Diskdump White Paper

Revision 0.1

Table of Contents

1. Introduction

1.1. What is diskdump?

Diskdump offers a function to execute so-called crashdump. When "panic" or "oops" happens, diskdump automatically saves system memory to the disk. We can investigate the cause of panic using this saved memory image.

1.2. How does it work?

This is 2-stage dump which is similar to traditional UNIX dump. The 1st stage starts when panic occurs, at which time the register state and other dump-related data is stored in a header, followed by the full contents of memory, in a dedicated dump partition. The dump partition will have been pre-formatted with per-block signatures.

The 2nd stage is executed by rc script after the next system reboot, at which time the savecore command will create the vmcore file from the contents of the dump partition. After that, the per-block signatures will be re-written over the dump partition, in preparation for the next panic.

The handling of panic is essentially the same as netdump. It inhibits interrupts, freezes all other CPUs, and then for each page of data, issues the I/O command to the host adapter driver, followed by calling the interrupt handler of the adapter driver iteratively until the I/O has completed. The difference compared to netdump is that diskdump saves memory to the dump partition in its own loop, and does not wait for instructions from an external entity.

2. Design Overview

2.1. Safety

When diskdump is executed, of course the system is in serious trouble. Therefore, there is a possibility that user resources on the disk containing the dump partition could be corrupted. To avoid this danger, signatures are written over the complete dump partition. When a panic occurs, the diskdump module reads the whole dump partition, and checks if the signatures are written correctly. If the signatures match, the diskdump presumes that it will be writing to the correct device, and that there is a high possibility that it will be able to write the dump correctly. The signatures should be the ones which have low possibility to exist on the disk.

We decided that the following format will be written in each one block (page-size) unit on the partition. The signatures are created by a simple formulas based on the block number, making it a low possibility that the created signature would ever be the same as a user resource:

32-bit word 0: "disk"
32-bit word 1: "dump"
32-bit word 2: block number
32-bit word 3: (block number+3)*11
32-bit word 4: ((word 3)+3)*11
32-bit work 5: ((word 4)+3)*11
32-bit work 6: ((word 5)+3)*11
...
32-bit work 1023: ((word 1022)+3)*11

The diskdump module also verifies that its code and data contents have not been corrupted. The dump module computes CRC value of its module at the point that dump device is registered, and saves it. When panic occurs, the dump module re-computes the CRC value at that point and compares with the saved value. If the values aren't the same, the dump knows that it has been damaged and aborts the dump process.

2.2. Reliability

After panic occurs, I/O is executed by the diskdump module calling the queuecommand() function of the host adapter driver, and by polling the interrupt handler directly. The dump is executed by diskdump module and host adapter driver only. It is executed without depending on other components of kernel, so it works even when panic occurs in interrupt context. (XXX To be exact, a couple of drivers are not finished completely, because they calls kmalloc() as an extension of queuecommand())

In SCSI, a host reset is executed first, so it is possible to dump with a stable bus condition. In a couple of drivers, especially in the host reset process, timers and tasklets may be used. For these drivers, I created a helper library to emulate these functions. The helper library executes timer and tasklet functionality, which helps to minimize the modification required to support diskdump. The size of initrd increases slightly because the driver depends upon the helper library.

Multiple dump devices can be registered. When a panic occurs, the diskdump module checks each dump device condition and selects the first sane device that it finds.

Diskdump and netdump can co-exist. When both of modules are enabled, the diskdump works in preference to the netdump. If the signature checking fails, if a disk I/O error occurs, or if a double panic occurs in the diskdump process, it falls back to netdump.

2.3. Impact to kernel

The host adapter driver needs to be modified to support diskdump, but the required steps are small. For example, the modification patch for the aic7xxx/aic9xxx drivers contains 100 lines for each. At a minimum, a poll handler needs to be added, which is called from diskdump to handle I/O completion. If the adapter driver does not use timers or tasklets, that's all that is required. Even if timers or tasklets are used, it only requires a small amount of code from the emulation library.

