Making the most out of a system crash

OpenVMS European Technical Update Days, October 2006

Richard Bishop

Who cares, anyway?

- In a perfect world, there would be no system dumps!
- For many users, OpenVMS is perfect
- But the reality is that occasionally system crashes happen
- Partly because of their infrequency, mistakes are made and dumps are lost or useless
- Let's not waste a crash and have to recreate it

Topics

- Some definitions
- Setting up the dump file
  - Size
  - Location
- Setting relevant SYSGEN parameters
- Identifying processes that must be dumped
- Common mistakes
- Recent improvements

Some definitions

- Types of dumps
  - Full dump: all memory is dumped from lowest to highest physical address
  - Selective dump: only pages in use are dumped…
    - First, system space
    - Then each process in turn
    - Then global pages
  - Raw dump: each page is dumped “as is” (using 16 disk blocks for every page)
  - Compressed dump: each page is compressed before dumping (using as little as one byte for a page)
  - Full vs. Selective is orthogonal to Raw vs. Compressed
  - Default dump type is Compressed Selective
    - This is the design center for any future enhancements

Some definitions

- Key processes
  - In a selective dump, certain processes are dumped first, in the following order:
    - Current process on crash CPU
    - Swapper
    - Current processes on any CPUs that did not BUGCHECK
    - Current processes on remaining CPUs
    - Customer-defined priority processes
    - HP-defined priority processes (ACPs, some server processes, etc.)
    - Any processes in an MWAIT state (RWAST etc.)
  - These are called key processes
  - All global pages mapped to key processes are dumped next
  - Then all other processes
  - Finally all global pages mapped to these other processes
  - Only interesting if process exists and dump is too small

Setting up the dump file: size

- Start with AUTOGEN
- Warning: turn off file high-water marking first
  - Otherwise AUTOGEN will only create a small dump file
  - You must then extend it after the system reboots
- Fine-tune the size from there
  - Is there enough disk space?
  - Do you always get (at least) key processes and key global pages dumped?
  - Are you concerned by how long it takes to write the dump?
  - Trial and error from there on

Setting up the dump file: size

- Do a test run with a overly large dump file and a representative load
- Use SHOW DUMP to determine space used
  - For complete dump
  - Up to and including key global pages
- Decide if key processes & global pages is enough
- Add 10%-25% slop
- Big memory => big PFN database => big dump even if nothing happening on system
  - Just for the PFN database, a 64GB system uses
    ~220,000 blocks (Alpha)
    ~350,000 blocks (Integrity)
  - And that is taking compression into account!

Setting up the dump file: location

- By default: SYS$SYSTEM:SYSDUMP.DMP
  - If there's room, this is fine
  - If a cluster common system disk, may not be space
  - On Integrity, cannot use system disk if a satellite
- Dump Off System Disk
  - Disk:[SYSn.SYSEXE]SYSDUMP.DMP
  - Can be any directly connected drive (SCSI, Fibre, CI)
  - Cannot be MSCP-served to the system
  - If satellite, must be locally connected
- SCSI vs. Fibre: Fibre is usually faster, especially on Alpha
- DUMP_DEV environment variable
  - May need to use this even if only dumping to system disk
  - More on DUMP_DEV later

Setting relevant SYSGEN parameters

- DUMPSTYLE

Bit	Name	= 0	= 1
0	Full vs. Selective	Full	Selective
1	Console output	Minimum	Verbose
2	Dump location	System disk	Dump disk
3	Raw vs. Compressed	Raw	Compressed
4	Galaxy shared memory	Dump it	Do not dump
5*	Key vs. All	All processes	Key processes
6-31	Reserved to HP	Yes	Only if asked

* New in V8.3

Setting relevant SYSGEN parameters

- DUMPBUG
- BUGREBOOT
- SYSTEM_CHECK
  - MULTIPROCESSING
  - POOLCHECK
  - BUGCHECKFATAL
  - MON images => other sanity checks
- SAVEDUMP

DUMP_DEV

- Identifies path(s) to system disk and/or dump disk
- Alpha:

```
>>>SET DUMP_DEV dga2000.1002.0.3.0,dga2000.1003.0.3.0
```

  - Limit on combined length is 256 bytes
  - Allows for maximum of 8-9 paths (device-dependent)
  - Therefore cannot always specify all possible paths
- Integrity:

```
$ @sys$manager:boot_options
```

