System And Crash Analysis

Moussa Sagna
Senior Technical Support Engineer
Welcome and Presentation
Overview

Issue Description and Environment

System and Monitoring Tools

Application Profiling - Overview

Crash Dump Analysis - Overview

Documentation
Issue Description and Environment
Issue Description and Environment

- Type of Machine:
  - Physical machine
  - Virtual machine (VM): XEN, VMware, KVM etc ...
- Detailed description of the issue (crash, performance etc.)
- Recent changes before the issue appears for the first time
- Several similar machines (ex. cluster environment, etc ...)
- Recurrent or one-time issue
- Purpose of the system
- Presence of 3-rd party software
- Tainted kernel
Issue Description and Environment

• Hardware and firmware information (certified or not ?)
• Information from healthy state of the system
  - Load average
  - Running processes
  - Processes in a D state
  - Zombie processes
  - Any other important details...
System and Monitoring Tools
Tools

→ System Monitoring

• `vmstat`
  (from the procps package)

• Install with `YaST` or `zypper in procps`

• It reports virtual memory statistics

• `vmstat` is a multi-purpose tool; it gives information about processes, memory, paging, block IO, interrupts and CPU activity.

• `vmstat [option] [delay [count]]`
### Tools

→ **System Monitoring**

- Example of `vmstat`

```bash
tux@mercury:~> vmstat -a 2
procs -----------memory---------- ---swap-- -----io---- -system-- -----cpu------
r  b   swpd   free  inact active   si   so    bi    bo   in   cs us sy  id wa st
0  0      0 750992 570648 548848    0    0     0     1    8    9  0  0 100  0  0
0  0      0 750984 570648 548912    0    0     0     0   63   48  1  0 99   0  0
0  0      0 751000 570648 548912    0    0     0     0   55   47  0  0 100  0  0
0  0      0 751016 570648 548944    0    0     0     0   57   50  0  0 100  0  0

```

```bash
tux@mercury:~> vmstat 2
procs -----------memory----------- ---swap-- -----io---- -system-- -----cpu------
r  b   swpd   free   buff   cache   si   so    bi    bo   in   cs us sy id wa st
32  1  26236 459640 110240 6312648    0    0  9944 2 4552 6597 95  5  0  0  0
23  1  26236 396728 110336 6136224    0    0  9588 0 4468 6273 94  6  0  0  0
35  0  26236 554920 110508 6166508    0    0  7684 2 7992 4474 4700 95  5  0  0  0
28  0  26236 518184 110516 6039996    0    0 10830 4 4446 4670 94  6  0  0  0
21  5  26236 716468 110684 6074872    0    0  8734 2 0534 4512 4061 96  4  0  0  0
```
Tools

→ System Monitoring

- sar/sadc

(from the sysstat package)

- Install with `YaST` or `zypper in sysstat`

- sar collects, reports or saves detailed information on system activities like CPU, memory, IRQ usage, IO or networking. sar can either generate reports on the fly or query existing reports gathered by the system activity data collector (sadc).
System Monitoring

- Data collection with sadc with the following default settings:
  - data collected in /var/log/sa/saDD
  - data is collected every ten minutes
  - summary report is written in /var/log/sa/sarDD
  - summary report is generated every 6 hours (/etc/sysstat/sysstat.cron)
  - data is collected by /usr/lib64/sa/sa1 or /usr/lib/sa/sa1
Tools

→ System Monitoring

• Generate report with sar:

  - reports can be generated on the fly with an interval seconds and a count.

  - sar generates reports from `/var/log/sa/saDD` with the `\`-f` option

```
sar 2 10
sar -f ~/reports/sar_2010_05_03
sar
cd /var/log/sa &&
sar -f sa01 -f sa02
```

# on-the-fly report, 10 times every 2 seconds
# queries file `sar_2010_05_03`
# queries file from today in `/var/log/sa/`
# queries files `/var/log/sa/0[12]`
Tools

→ System Monitoring

• Example of sar report:

```
m mercury:~ # sar 10 5
Linux 2.6.31.12-0.2-default (mercury) 03/05/10   _x86_64_   (2 CPU)
14:15:43   CPU    %user   %nice   %system   %iowait    %steal     %idle
14:15:53   all    38.55    0.00      6.10      0.10      0.00     55.25
14:16:03   all    12.59    0.00      4.90      0.33      0.00     82.18
14:16:13   all    56.59    0.00      8.16      0.44      0.00     34.81
14:16:23   all    58.45    0.00      3.00      0.00      0.00     38.55
14:16:33   all    86.46    0.00      4.70      0.00      0.00      8.85
Average:   all    49.94    0.00      5.38      0.18      0.00     44.50
```
Tools

→ System Monitoring

- Example of paging statistics with sar:

```
mercury:~ # sar -B 10 5
Linux 2.6.31.12-0.2-default (mercury) 03/05/10   _x86_64_   (2 CPU)
16:11:43 pgpgin/s pgpgout/s fault/s majflt/s pgfree/s pgscank/s pgscand/s pgsteal/s   %vmeff
16:11:53   225.20    104.00  91993.90     0.00  87572.60      0.00      0.00      0.00    0.00
16:12:03   718.32    601.00  82612.01     2.20  99785.69    560.56    839.24   1132.23   80.89
16:12:13  1222.00   1672.40 103126.00     1.70 106529.00   1136.00    982.40   1172.20   55.33
16:12:23   112.18     77.84 113406.59     0.10  97581.24     35.13    127.74    159.38   97.86
16:12:33   817.22     81.28 121312.91     9.41 111442.44    0.00      0.00      0.00    0.00
Average: 618.72    507.20 102494.86     2.68 100578.98    346.24    389.76    492.60   66.93
```

