRACE
SDA> SET OUTPUT/SING/NOHEAD FLT_TRACE.TXT
SDA> FLT SHOW TRACE
SDA> SET OUTPUT/SING/NOHEAD   FLT_TRACE_SUMMARY.TXT
SDA> FLT SHOW TRACE/SUMM
SDA> FLT UNLOAD
```

If FLT traces need to be collected for a particular process, the following commands should be used sequentially to get the FLT trace data collected:

```
SDA> READ SYSDEF
SDA> READ/EXEC
SDA> SET PROC/ID=<PID>----------> Process which is consuming CPU
SDA> SHOW PROC/IMAGE
SDA> FLT LOAD
SDA> FLT START TRACE/INDEX=<PID>
SDA> SPAWN WAIT 00:03:0.0 (wait for 3 minutes)
SDA> FLT STOP TRACE
SDA> SET OUTPUT/SING/NOHEAD FLT_SUMMARY.TXT
SDA> FLT SHOW TRACE/SUMM
SDA> SET OUTPUT/SING/NOHEAD FLT_DETAIL.TXT
SDA> FLT SHOW TRACE
SDA> FLT UNLOAD
```

The following command script can also be used to collect the FLT trace:

```
$!
$! FLT.COM
$!
$! This command procedure should be executed in a DCL terminal window.
$! It will run for 3 minutes and create two files: FLT_SUMMARY.TXT
$! and FLT_DETAIL.TXT. Please make these files available to OpenVMS
$! Engineering. If possible, please look at the top few faulting PC's
$! in FLT_SUMMARY.TXT and search the FLT_DETAIL.TXT file for these
$! PC's. Note their 8-digit EPIDs (pppppppp). Go back
$! into SDA and issue SHOW PROCESS /IMAGE=ALL /ID=pppppppp
$! This will tell you the process and image names for the top
$! faulting PC's.
$!
$ analyze /system
 flt load
 spawn write sys$output "Collecting data for 3 Minutes..."
 flt start trace
 spawn wait 00:03:0.0
 flt stop trace
 set output /single FLT_SUMMARY.TXT
 flt show trace /summary
 spawn write sys$output "Writing FLT_DETAIL.TXT. May take a few minutes..."
 set output /single FLT_DETAIL.TXT
 flt show trace
 flt unload
 exit
$
```

Please use the FLT command from SDA prompt for additional usage of FLT extension.

PCS Trace:

High CPU consumption by some process may be a performance issue for the customer. With the PCS trace data collected, we can find out where the process is consuming most of its CPU time. PC sampling helps to trace interrupted state time, as well as that of general process level code.

The following commands can be used sequentially to collect the PCS trace:

SDA> READ SYSDEF
SDA> READ/EXEC
SDA> PCS LOAD
SDA> PCS START TRACE
SDA> SPAWN WAIT 00:03:0.0 (wait for 3 minutes)
SDA> PCS STOP TRACE
SDA> SET OUTPUT/SING/NOHEAD PCS_SUMMARY.TXT
SDA> PCS SHOW TRACE/SUMM
SDA> SET OUTPUT/SING/NOHEAD PCS_TRACE.TXT
SDA> PCS SHOW TRACE
SDA> SET OUTPUT/SING/NOHEAD PCS_STATISTIC.TXT
SDA> PCS SHOW TRACE/ STATISTIC
SDA> PCS UNLOAD

If PCS traces need to be collected for a particular process, the following commands should be used sequentially to get the PCS trace data collected:

PRF Trace:

The PRF utility has the ability to do PC sampling. With the use of this tool it is easy to find the hot spots where most CPU cycles are spent. The PC along with module or offset and symbolization information is displayed as well as the count of how many times it has been recorded. This percentage will give you an idea on how this count translates to the total number of collected PC samples.

The following command script can be used to collect the PRF trace for a specific process:

```
$!
$! PRF.COM
$!
$ set noverify
$! Determine output file name
$!
$ nodename = f$getsyi("nodename")
$ systime = f$edit(''f$time()', "TRIM")
$ date = f$element(0, " ", systime)
$ time = f$element(1, " ", systime)
$ day = f$element(0, "-", date)
$ mon = f$element(1, "-", date)
$ year = f$element(2, "-", date)
$ hour = f$element(0, ":", time)
$ min = f$element(1, ":", time)
$! Get the PID of the process for invoking start miss event profiling $
$ Open/Read Stdin Sys$Command:
$ Read/Error=Get_Out -
/Prompt="Please provide the Process ID for profiling : " Stdin PID_string
$ PID = F$Integer(PID_string)
$$
$! write sys$output PID
$ if f$length(day) .eq. 1 then day = "0"+day $ filename =
nodename+"_SPL_"+day+mon+year+"_"+hour+min+".TXT"
$!
$ prfcom = ''(filename-"TXT"+"COM")'
$ open/write/error=OpenError_ outcom 'PRFCOM'
$ write outcom "$!"
$ write outcom "$! This command procedure is written by another command"
$ write outcom "$! procedure so that it may adapt to the environment"
$ write outcom "$! where it is used. Therefore it should not be"
6
$ write outcom "$! modified directly."
$ write outcom "$!"
$ write outcom "$ write sys$output ""Collecting PRF Trace Data"""
$ write outcom "$ write sys$output """"
$ write outcom "$ anal/sys"
$ write outcom "prf load/OVERRIDE"
$ write outcom "SET OUTPUT ''filename'"
$ write outcom "PRF START PROFILE/CACHE=L2/INDEX=""''PID'"""
$ write outcom "PRF START COLLECT"
$ write outcom "wait 00:00:15.0"
$ write outcom "PRF STOP COLLECT"
$ write outcom "PRF STOP PROFILE"
$ write outcom "PRF SHOW COLLECT/NOINSTR/THRESH=.5"
$ write outcom "SET OUTPUT SYS$OUTPUT"
$ write outcom "PRF UNLOAD"
$ write outcom "exit"
```