  - Choose option D, then option 1 for each drive to be added
  - All paths to each drive will be found
  - Up to 99 paths
  - Therefore allows all possible paths to be specified

DUMP_DEV

- Use DUMP_DEV to identify dump disk and for multiple paths or members of system disk
- If DUMP_DEV not set, only drive & path used at boot time is accessible to BUGCHECK
- Specify all paths and/or members
- Allows error log buffers to be saved and restored
- If both dump disk and system disk, include dump disk paths first
- Dump disk + 3-member system disk, all with 4 paths: 16 possible paths

DUMP_DEV

- DUMP_DEV only relevant for SYSDUMP.DMP
- Except:
  - Shadowed system disks
    - DUMP_DEV used to find all members
  - Integrity satellites
    - SYS$ERRLOG.DMP must be on a local disk (in [SYSn.SYSEXE])
    - DUMP_DEV must include this disk
- Dump device should be mounted
  - Use SYCONFIG.COM
  - AUTOGEN can create/resize it
  - Required for CLUE processing
    - CLUE$DOSD_DEVICE logical name also required
  - On Integrity satellites, disk must be mounted for error log buffer recovery by ERRFMT

Identifying processes that must be dumped

- Dump contains system space, key processes, key global pages, other processes, remaining global pages
- If dump file too small, some processes will not be dumped
- You can control which processes get dumped!!
- SYSMAN>DUMP_PRIORITY ADD etc.
  - Or edit a data file in older releases
- My conjecture:
  - 90% of crashes solved with just current process
  - 98% of crashes solved with key processes
  - 99% of crashes solved with complete selective dump

Identifying processes that must be dumped

- SYSMAN DUMP_PRIORITY commands
  - ADD – Adds a new process to the data file
  - LIST – Lists contents of the data file
  - LOAD – Loads data file into memory for BUGCHECK
  - MODIFY – Modifies a process entry in the data file
  - REMOVE – Removes a process from the data file
  - SHOW – Displays in-memory list
  - UNLOAD – Deletes the in-memory list
- Usual sequence

```
SYSMAN> DUMP ADD OPCOM /UIC=[SYSTEM]
SYSMAN> DUMP ADD FAL_* /UIC=[*] /WILD_CARD
SYSMAN> DUMP LOAD
```

Identifying processes that must be dumped

- Process name
  - May need to be quoted
  - * and % legal characters so need to say if they mean wildcards
- /UIC
  - [name] or [*]
  - [gnum,mnum] or [gnum,*]
  - [gname,*]
  - Not [gname*,*] or [gnum*,*]
- /WILD_CARD
  - Identifies * and % in process name as wildcard characters
- /[NO]INFORMATIONAL
  - Use /NOINFO to make “silent” changes during product installation

Identifying processes that must be dumped

- Reminder: list doesn't limit what is dumped
  - No effect if a named process not in system
    - BUGCHECK looks for a match but doesn't care if no match found
  - If sufficient space, only effect is ordering within dump
    - All processes will get dumped eventually if room
  - Unless KEY_ONLY bit set in DUMPSTYLE
- HP-defined priority processes:

AUDIT_SERVER	LES$ACP
NETACP	SHADOW_SERVER
NET$ACP	TCPIP$* (ACPs)
REMACP	

Common mistake: when moving SYSDUMP.DMP

- Do not use BACKUP to move SYSDUMP.DMP
  - It is marked /NOBACKUP
  - BACKUP only allocates and sets the file high-water mark to zero without copying any data
  - BUGCHECK writes the dump (it doesn't know about HWM) – so all looks well
  - But: SDA (& DUMP) gets zeroes back whenever the file is read – leads to the name "phantom dump"
  - %SDA-E-BADHWM error starting in V8.2
  - Can be fixed by:

```
$ SET VOLUME/NOHIGHWATER ddcn:  ! If necessary
$ SET FILE/END ddcn:[SYSn.SYSEXE]SYSDUMP.DMP
$ SET VOLUME/HIGHWATER ddcn:    ! If necessary
```

Common mistake: when saving a dump

- Do not use DCL COPY to save contents of a system dump (or BACKUP /IGNORE=NOBACKUP)
- Multiple reasons to use SDA COPY
  - BUGCHECK probably didn't use the entire file
    - SDA COPY only saves used blocks
  - Integrity system dumps need process unwind data
    - SDA COPY collects it and appends it to the copy
  - File ID to filename translation data may be useful
    - SDA COPY collects it and appends it to the copy
  - SDA COPY will compress the dump if originally written as a raw dump
  - SDA will read blocks from the master member if system disk is shadowed