2.4. Supported device

- Adaptec AIC7xxx Fast/U160 Driver
- Adaptec AIC79xx U320 Driver
- LSI Logic Fusion MPT ScsiHost Driver
- SYMBIOS/LSILOGIC SYM53C8XX family PCI-SCSI controller

- IBM Power Linux RAID adapter
- IDE

3. Comparison with other crash dump

There are other crash dump implementations for Linux:

- LKCD
- netdump
- kdump
- mkdump

In this section, let us see the characteristics and problems of each implementation and compare them.

3.1. Characteristics

These points need to be considered to discuss crash dump functionality:

- Robustness
- Safety
- Trigger
- Dump device
- Additional functions
- Format of core file
- Analysis tools
- Platform dependency

There could be other points of view such as wideness of usage, adoption of distributions and activeness of community. However, we focus on technical aspects in this section.

3.2. LKCD

LKCD

http://lkcd.sourceforge.net/

LKCD implements 3 different approaches of crash dump:

- dump to disk
- dump to network
- dump to memory (and reboot preserving the memory)

LKCD started with dump-to-disk approach using generic block I/O of Linux. It evolves to integrate dump-to-network method (similar to netdump) and dump-to-memory method (similar to kdump/mkdump).

From the robustness viewpoint, as LKCD uses kernel's generic I/O mechanism to write a dump to disk, if the crash dump is triggered while the related resources are held, it will cause deadlock. From the safety viewpoint, it could destroy disk contents in theory if some of the data struct maliciously broken, though such destruction has not been observed in reality.

See netdump section for dump-to-network method and kdump section for dump-to-memory method. Advantages and problems are same with these implementations.

To trigger crash dump, panic and "SysRq + Alt + c" are used for LKCD.

As to the dump device, LKCD uses a swap partition as a dump device. It writes a dump to the dump device and, after reboot, it will be converted to a normal file for analysis.

LKCD has rich functionality such as compression, partial dump and live dump. Compression includes RLE and gzip. Partial dump function can select memory pages to be dumped: kernel pages only, used pages only or all pages. Livedump function has a problem that various operations in user space timeouts during crash dump.

Core file of LKCD has its original format. lcrash and crash can read the file.

Except for dump-to-memory method, most of the LKCD code is architecture independent. It has been ported to ppc64, ia64, x86-64 and arm.

3.3. netdump

netdump

http://www.redhat.com/support/wpapers/redhat/netdump/

netdump uses polling mode I/O to send memory image to dump server in network. Polling mode I/O function is implemented in a few NIC driver.

Though the I/O operation itself is robust due to polling mode I/O, the robustness in total is questionable because of disturbances in network environment like unexpected packet lost.

From the safety viewpoint, as it uses network, the problem of destroying disk contents is less possible than dump-to-disk solution.

netdump shares its triggers with diskdump. So triggers for diskdump also work for netdump: NMI switch, nmi_watchdog, IPF OS_INIT, SysRq + Alt + c, panic.

As the netdump server will convert the image on-the-fly to core file, netdump requires less storage space than LKCD/diskdump which require additional dump device. netdump is useful if the amount of disk space is limited on the host because the disk space is required on server side.

netdump has a functionality to send the contents of kernel log buffer to server before dumping memory image. So even if memory dump fails, it is possible to see the kernel log before crash. As for the security concern of using network, netdump provides authentication mechanism to avoid spoofing.

The format of the core file is ELF core format. crash and gdb can read the file, except that gdb cannot read the dump of over-4GB system on IA-32.

Core part of netdump is less architecture dependent but it requires NIC specific polling functions.

3.4. kdump

kdump uses kexec(*) to boot a dumper kernel with preserving memory image and the dumper kernel serves the saved memory image to user programs for any operations to put the image to storage.

As kexec is a functionality to bypass BIOS to boot kernel fast, the robustness of using it in crashed kernel depends mostly on the quality of device drivers such as dependency to firmware initialization.

