Documentation for Kdump - The kexec-based Crash Dumping Solution

This document includes overview, setup and installation, and analysis information.

Overview

Kdump uses kexec to quickly boot to a dump-capture kernel whenever a dump of the system kernel's memory needs to be taken (for example, when the system panics). The system kernel's memory image is preserved across the reboot and is accessible to the dump-capture kernel.

You can use common commands, such as cp and scp, to copy the memory image to a dump file on the local disk, or across the network to a remote system.

Kdump and kexec are currently supported on the x86, x86_64, ppc64 and ia64 architectures.

When the system kernel boots, it reserves a small section of memory for the dump-capture kernel. This ensures that ongoing Direct Memory Access (DMA) from the system kernel does not corrupt the dump-capture kernel. The kexec -p command loads the dump-capture kernel into this reserved memory.

On x86 machines, the first 640 KB of physical memory is needed to boot, regardless of where the kernel loads. Therefore, kexec backs up this region just before rebooting into the dump-capture kernel.

Similarly on PPC64 machines first 32KB of physical memory is needed for booting regardless of where the kernel is loaded and to support 64K page size kexec backs up the first 64KB memory.

All of the necessary information about the system kernel's core image is encoded in the ELF format, and stored in a reserved area of memory before a crash. The physical address of the start of the ELF header is passed to the dump-capture kernel through the elfcorehdr= boot parameter.

With the dump-capture kernel, you can access the memory image, or "old memory," in two ways:

- Through a /dev/oldmem device interface. A capture utility can read the device file and write out the memory in raw format. This is a raw dump of memory. Analysis and capture tools must be intelligent enough to determine where to look for the right information.

- Through /proc/vmcore. This exports the dump as an ELF-format file that you can write out using file copy commands such as cp or scp. Further, you can use analysis tools such as the GNU Debugger (GDB) and the Crash tool to debug the dump file. This method ensures that the dump pages are correctly ordered.

Setup and Installation
======================

Install kexec-tools
----

1) Login as the root user.

2) Download the kexec-tools user-space package from the following URL:

http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/kexec-tools.tar.gz

This is a symlink to the latest version.

The latest kexec-tools git tree is available at:

git://git.kernel.org/pub/scm/linux/kernel/git/horms/kexec-tools.git
or
http://www.kernel.org/git/?p=linux/kernel/git/horms/kexec-tools.git

More information about kexec-tools can be found at
http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/README.html

3) Unpack the tarball with the tar command, as follows:

   tar xvpzf kexec-tools.tar.gz

4) Change to the kexec-tools directory, as follows:

   cd kexec-tools-VERSION

5) Configure the package, as follows:

   ./configure

6) Compile the package, as follows:

   make

7) Install the package, as follows:

   make install

Build the system and dump-capture kernels

There are two possible methods of using Kdump.

1) Build a separate custom dump-capture kernel for capturing the
   kernel core dump.

2) Or use the system kernel binary itself as dump-capture kernel and there is
   no need to build a separate dump-capture kernel. This is possible
   only with the architectures which support a relocatable kernel. As

of today, i386, x86_64, ppc64 and ia64 architectures support relocatable kernel.

Building a relocatable kernel is advantageous from the point of view that one does not have to build a second kernel for capturing the dump. But at the same time one might want to build a custom dump capture kernel suitable to his needs.

Following are the configuration setting required for system and dump-capture kernels for enabling kdump support.

System kernel config options
---

1) Enable "kexec system call" in "Processor type and features."

   CONFIG_KEXEC=y

2) Enable "sysfs file system support" in "Filesystem" -> "Pseudo filesystems." This is usually enabled by default.

   CONFIG_SYSFS=y

   Note that "sysfs file system support" might not appear in the "Pseudo filesystems" menu if "Configure standard kernel features (for small systems)" is not enabled in "General Setup." In this case, check the .config file itself to ensure that sysfs is turned on, as follows:

   grep 'CONFIG_SYSFS' .config

3) Enable "Compile the kernel with debug info" in "Kernel hacking."

   CONFIG_DEBUG_INFO=Y

   This causes the kernel to be built with debug symbols. The dump analysis tools require a vmlinux with debug symbols in order to read and analyze a dump file.

Dump-capture kernel config options (Arch Independent)
---

1) Enable "kernel crash dumps" support under "Processor type and features":

   CONFIG_CRASH_DUMP=y

2) Enable "/proc/vmcore support" under "Filesystems" -> "Pseudo filesystems".