```
$ write outcom "$ write sys$output ""Created ''filename'"""
$ write outcom "$ exit"
$ close outcom
$ @'PRFCOM'
$ delete/nolog 'PRFCOM';
$ exit
$ Get_Out:
$ Close Stdin
$OpenError_:
$ write sys$output "Unable to create ''PRFCOM'."
$ exit
```

Please use the PRF command from SDA prompt for additional usage of PRF extension.

EXC Trace:

Exception handling on IA64 may be a performance issue for various customers. Finding and solving it is not as easy task. In a scenario like this, the EXC trace can be used to find where exceptions occur within an application.

The following commands can be used sequentially to collect the EXC trace:

SDA> EXC LOAD
SDA> EXC START TRACE
SDA> SPAWN WAIT 00:00:20.0 (Wait for 20 seconds)
SDA> EXC STOP TRACE
SDA> SET OUTPUT/SING/NOHEAD EXC_TRACE.TXT
SDA> EXC SHOW TRACE
SDA> EXC UNLOAD

Please use the EXC command from SDA prompt for additional usage of EXC extension.

SPL Trace:

SPL trace is used to collect VMS Kernel spinlock data. Sometimes there may be a performance issue that the customer may observe with various spinlocks used by VMS Kernel. The SPL trace is useful to find out which spinlocks are taken and who is responsible for holding the spinlock.

The following command script can be used to collect the SPL trace:

@SYS$EXAMPLES: SPL.COM

Please use the SPL command from SDA prompt for additional usage of SPL extension.

SPL tracing for a longer time interval can also be triggered from the T4 data collection. To start SPL trace inside the T4 collection, the user needs to define a logical name T4$EXPERT to 1.

$ DEFINE/SYSTEM/EXEC T4$EXPERT 1

Sample T4 Collection triggering SPL Tracing:

```
$ @T4$SYS:T4$CONFIG
Copyright 2000-2008 Hewlett-Packard Development Company, L.P.
T4 Version v4.3
Executing T4$CONFIG.COM on node FPAR1 - Date/Time is now 17-SEP-2010 01:09:53.30
T4$SYS defined as FPAR1$DKA0:[SYS0.SYSCOMMON.T4$SYS]
T4$DATA logical name has NOT been defined
Please see T4$SYS:T4_README.TXT if you need assistance configuring or running T4.
Collection Start Time [17-SEP-2010 08:00:00.00] : NOW
Collection End Time [17-SEP-2010 23:59:00.00] : +00:10:00
Batch queue name [SYS$BATCH] : T4$BATCH
Network Interface Device (? for list, type RETURN to finish) :
Sampling Interval (seconds) [60] : 1
Default Data directory can be defined with the system logical T4$DATA
Destination Directory [] : FPAR1$DKA0:[T4$DATA]
Destination Directory is on the SYSTEM DISK,
Write data to system disk ? [N] : Y
Collect SPL traces at regular intervals? [N] : Y
SPL trace intervals (seconds) [600] :
SPL trace buffer size [500] :
Time to fill SPL trace buffer (<59 seconds) [1] : 30
Checking for EVA controllers - please wait ...
Re-Submit data collection job daily [N] :
Email address :
Job T4$COLLECT (queue T4$BATCH, entry 1) started on T4$BATCH
```

Please refer to T4 supplied readme.txt file for more details.

IO Trace:

When it has been determined that buffered or direct IO rates are high (due to MONITOR or other utilities), you may want to get more information on the nature of the IOs. IO tracing helps to find out what is the reason behind it.

The following commands can be used sequentially to collect the IO trace. Options given in square bracket are optional. Please use as per requirements.

```
SDA> IO LOAD
SDA> IO START TRACE [/BUFFER=pages] [/[NO]BIO] [/[NO]DIO] [/XQP]
SDA> IO STOP TRACE
SDA> IO SHOW TRACE [/BIO] [/DIO] [/ALL]
SDA> IO START COLLECT [/BIO] [/DIO] [/DEVICE][/PROCESS]
SDA> IO STOP COLLECT
SDA> IO SHOW COLLECT [/FULL] [/TOP=n] [/RATES] [/TOTALS]
SDA> IO UNLOAD
```

Please use the IO command from SDA prompt for additional usage of IO extension.

LNM Trace:

This tracing can be used to keep track of logical name translation rates.

The following commands can be used sequentially to collect the LNM trace. Options given in square bracket are optional. Please use as per requirements.