If the reboot succeeds, successive process such as saving memory image to disks will work safely.

Currently, it is triggered only by panic.

From the functionality viewpoint, kdump does nothing but to provide /dev/oldmem and /proc/vmcore to export memory image to user space. However, it could be extended by user space programs flexibly.

/proc/vmcore is ELF core format and gdb can be used. However, gdb has a limitation of 4GB address space on IA-32.

kexec depends highly on platform by comparison with LKCD, netdump and diskdump, because it needs architecture specific implementations to reboot. Also, it implicitly depends on device driver implementations for them to work properly after kexec on panic.

3.5. mkdump

mkdump

http://mkdump.sourceforge.net/

mkdump is a similar approach to kdump except for the following points:

- reboot mechanism
- trigger
- dumper kernel

While kdump need to copy the first 640KB to reserved area before booting dumper kernel (on x86 platform), mkdump doesn't need.

mkdump is triggered by both panic and nmi_watchdog.

While kdump let user space programs to do any work after reboot, mkdump automatically writes memory image to disk by dumper kernel and reboot with normal kernel.

mkdump announces the existence of utility program to conver core file format between LKCD format and ELF core format. Thus lcrash and crash can be used for analysis.

3.6. diskdump

Same as netdump, it doesn't depend on kernel generic I/O subsystem and interrupt. In addition to it, its robustness is less dependent to environment like packet lost.

The evaluation result of diskdump vs. LKCD is as following:

Test:	Trigger crash dump by OS_INIT with random timeing while putting consistently high I/O load on disks.
Result:	Successful dump write ratio

diskdump (RHEL3)	100%	(300 success in 300 trials)
LKCD 4.1.1	69%	(49 success in 71 trials)
modified LKCD 4.1.1	91%	(131 success in 143 trials)

(*) Tested system: IPF 4CPUs, RAM 8GB, 2.4 series kernel LSI Logic mptfusion driver

(*) The modified LKCD includes changes in kernel to unlock several key resources before doing dump I/O.

As the result shows, the robustness of diskdump is very good.

For the safety, diskdump checks its sanity before executing crash dump to reduce the possibility of destructing disk contents.

diskdump can be triggered by NMI switch, nmi_watchdog, IPF OS_INIT.

From the dump device viewpoint, it requires about twice as large space as memory size because diskdump once write memory image to dump partition of memory size and convert it to core file after reboot.

The format of core file is ELF core format. crash and gdb can read the file, except that gdb cannot read the dump of over-4GB system on IA-32.

Like netdump, the core part of diskdump is less dependent to platform and has been ported to x86-64, ia64 and ppc64. On the other hand, it needs polling function in disk controller drivers and the availability is limited to the devices which has the function.

3.7. Conclusion

LKCD has rich functionality but its dump-to-disk method has structural problem to make it difficult to avoid deadlock in crash dump.

Netdump has a special advantage that it doesn't require local disk to save crash dump. However, its robustness needs improvement.

Kdump/mkdump is safe approach if the reboot by kexec succeeds and has capability to extend for higher functionality. However evaluation is needed case-by-case for the stability of device initialization process which is normally done by BIOS.

Diskdump has a problem that currently only a few disk controllers are diskdump-capable. However, it's a simple and robust implementation as of now.

4. How to support driver

4.1. Summary of driver reconstruction

The following is reconstruction of HBA driver.

1) Create new diskdump I/F functions.
2) Add those new I/F function pointers to the Scsi_Host_Template structure.
3) Don't enable the interrupt in the diskdump functions.
4) Use diskdump_mdelay() instead of mdelay() in the diskdump functions.
5) Beware that there are some timer functions that can't be used in diskdump functions.

4.2. Reference source

In this example, we used the Fusion MPT driver included in the kernel 2.6.9. You can see the necessary modifications in the following files.

- linux/driver/message/fusion/mptbase.c
- linux/driver/message/fusion/mptbase.h
- linux/driver/message/fusion/mptscsih.c
- linux/driver/message/fusion/mptscsih.h

4.3. Details of reconstruction

1) Create new diskdump I/F functions.