Common mistake: when saving a dump

- Recommended method
  - Create an SDA procedure that includes the COPY command
  - Ensure the COPY command is the last command
  - Define logical name CLUE$SITE_PROC (/SYSTEM)
  - If using DOSD, ensure disk is mounted and logical name CLUE$DOSD_DEVICE defined
- Possible difficulty
  - If data disks not mounted, collection may not work
  - May prefer to delay copy, but will lose dump with another crash
  - Do separate collection with COLLECT /SAVE

Common mistake: when analyzing a dump on a shadowed system disk

- Be careful when analyzing a system dump written to a shadowed system disk
  - BUGCHECK knows little of shadowing
  - Writes dump to master member of shadow set
  - Prior to V7.3-1 had to rely on merge/copy of shadow set
  - SDA now reads only from master member
  - Use ANALYZE/CRASH/SHADOW_MEMBER=ddcn to override if correct member known
  - Or ANALYZE/CRASH/SHADOW_MEMBER to display choices
    - Gives BUGCHECK code, date/time, SCSNODE, & version for each shadow set member
    - Enter: USE ddcn to choose member

Common mistake: when setting up a system disk shadow set

- Do not use system disk shadow set members with same unit number
  - For example: DSAn: with members DKA0:, DKB0:
  - BUGCHECK cannot distinguish separate disks from multiple paths to same disk
  - Memory dump may be written to wrong member
  - Error logs will not be written to all members
    - Could lead to corruption of ERRLOG.SYS if master member has changed
  - Considering possible future change to BUGCHECK to handle this
    - Would require dump header to be read back and compared

Common mistake: analyzing a dump from elsewhere in a cluster

- Be careful when analyzing SYSDUMP.DMP from another cluster member
  - Session left at SDA> prompt
  - System that crashed crashes again
  - Exit from SDA and reissue ANALYZE command
  - Unable to analyze dump
  - Dump header gets written back when you exit SDA
  - You end up with dump header for first dump & rest of dump from second dump
  - 99.99% fixed in V7.3-1: SDA re-reads the dump header and compares it immediately before updating it and writing it back

Common mistake: ignoring mismatch warnings

- Ignoring LINKTIMEMISM and/or SDALINKMISM messages
- Assume these mean trouble!
- SDA will appear to work but oddities may show up later – often very subtle
- Only time the messages can be ignored is if SDA$SHARE.EXE or SYS$BASE_IMAGE.EXE have shipped in a patch kit and are out of step
- From V8.3 onwards the crash banner will include the version:

```
Dump taken on 28-APR-2006 15:09:33.83 using version XB8V-N2O
```

Common mistake: when using PAGEFILE.SYS for dumps

- Dates back to small system disks and no DOSD
- Not recommended but is supported
- SAVEDUMP must be set
- SDA COPY will release pagefile blocks on completion
  - SDA will immediately exit
  - Hence COPY should be last command in procedure CLUE$SITE_PROC
- Ensure that the dump is copied or space released
  - Otherwise you can quickly run out of pagefile space

Recent improvements

- Items already mentioned
  - Size of key portion of dump now displayed by SHOW DUMP (V8.3)
  - SYSMAN DUMP_PRIORITY commands (V7.3-2)
  - %SDA-E-BADHWM when file high-water mark prevents access to the dump header (V8.2)
  - Collection of unwind and file ID translation data (V8.2-1 & V8.3)
  - Dump accessed from master member of shadow set (V7.3-1)
  - SDA avoids overwriting changed dump header (V7.3-1)
  - Crash banner includes version from dump (V8.3)

Recent improvements

- Performance on Integrity
  - Physically contiguous pages written as a single I/O
    - On Alpha & early Integrity versions, each page requires a separate I/O
  - Double buffering technique used
    - On Alpha & early Integrity versions, each I/O must be completed before compression begins for next I/O
  - Up to 40% speed improvement
  - Possible because more work done in OpenVMS
  - Alpha? Unlikely:
    - Contiguous pages change investigated but found to be too risky
    - Double buffering change would need a new interface between OpenVMS and console

Audience Participation Time!

- Any questions?
- Any enhancement requests in SDA?
- Any more “common mistakes” you think I’ve missed?
- Anything else?


invent