How to configure dump devices

In order to understand the following text you should be familiar with the basic concept of the Logical Volume Manager LVM. I make use of the these abbreviations:

VG = Volume Group
LV = Logical Volume
PV = Physical Volume

Choosing dump devices

Dump devices are volumes on the disk that are used to hold the entire memory image when the system crashes. The cumulative size of all specified dump devices has to be some MB larger than the amount of memory in order to hold the entire core. To determine the current size of physical memory:

```
# dmesg | grep Physical
Physical:524288 KB ,lockable:386672 KB ,available:454144 KB
```

As of UX 11.00 you can use `crashconf(1M)`:

```
# crashconf | grep Total
Total pages on system:            131072
Total pages included in dump:      30832
```

(A page is always 4KB)

NOTE: Increasing the amount of dumpspace is an important thing to do when adding more physical memory to the system.

Formerly the maximum size of a dump device was 2GB or more precise: the dump LV had to be placed within the first 2GB of the PV whereas newer systems support dump devices up to 4GB or since UX 11.00 even greater than 4GB.
It's important to mention that it's the Interface Card, not the disk, that defines whether the disk can be used for more than 4GB of dump. Cards in the systems like L-, N-, V-Class and newer all support this. Details can be found in KMINE document S3100004913.

A swap device can also be used as dump device in order to save disk space but there are two disadvantages:

1) Is the **primary** swap device (usually /dev/vg00/lvol2) also configured as dump device, it takes more time for the system to bootup after a systemcrash.
   Reason: When a dump is found on the dump device during startup it will be written to the local filesystem (by the rc command savecrash). In the case that the dump device is also the primary swap, savecrash cannot run in the background because the swap area may be used during further startup.

2) Were there any problems with savecrash (lack of space in the crash directory) you still

have the possibility to run it again after the system boot completed (-r Option for resave dump). In case of a swap device there is a risk that parts of the dump are overwritten by "swapping" activities and therefor unusable.

You can influence the interaction of savecrash/core and swapon in the config file of savecrash/core. (see manpage of savecrash/core -w option)

Configuration steps

Creating the logical volumes that should be used for dump

You can specify up to 32 different dump devices. Each dump logical volume has to be ***contiguous***, i.e. all physical extents are placed one after another and reside on a single PV. Such a LV can be created with the option `-C y` of `lvcreate` command. ***Bad block relocation*** must be disabled (`–r n`):

```
# lvcreate -L <size in MB> -n lvdump -C y –r n /dev/vg00
```

You can check the LV parameters with lvdisplay:

```
# lvdisplay /dev/vg00/lvdump | grep Allocation
Allocation                 strict/contiguous

# lvdisplay /dev/vg00/lvdump | grep Bad
Bad block                  off
```

The dump LVs must not contain a filesystem of course.

Activating these logical volumes, i.e. tell the system to use them for dump

A *traditional dump LV* has to be located in the root VG (vg00) and the `lvlnboot` command is used to tell the system to uses these LVs for dump. A reboot is neccessary in order to activate them. Here's how to configure such a dump device:

Display the current settings:

```
# lvlnboot -v
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
  /dev/dsk/c0t6d0 (10/0.6.0) -- Boot Disk
  /dev/dsk/c0t5d0 (10/0.5.0)
Root: lvol1  on:   /dev/dsk/c0t6d0
Swap: lvol2  on:   /dev/dsk/c0t6d0
No Dump Logical Volume configured
```

Option `-d` sets the dump device:

```
# lvlnboot -d lvol2 /dev/vg00
# lvlnboot -d lvdump /dev/vg00
```

Check it:

```
# lvlnboot -v | grep dump
Dump: lvol2     on:     /dev/dsk/c0t6d0, 0
```

```
Dump: lvdump    on:     /dev/dsk/c0t6d0, 1
```

If the dump devices are configured according your needs you have to reboot in order to make the changes take effect. The message buffer displays all valid dumpdevices during reboot:

```
# dmesg | grep DUMP
Logical volume 64, 0x2 configured as DUMP
Logical volume 64, 0x9 configured as DUMP
```

If you like to use a dump device for other purposes you have to deconfigure it using `lvrmboot`. Only the last dump device can be deconfigured:

```
# lvrmboot -d lvdump /dev/vg00
```

NOTE: An entry in the kernel (`/stand/vmunix`) is necessary if you like to have more than one (traditional) dump device with LVM. This entry is set by default:

```
# strings /stand/vmunix | grep "dump lvol"
dump lvol
```

As of UX 11.00 you have the possiblility to configure additional dump devices online, i.e. without the need of a reboot. These dump LVs must not be configured using `lvlnboot –d` but with `crashconf(1M)`. You are no longer restricted to choose a dump LV from the root VG only. The configuration of such dump devices is similar to the configuration of secondary swap devices. Here's how to configure a dump device online:

Add a line for each dump device to `/etc/fstab`, e.g.:

```
/dev/vg01/lvdump  /  dump  defaults  0 0
```

Then run `crashconf -a` to activate it and `crashconf` to verify that it is enabled. Configuring non-root dump devices is similar to configuring secondary swap devices.