   HBA driver have to create new functions to support the diskdump. Refer to "4.4 Details of each I/F function" for the details of each I/F function.

2) Add those new I/F function pointers to the Scsi_Host_Template structure.

   Add new structure member, that is ".dump_sanity_check" and ".dump_poll", to Scsi_Host_Template structure.

ex.

```
static struct scsi_host_template driver_template = {
    .proc_name                    = "mptscsih",
    .proc_info                    = mptscsih_proc_info,
    .name                         = "MPT SCSI Host",
    .info                         = mptscsih_info,
    .queuecommand                 = mptscsih_qcmd,
        .slave_alloc                    = mptscsih_slave_alloc,
        .slave_configure                = mptscsih_slave_configure,
        .slave_destroy                  = mptscsih_slave_destroy,
        .eh_abort_handler               = mptscsih_abort,
        .eh_device_reset_handler        = mptscsih_dev_reset,
        .eh_bus_reset_handler           = mptscsih_bus_reset,
        .eh_host_reset_handler          = mptscsih_host_reset,
        .bios_param                     = mptscsih_bios_param,
        .can_queue                      = MPT_SCSI_CAN_QUEUE,
        .this_id                        = -1,
        .sg_tablesize                   = MPT_SCSI_SG_DEPTH,
        .max_sectors                    = 8192,
        .cmd_per_lun                    = 7,
        .use_clustering                 = ENABLE_CLUSTERING,
*       .dump_sanity_check              = mptscsih_sanity_check,
*       .dump_poll                      = mptscsih_poll,
    };
```

3) Don't enable the interrupt in the diskdump functions.
   If there are some functions that enable the interrupt while the diskdump is running, you have to modify those functions not to enable the interrupt.

   ex.

```
#define MPT_HOST_LOCK(host_lock)          ¥
    if (crashdump_mode())                 ¥
            spin_lock(host_lock);         ¥
    else                                  ¥
            spin_lock_irq(host_lock);

#define MPT_HOST_UNLOCK(host_lock)        ¥
    if (crashdump_mode())                 ¥
            spin_unlock(host_lock);       ¥
    else                                  ¥
            spin_unlock_irq(host_lock);
```

4) Use diskdump_mdelay() instead of mdelay() in the diskdump functions.

When mdelay() is used for the diskdump, there are the following problems. Therefore, it must be changed into diskdump_mdelay().

- If mdelay() is called when in_irq() is 1, stack trace will be displayed for WARN_ON.
- When the waiting time in mdelay() is long, dumping may fail by nmi_watchdog.

ex.

```
#define MPT_MDELAY(n)                     ¥
    if (crashdump_mode())                 ¥
            diskdump_mdelay(n);           ¥
    else                                  ¥
            mdelay(n);
```

5) The notes for the timer function

The diskdump does not use the interruption. You can't use the timer function with possibility of sleeping. But, the following timer functions can be used for the diskdump.

- add_timer()
- mod_timer()
- del_timer()
- del_timer_sync()
- tasklet_schedule()
- schedule_work()
- schedule_timeout()
- msleep()

The timer functions can't be used for the diskdump. In using it, please consult by the mailing list of the following sites.

http://sourceforge.net/projects/lkdump/

The following timer functions can't be used for the diskdump.

- schedule()
- tasklet_hi_schedule()

4.4. Details of each I/F function

1) sanity_check()

int (*sanity_check)(Scsi_Device *);

return value:

0	normal return
except 0	error return

Confirm whether I/O can be issued. You should return an error code if you decide that the device is not active.

ex.

```
int
mptscsih_sanity_check(struct scsi_device *sdev)
{
	MPT_ADAPTER	*ioc;
	MPT_SCSI_HOST	*hd;

	hd = (MPT_SCSI_HOST *) sdev->host->hostdata;
	if (!hd)
		return -ENXIO;
	ioc = hd->ioc;

	/* message frame freeQ is busy */
	if (spin_is_locked(&ioc->FreeQlock))
		return -EBUSY;

	return 0;
}
```

2) quiesce()

int (*quiesce)(Scsi_Device *);

return value

0	normal return
except 0	error return

Called after device is chosen as a dump device (by sanity_check). You have to do the following initialize operation.