   CONFIG_PROC_VMCORE=y
   (CONFIG_PROC_VMCORE is set by default when CONFIG_CRASH_DUMP is selected.)

Dump-capture kernel config options (Arch Dependent, i386 and x86_64)
---

1) On i386, enable high memory support under "Processor type and features":

CONFIG_HIGHMEM64G=y
or
CONFIG_HIGHMEM4G

2) On i386 and x86_64, disable symmetric multi-processing support
under "Processor type and features":

CONFIG_SMP=n

(If CONFIG_SMP=y, then specify maxcpus=1 on the kernel command line
when loading the dump-capture kernel, see section "Load the Dump-capture
Kernel".)

3) If one wants to build and use a relocatable kernel,
Enable "Build a relocatable kernel" support under "Processor type and
features"

CONFIG_RELOCATABLE=y

4) Use a suitable value for "Physical address where the kernel is
loaded" (under "Processor type and features"). This only appears when
"kernel crash dumps" is enabled. A suitable value depends upon
whether kernel is relocatable or not.

If you are using a relocatable kernel use CONFIG_PHYSICAL_START=0x100000
This will compile the kernel for physical address 1MB, but given the fact
kernel is relocatable, it can be run from any physical address hence
kexec boot loader will load it in memory region reserved for dump-capture
kernel.

Otherwise it should be the start of memory region reserved for
second kernel using boot parameter "crashkernel=Y@X". Here X is
start of memory region reserved for dump-capture kernel.
Generally X is 16MB (0x1000000). So you can set
CONFIG_PHYSICAL_START=0x1000000

5) Make and install the kernel and its modules. DO NOT add this kernel
to the boot loader configuration files.

Dump-capture kernel config options (Arch Dependent, ppc64)
---

1) Enable "Build a kdump crash kernel" support under "Kernel" options:

CONFIG_CRASH_DUMP=y

2) Enable "Build a relocatable kernel" support

CONFIG_RELOCATABLE=y

Make and install the kernel and its modules.

Dump-capture kernel config options (Arch Dependent, ia64)

- No specific options are required to create a dump-capture kernel
  for ia64, other than those specified in the arch independent section
  above. This means that it is possible to use the system kernel
  as a dump-capture kernel if desired.

  The crashkernel region can be automatically placed by the system
  kernel at run time. This is done by specifying the base address as 0,
  or omitting it all together.

  crashkernel=256M@0
  or
  crashkernel=256M

  If the start address is specified, note that the start address of the
  kernel will be aligned to 64Mb, so if the start address is not then
  any space below the alignment point will be wasted.

Extended crashkernel syntax
==========

While the "crashkernel=size[@offset]" syntax is sufficient for most
configurations, sometimes it's handy to have the reserved memory dependent
on the value of System RAM -- that's mostly for distributors that pre-setup
the kernel command line to avoid a unbootable system after some memory has
been removed from the machine.

The syntax is:

    crashkernel=<range1>:<size1>[,<range2>:<size2>,...][@offset]
    range=start-[end]

    'start' is inclusive and 'end' is exclusive.

For example:

    crashkernel=512M-2G:64M,2G-:128M

This would mean:

    1) if the RAM is smaller than 512M, then don't reserve anything
       (this is the "rescue" case)
    2) if the RAM size is between 512M and 2G (exclusive), then reserve 64M
    3) if the RAM size is larger than 2G, then reserve 128M

Boot into System Kernel
==========

1) Update the boot loader (such as grub, yaboot, or lilo) configuration
   files as necessary.

2) Boot the system kernel with the boot parameter "crashkernel=Y@X",
   where Y specifies how much memory to reserve for the dump-capture kernel
   and X specifies the beginning of this reserved memory. For example,

"crashkernel=64M@16M" tells the system kernel to reserve 64 MB of memory
starting at physical address 0x01000000 (16MB) for the dump-capture kernel.

On x86 and x86_64, use "crashkernel=64M@16M".

On ppc64, use "crashkernel=128M@32M".

On ia64, 256M@256M is a generous value that typically works.
The region may be automatically placed on ia64, see the
dump-capture kernel config option notes above.

Load the Dump-capture Kernel
============================

After booting to the system kernel, dump-capture kernel needs to be
loaded.

Based on the architecture and type of image (relocatable or not), one
can choose to load the uncompressed vmlinux or compressed bzImage/vmlinuz
of dump-capture kernel. Following is the summary.