Refer to the `crashconf(1m)` and `fstab` manual pages for details.

NOTE: Whenever you have dump devices that are not also used for swap activity, make sure that they are configured last. This will cause them to be used first (dump goes from the end backward), which will minimize the chance of writing into an area shared by swap. Writing into swap space is undesirable because it will slow down your reboot processing (see section above).

NOTE: There are often questions like: "Why is the dump LV not mirrored like root, boot and swap LVs are?"

```
# lvlnboot -v
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
        /dev/dsk/c0t6d0 (10/0.6.0) -- Boot Disk
        /dev/dsk/c0t5d0 (10/0.5.0) -- Boot Disk
Root: lvol1     on:     /dev/dsk/c0t6d0
                        /dev/dsk/c0t5d0
Swap: lvol2     on:     /dev/dsk/c0t6d0
```

```
                        /dev/dsk/c0t5d0
Dump: lvol2     on:     /dev/dsk/c0t6d0, 0
Dump: lvdump    on:     /dev/dsk/c0t6d0, 1
```

The answer: the system dumps onto a previously configured area of the disk. The dump process is a low level routine that bypasses the LVM layer, hence the data is not going to be mirrored. The OS simply stored the hardware path of the disk and the starting and ending offset on this disk at the time you activated it. This information is given by the dump LV. This is the reason why dump LVs must be contiguous.

The dump/savecrash process

Writing the memory image to the dump devices

The kernel routine responsible for dumping is dumpsys().

Dump formats

There are four known dump formats. Which format you deal with can be found in the INDEX file (`grep version INDEX`):

COREFILE (Version 0)
This format, used up through HP-UX 10.01, consists of a single file containing the physical memory image, with a 1-to-1 correspondence between file offset and memory address. Normally there is an associated file containing the kernel image. Sources or destinations of this type must be specified as two pathnames to plain files, separated by whitespace; the first is the core image file and the second is the kernel image file.

COREDIR (Version 1)
This format, used in HP-UX 10.10, 10.20, and 10.30, consists of a `core.n` directory containing an INDEX file, the kernel (vmunix) file, and numerous `core.n.m` files, which contain portions of the physical memory image.

CRASHDIR (Version 2)
This format, used in HP-UX 11.00, consists of a crash.n directory containing an `INDEX` file, the kernel and all dynamically loaded kernel module files, and numerous `image.X.Y` files, each of which contain portions of the physical memory image and metadata describing which memory pages were dumped and which were not.

PARDIR (Version 5)
This format is used in UX 11.11 and later. It is very similar in structure to the CRASHDIR format in that it consists of a `crash.n` directory containing an `INDEX` file, the kernel and all dynamically loaded kernel module files, and numerous `image.X.Y` files, each of which contain portions of the physical memory image and metadata describing which memory pages were dumped and which were not. In addition to the primary `INDEX` file, there are auxiliary index files (`indexX.Y`), that contain metadata describing the image files containing the memory pages. This format will be used when the dump is compressed. See crashconf(1M).

Other formats, for example tape archival formats, may be added in the future.

Selective dumps

The most significant change compared to UX 10.X is the possibility of configuring **selective dumps**. Dumps no longer contain the entire contents of physical memory. With memory sizes growing in leaps and bounds, it become critical that HP-UX dump only those parts of physical memory which are considered useful in debugging a problem. By default you get a core of approx. 5-40% of physical memory, variing with the state of the system at dumptime. Configuration can be checked and modified with the crashconf utility:

```
# crashconf

CLASS          PAGES  INCLUDED IN DUMP  DESCRIPTION
--------  ----------  ----------------  ----------------------------
UNUSED          14253  no,  by default   unused pages
USERPG          23876  no,  by default   user process pages
BCACHE         129981  no,  by default   buffer cache pages
KCODE            2044  no,  by default   kernel code pages
USTACK            451  yes, by default   user process stacks
FSDATA            753  yes, by default   file system metadata
KDDATA          72447  yes, by default   kernel dynamic data
KSDATA          17699  yes, by default   kernel static data

Total pages on system:            261504
Total pages included in dump:      91350

DEVICE        OFFSET(kB)   SIZE (kB)  LOGICAL VOL.  NAME
------------  ----------  ----------  ------------  -----------------
 31:0x006000       72544      524288   64:0x000002  /dev/vg00/lvol2
                          ----------
                              524288
```

Compressed dumps

Even with selective dump feature a Superdome equipped with 256GB RAM would take hours to write the dump to the dump devices. The bottleneck of copying system moemory to disk is the I/O path. This could be alleviated by dumping to multiple disks in parallel but the system firmware (IODC) isn't designed to permit multiple simultaneous I/O requests. Thus the only approach is to limit the amount of I/O that has to be done.

There is a new feature called *compressed dumps* available as of HP-UX Itanium release UX 11i v2 (i.e. UX 11.23) and additionally for UX 11i v1 (i.e. UX 11.11). The data is compressed (using LZO algorithm) before being written out to the dump device. When the system crashes, the dump subsystem assigns one processor to perform the writes to the dump device(s). It assigns another four processors to perform compression.