- Release any driver lock

3) poll()

void (*poll)(Scsi_Device *);

The usual implementation of this function calls an interrupt service function, or carries out the same processing with interrupt service.

ex.

```
void
mptscsih_poll(struct scsi_device *sdev)
{
        MPT_SCSI_HOST   *hd;

        hd = (MPT_SCSI_HOST *) sdev->host->hostdata;
        if (!hd)
                return;

        /* check interrupt pending */
        mpt_poll_interrupt(hd->ioc);
}
```

ex.

```
mpt_poll_interrupt(MPT_ADAPTER *ioc)
{
        u32 intstat;

        intstat = CHIPREG_READ32(&ioc->chip->IntStatus);

        if (intstat & MPI_HIS_REPLY_MESSAGE_INTERRUPT)
                mpt_interrupt(0, ioc, NULL);
}
```

5. General Usage

5.1. Installation

1) Download software
   1. Linux kernel version 2.6.9

      linux-2.6.9.tar.bz2 can be downloaded from

      ftp://ftp.kernel.org/pub/linux/kernel/v2.6/
   2. diskdump kernel patch

      diskdump-1.0.tar.gz can be downloaded from the project page.

      http://sourceforge.net/projects/lkdump
   3. diskdumputils

      diskdumputils-0.6.1-1.tar.bz2 can be downloaded from the project page.
   4. crash command

      Download from here: ftp://people.redhat.com/anderson/
2) Build and Install Kernel
   1. Untar Linux kernel source

      ```
      # tar -xjvf  linux-2.6.9.tar.bz2
      ```
   2. Apply all patches in the diskdump-1.0.tar.gz

      ```
      # tar -xzvf  diskdump-1.0.tar.gz
      # cd linux-2.6.9
      # patch -p1 < ../diskdump-1.0
      ```
   3. Kernel Configuration

      ```
      # make menuconfig
      ```

      a. Under "Device Drivers"-> "Block devices", select the following:
         i. Select "m" for "Disk dump support".

      b. Under "Device Drivers" -> "SCSI device support", select the following:
         i. Select "m" for "SCSI dump support".

      c. Under "Kernel hacking", select the following:
         i. Select "y" for "Kernel debugging".
         ii. Select "y" for "Magic SysRq key". (optional)
         iii. Select "y" for "Compile the kernel with debug info".

      d. Configure other kernel config settings as needed.

4. Build kernel
```
# make
```
5. Install modules
```
# make modules_install
```
6. Build initrd if you need

   ex.
```
# mkinitrd initrd-2.6.9.img 2.6.9
```
7. Copy the kernel image to the boot directory

   ex.
```
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.9-diskdump
```
8. Reboot

3) Build and Install diskdumputils

1. Untar diskdumputils package
```
# tar -xjvf  diskdumputils-0.6.1-1.tar.bz2
```
2. Build
```
# make
```
3. Install
```
# make install
```

4) Build and Install crash

1. Untar crash package
```
# tar -xjvf  crash-3.10-11.tar.gz
```
2. Build
```
# make
```
3. Install
```
# make install
```

5.2. Setup

The setup procedure is as follows. First a dump device must be selected. Either whole device or a partition is fine. The dump device is wholly formatted for dump, it cannot be shared with a file system or as a swap partition. The size of dump device should be big enough to save the whole dump. The size to be written by the dump is the size of whole memory plus a header field. To determine the exact size, refer to the output kernel message after the diskdump module is loaded:
```
# modprobe diskdump
# dmesg | tail
```

ex.

```
disk_dump: total blocks required: 261909 (header 3 + bitmap
8 + memory 261898)
```

The total last number is the data size in pagesize units that will be written by the diskdump function.