For i386 and x86_64:
	- Use vmlinux if kernel is not relocatable.
	- Use bzImage/vmlinuz if kernel is relocatable.
For ppc64:
	- Use vmlinux
For ia64:
	- Use vmlinux or vmlinuz.gz

If you are using a uncompressed vmlinux image then use following command
to load dump-capture kernel.

kexec -p <dump-capture-kernel-vmlinux-image> \
--initrd=<initrd-for-dump-capture-kernel> --args-linux \
--append="root=<root-dev> <arch-specific-options>"

If you are using a compressed bzImage/vmlinuz, then use following command
to load dump-capture kernel.

kexec -p <dump-capture-kernel-bzImage> \
--initrd=<initrd-for-dump-capture-kernel> \
--append="root=<root-dev> <arch-specific-options>"

Please note, that --args-linux does not need to be specified for ia64.
It is planned to make this a no-op on that architecture, but for now
it should be omitted

Following are the arch specific command line options to be used while
loading dump-capture kernel.

For i386, x86_64 and ia64:
	"1 irqpoll maxcpus=1 reset_devices"

For ppc64:
	"1 maxcpus=1 noirqdistrib reset_devices"

Notes on loading the dump-capture kernel:

* By default, the ELF headers are stored in ELF64 format to support
  systems with more than 4GB memory. On i386, kexec automatically checks if
  the physical RAM size exceeds the 4 GB limit and if not, uses ELF32.
  So, on non-PAE systems, ELF32 is always used.

  The --elf32-core-headers option can be used to force the generation of ELF32
  headers. This is necessary because GDB currently cannot open vmcore files
  with ELF64 headers on 32-bit systems.

* The "irqpoll" boot parameter reduces driver initialization failures
  due to shared interrupts in the dump-capture kernel.

* You must specify <root-dev> in the format corresponding to the root
  device name in the output of mount command.

* Boot parameter "1" boots the dump-capture kernel into single-user
  mode without networking. If you want networking, use "3".

* We generally don' have to bring up a SMP kernel just to capture the
  dump. Hence generally it is useful either to build a UP dump-capture
  kernel or specify maxcpus=1 option while loading dump-capture kernel.

Kernel Panic
============

After successfully loading the dump-capture kernel as previously
described, the system will reboot into the dump-capture kernel if a
system crash is triggered. Trigger points are located in panic(),
die(), die_nmi() and in the sysrq handler (ALT-SysRq-c).

The following conditions will execute a crash trigger point:

If a hard lockup is detected and "NMI watchdog" is configured, the system
will boot into the dump-capture kernel (die_nmi()).

If die() is called, and it happens to be a thread with pid 0 or 1, or die()
is called inside interrupt context or die() is called and panic_on_oops is set,
the system will boot into the dump-capture kernel.

On powerpc systems when a soft-reset is generated, die() is called by all cpus
and the system will boot into the dump-capture kernel.

For testing purposes, you can trigger a crash by using "ALT-SysRq-c",
"echo c > /proc/sysrq-trigger" or write a module to force the panic.

Write Out the Dump File
=======================

After the dump-capture kernel is booted, write out the dump file with
the following command:

   cp /proc/vmcore <dump-file>

You can also access dumped memory as a /dev/oldmem device for a linear and raw view. To create the device, use the following command:

```
mknod /dev/oldmem c 1 12
```

Use the dd command with suitable options for count, bs, and skip to access specific portions of the dump.

To see the entire memory, use the following command:

```
dd if=/dev/oldmem of=oldmem.001
```

Analysis

Before analyzing the dump image, you should reboot into a stable kernel.

You can do limited analysis using GDB on the dump file copied out of /proc/vmcore. Use the debug vmlinux built with -g and run the following command:

```
gdb vmlinux <dump-file>
```

Stack trace for the task on processor 0, register display, and memory display work fine.

Note: GDB cannot analyze core files generated in ELF64 format for x86. On systems with a maximum of 4GB of memory, you can generate ELF32-format headers using the --elf32-core-headers kernel option on the dump kernel.

You can also use the Crash utility to analyze dump files in Kdump format. Crash is available on Dave Anderson's site at the following URL:

```
http://people.redhat.com/~anderson/
```

To Do

1) Provide relocatable kernels for all architectures to help in maintaining multiple kernels for crash_dump, and the same kernel as the system kernel can be used to capture the dump.

Contact

Vivek Goyal (vgoyal@in.ibm.com)
Maneesh Soni (maneesh@in.ibm.com)