The dump compression features is targeted for large memory systems. Following requirements must be met:

Systems:	Superdome, Keystone, Matterhorn and Prelude
OS:	PA-RISC: UX 11i v1 (11.11) + patch Itanium: UX 11i v2 (11.23)
Configuration:	at least 2GB RAM, at least 5 processors

The compression option is turned ON by default. But it just a hint to the kernel. At the time of a system crash, the dump subsystem examines the state of the system and its resources to determine whether it is possible to use compression. Depending on the resources available, the system decides dynamically whether to dump compressed or not.

Other situations can cause the dump subsystem to decide not to dump compressed: recursive panic, memory allocation failure - all logged on system console at crash dump and flagged in the kernel.

HP can't guarantee a specific compression factor. All compression tends to be dependent on the type of data being compressed, in particular how random it is. The dump should speedup by at least a factor of 3 with default selective dump configuration. More typically, customers will experience a factor of 7.

The crashconf(1M) command was enhanced to be able to configure dump compression:

```
# crashconf -c on

# crashconf -v
CLASS          PAGES  INCLUDED IN DUMP  DESCRIPTION
--------  ----------  ----------------  ------------------------------
UNUSED       3645411  no, by default    unused pages
USERPG          7113  no, by default    user process pages
BCACHE        210990  no, by default    buffer cache pages
KCODE           2670  no, by default    kernel code pages
USTACK           264  yes, by default   user process stacks
FSDATA           116  yes, by default   file system metadata
KDDATA         68736  yes, by default   kernel dynamic data
KSDATA        259004  yes, by default   kernel static data

Total pages on system: 4194304
Total pages included in dump: 328120

Dump compressed: ON

DEVICE        OFFSET(kB)   SIZE (kB)  LOGICAL VOL.  NAME
------------  ----------  ----------  ------------  ----------------
 31:0x03a000      310112     4194304   64:0x000002  /dev/vg00/lvol2
                           ----------
                              4194304
```

If you like to make the configuration changes either for selective dump or for compressed dumps resistant across reboots you need to modify the rc-script `/etc/rc.config.d/crashconf`. Usually there should be no need to change the defaults.

The compressed dump feature uses a new crash dump format, PARDIR, for saving the dumps. You recognise a compressed dump with this evidences:

- In the INDEX file you will find a version 5.
- In the dump directory you will find `indexX.Y` files along with the usual `image.X.Y` files.

The dumpreading tools (p4, crashinfo, kmeminfo, etc...) are aware of this new format.

Since the dump is compressed you have little gain to compress it again with gzip, yet since

the compression is done with a 'compress(1)' compatible algorithm and small chunks, gzip'ing the dump still reduce it a bit sometime.

A consequence of the compressed dump is indeed a faster ”time to dump” and a somewhat faster “time to reboot” but the dumpreading tools suffer a serious performance penalty, making the “time to diagnose” or “time to fix” significantly longer.

NOTE: To enable compressed dump feature at UX 11.11 you need to install the CDUMP11i product from http://www.software.hp.com/ER_products_list.html. This product contains a set of enabling patches. At UX 11.23 the compressed dump feature is enabled in core, hence no product or patches are needed.

Documentation about the compressed dump feature can be found at in the “Managing Systems and Workgroups” paper at
http://www.docs.hp.com/hpux/os/11i/index.html#System%20Administration

Saving the dump to the filesystem

After the system has finished to write the whole or only parts of the dump to the dump devices, the system reboots and automatically starts up again. When booting up, the system starts a rc script to copy the dump into the file system.

As of UX 11.00 the rc script itself is `/sbin/init.d/savecrash`. The configuration file is stored at `/etc/rc.config.d/savecrash`. The default location is `/var/adm/crash` with sub directories named `crash.n` for every saved crash. The `crash.n` directory contains an ASCII file named `INDEX` that contains some metadata of the dump, a copy of the current kernel `vmunix` and files for every saved contiguous chunk of memory named `image.m.n`. If the kernel contains loadable modules, those are copied to the dump directory too.

You can configure crash directory, compression mode, etc. in the appropriate configuration file `/etc/rc.config.d/savecrash`:

Here are the most important options:

`SAVECRASH`	`1` = save a crashdump (default) `0` = do not save a crashdump
`SAVECRASH_DIR`	directory for the crashfiles. Default is `/var/adm/crash`
`COMPRESS`	`0` = never compress `1` = always compress `2` = compress in case of insufficient space in crasdirectory (default)

Further options (`MINFREE`, `SWAP_LEVEL`, `CHUNK_SIZE`, `SAVE_PART`, `FOREGRD`, `LOG_ONLY`) are explained in the comments of the config file.

Saving the dump manually

If the dump was not saved completely due to lack of space in the crash directory you have the possibility to save the dump again. The `-r` option (resave) need to be included when this is not the first time that savecrash runs.

```
# savecrash -v [-r] <crash directory>
```

There is also the possibility to save the dump directly to a DDS tape:

```
# savecrash -v [-r] -t /dev/rmt/0m
```