Select the dump partition in /etc/sysconfig/diskdump, as in the following example:

```
-------------------
DEVICE=/dev/sde1
-------------------
```

Next, Format the dump partition. The administrator needs to execute this once.

```
# service diskdump initialformat
```

Lastly, enable the service:

```
# chkconfig diskdump on
# service diskdump start
```

If /proc/diskdump exists, and it shows the registered dump device, the diskdump has been activated:

```
# cat /proc/diskdump
/dev/sde1 514080 1012095
```

5.3. Test the diskdump

To test the diskdump, use Alt-SysRq-C or "echo c > /proc/sysrq-trigger". After completing the dump, a vmcore file will created during the next reboot sequence, and saved in a directory of the name format:

```
/var/crash/127.0.0.1-<date>
```

The dump format is same as the netdump one, so we can use crash command to analyze.

```
# crash vmlinux vmcore
```

5.4. Tuning parameters

The following module parameters can be set.

block_order:

Set a I/O block size order for dumping. The default value is 2 and this means that I/O block size is "pagesize << 2". On i386 architecture the size is actually 16KB. The bigger value is faster to dump but memory consumption will increase.

sample_rate:

Set a number of blocks in the dump partition which is inspected before dumping. The default value is 8 and this means "1 << 8" (256) blocks are inspected. 0 means all blocks.

ex.
The following means that I/O block size is 32KB and all blocks are inspected.

```
option diskdump 'block_order=3' 'sample_rate=0'
```

5.5. Note

If /var/crash partition is not enough to dump, the diskdump invokes /var/crash/script/diskdump-nospace script. Using diskdump-nospace script can make space in /var/crash partition.

6. TODO

In this section, future improvement of diskdump is discussed (part of them are common with other crash dump solution).

6.1. Partial dump

These days, servers with tens or hundreds of giga-bytes of memory are not rare. As current implementation of diskdump reserves disk space of memory size to write a dump, it is a natural requirement to reduce the dump size to save disk space. More serious problem is system down time during the dump. If the disk performance is 50MB/s, it will take 40 minutes to write a dump on the system with 128GB memory. This is critical to the situation on which high availability is required. Clustering may be a solution to avoid this problem, if you can afford it. This problem is common between other crash dump implementations.

If diskdump can select the part of memory areas which are useful for analysis and dump them only, both time and space will be saved.

6.2. Efficiency of dump device format

Current implementation writes dummy data to dump device while formatting. This makes it possible to detect the end of the dump even it was accidentally terminated in the middle. It is also possible to check the dummy data to avoid writing wrong device.

On the other hand, the formatting requires, for example, 40 minutes for dump partition of 128GB with 50MB/s disk to fill the entire partition with dummy data. If the formatting will be done while system boot up, it affects the down time. Otherwise if it's done after boot up, crash dump is not available while the formatting.

It is possible to reduce the formatting time by make the dummy data sparse, for example.

6.3. Redundancy in dump device

Dump unavailability by single disk/controller failure should be avoided especially on the production system, which requires dump analysis.

Though the current implementation checks sanity of disks and controllers, it just give up dump if the specified dump device is insane.

Redundancy in dump device is possible by permitting registration of multiple dump devices and let diskdump to select the device by the result of sanity check.

6.4. Driver porting

Diskdump is available only for diskdump capable drivers. Currently diskdump-capable drivers are aic7xxx, aic79xx, mptfusion, sym53c8xx, ipr and IDE. Diskdump for SATA, megaraid2, dpt_i2o and Fibre Channel will be required.

6.5. Saving swap space

Current implementation saves only the memory image in RAM. In case the analysis of user process is needed, saving memory image in swap space and recovering whole process image is necessary.

Appendix A. Support Platform List

- i386
- x86_64
- ia64
- ppc64

Appendix B. Support Driver List

- aic7xxx
- aic79xx
- mptfusion
- sym53c8xx
- ipr
- IDE