```
SDA> LNM LOAD
SDA> LNM START TRACE [/BUFFER=pages]
SDA> LNM STOP TRACE
SDA> LNM SHOW TRACE [/IDENTIFICATION=epid] [/LOGICAL=logical[*]]
SDA> LNM START COLLECT /LOGICAL [/PROCESS]
SDA> LNM STOP COLLECT
SDA> SET OUTPUT/SING/NOHEAD LNM_COLLECT.TXT
SDA> LNM SHOW COLLECT [/RATES or /TOTALS]
SDA> LNM UNLOAD
```

Please use the LNM command from SDA prompt for additional usage of LNM extension.

LCK Trace:

When there is an unusual activity related to LOCK ENQ and when DEQ rate is observed, the LCK trace can be used to find out lock manager statistics as well as various lock activities.

The following commands can be used sequentially to collect the LCK trace. Options given in square bracket are optional. Please use as per requirements.

Dedicated lock manager usage:

```
SDA> LCK LOAD
SDA> LCK START COLLECT
SDA> LCK SHOW LCKMGR /INTERVAL=10 /REP=10
SDA> LCK STOP COLLECT
SDA> LCK UNLOAD
```

Per-Process lock activity example:

```
SDA> LCK LOAD
SDA> LCK START TRACE
SDA> LCK START COLLECT /PROCESS [/STATISTICS]
SDA> LCK STOP COLLECT
SDA> SET OUTPUT/SING/NOHEAD LCK_PROCESS.TXT
SDA> LCK SHOW COLLECT
SDA> LCK STOP TRACE
SDA> LCK UNLOAD
```

Please use the LCK command from SDA prompt for additional usage of LCK extension.

MTX Trace:

Mutex tracing allows you to track mutex locks and unlocks.

The following commands can be used sequentially to collect MTX traces. Options given in square bracket are optional. Please use as per requirements.

SDA> MTX LOAD
SDA> MTX START TRACE [/BUFFER=pages] [/CPU=n] [/MUTEX=mutex]
SDA> MTX STOP TRACE
SDA> MTX SHOW TRACE [/MUTEX=(mutex[,...])] [/SUMMARY] [/CPU=n] [/TOP=n]
SDA> MTX UNLOAD

Please use the MTX command from SDA prompt for additional usage of MTX extension.

TQE Trace:

MONITOR TIMER identifies the TQE expiration rate on the system. If these numbers get high, you may want to investigate whose timers are expiring. TQE traces generate summary listings of all the process and subroutine causing this timer expire.

The following commands can be used sequentially to collect the TQE trace. Options given in square bracket are optional. Please use as per requirements.

SDA> TQE LOAD
SDA> TQE START TRACE [/BUFFER=pages]
SDA> TQE STOP TRACE
SDA> TQE SHOW TRACE /SUMMARY
[/RATES] (default)
[/TOTALS]
[/IDENTIFICATION=pid]
[/ADDRESS=address]
SDA> TQE UNLOAD

Please use the TQE command from SDA prompt for additional usage of TQE extension.

System Configuration Details:

Please use the following script for getting system configuration details before reporting any performance related issue to OpenVMS Engineering.

```
$!
$! SYSTEM_CONFIGURATION_DETAILS.COM
$!
$set noverify
$set term/inq/wid=132
$write sys$output "***********************************************"
$write sys$output " RAD Configuration(RAD.COM) "
$write sys$output "***********************************************"
$@sys$examples:rad
$write sys$output "***********************************************"
$write sys$output " RAD Configuration(SHOW RAD command) "
$write sys$output "***********************************************"
$anal/sys
show rad
```

```
$write sys$output "************************************************"
$write sys$output " CPU Configuration "
$write sys$output "************************************************"
$show cpu/all
$write sys$output "************************************************"
$write sys$output " Full CPU Configuration "
$write sys$output "************************************************"
$ SHOW FAST
$write sys$output "************************************************"
$write sys$output " SHOW SUMMARY "
$write sys$output "************************************************"
$ anal/syst
show sum
$write sys$output "************************************************"
$write sys$output " SYSTEM PARAMETERS "
$write sys$output "************************************************"
$ mc sysgen show/all
$write sys$output "************************************************"
$write sys$output " ACCOUNT INFORMATION "
$write sys$output "************************************************"
$ mc authorize show system$show cpu/full
$write sys$output "************************************************"
$write sys$output " SDA CLUE CONFIG output "
$write sys$output "************************************************"
$ anal/syst
clue config
$write sys$output "************************************************"
$write sys$output " DG Device Configuration "
$write sys$output "************************************************"
$show dev dg /full
$write sys$output "************************************************"
$write sys$output " System Memory Configuration "
$write sys$output "************************************************"
$ SHOW MEMORY
$write sys$output "************************************************"
$write sys$output " Fastpath Configuration "
$write sys$output "************************************************"
```