- For more information about sar/sadc, have a look at the man page.
Tools

→ System Monitoring

• Example of block device statistics report with sar:

```plaintext
mercury:~ # sar -d -p 10 5
Linux 2.6.31.12-0.2-default (neo)       03/05/10       _x86_64_       (2 CPU)
16:28:31 DEV  tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:28:41 sdc  11.51     98.50    653.45     65.32      0.10   8.83   4.87   5.61
16:28:41 scd0   0.00      0.00      0.00      0.00      0.00   0.00   0.00   0.00
16:28:51 DEV  tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:28:51 sdc  15.38    329.27    465.93     51.69      0.10   6.39   4.70   7.23
16:28:51 scd0   0.00      0.00      0.00      0.00      0.00   0.00   0.00   0.00
16:29:01 DEV  tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:29:01 sdc  32.47   876.72    647.35     46.94      0.33  10.20   3.67  11.91
16:29:01 scd0   0.00      0.00      0.00      0.00      0.00   0.00   0.00   0.00
16:29:11 DEV  tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz  await  svctm  %util
16:29:11 sdc  48.75  2852.45    366.77     66.04      0.82  16.93   4.91  23.94
```

• Network statistics reports sar -n KEYWORD
Tools

→ System Monitoring

• iostat
  (from the sysstat package)

• Install with `YaST` or `zypper in sysstat`

• iostat reports CPU statistics and input/output statistics for devices, partitions and network filesystems (NFS).

• This tool generates reports that can be used to better balance the load between the different CPUs, devices etc …
Tools

→ System Monitoring

• Example of iostat output

```
tux@dp-t3500:~> iostat
Linux 2.6.32.7-0.2-default (geeko@buildhost) 02/24/10  _x86_64_
avg-cpu:  %user  %nice %system %iowait %steal  %idle
         0,49   0,01   0,10   0,31   0,00  99,09
Device:   tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda      1,34       5,59       25,37   1459766   6629160
sda1     0,00       0,01       0,00    1519        0
sda2     0,87       5,11       17,83   1335365   4658152
sda3     0,47       0,47       7,54    122578    1971008
```

• iostat adds statistics of NFS, if it is invoked with the `-n` option.

• With the `-p` option, it is possible to specify a single device to be monitored.

More information about iostat is available in the man page.
Tools

→ System Monitoring

· mpstat

(from the sysstat package)

· Install with `YaST` or `zypper in sysstat`.

· mpstat reports processors related statistics. mpstat examines the activity of each available processor. If it is an uniprocessor system, global average statistics are reported.
Tools

→ System Monitoring

• Example of mpstat

tux@mercury:~> mpstat -P 1 2 5
Linux 2.6.32.7-0.2-default (geeko@buildhost)    02/24/10        _x86_64_
08:57:10    CPU  %usr   %nice  %sys %iowait  %irq  %soft  %steal  %guest  %idle
%guest  %idle
08:57:12    1    4.46    0.00    5.94    0.50    0.00    0.00    0.00  89.11
0.00  89.11
08:57:14    1    1.98    0.00    2.97    0.99    0.00    0.99    0.00  93.07
0.00  93.07
08:57:16    1    2.50    0.00    3.00    0.00    0.00    1.00    0.00  93.50
0.00  93.50
08:57:18    1   14.36    0.00    1.98    0.00    0.00    0.50    0.00  83.17
0.00  83.17
08:57:20    1    2.51    0.00    4.02    0.00    0.00    2.01    0.00  91.46
0.00  91.46
Average:    1    5.17    0.00    3.58    0.30    0.00    0.90    0.00  90.05
Tools
→ System Monitoring

• **pidstat**
  (part of package sysstat)

• Install with `YaST` or `zypper in sysstat`.

• pidstat reports statistics for linux tasks.
  It shows the load a particular process applies to the system.

• The `-P` option can be used to specify the processor for which statistics are to be reported.
Tools

→ System Monitoring

• Example of pidstat

```bash
tux@mercury:~> pidstat -C top 2 3
Linux 2.6.27.19-5-default (geeko@buildhost) 03/23/2009 _x86_64_
09:25:42 AM   PID   %usr %system %guest    %CPU   CPU  Command
09:25:44 AM   23576   37.62   61.39    0.00   99.01     1  top
09:25:46 AM   23576   37.00   62.00    0.00   99.00     1  top
09:25:48 AM   23576   38.00   61.00    0.00   99.00     1  top
Average:       PID   %usr %system %guest    %CPU   CPU  Command
 Average: 23576 37.54 61.46 0.00 99.00    -  top
```

• The `-C <string>` displays only tasks whose command name includes the string `<string>`. 

![SUSE logo](image)
Tools

→ System Monitoring

- **dmesg**
  (part of the util-linux package)
- Install with `YaST` or `zypper` in util-linux
- dmesg prints or controls the kernel ring buffer where the linux kernel keeps certain messages.
- Older events are logged into `/var/log/messages` or `/var/log/warn`.
- The size of this buffer can be increased by using the `-s` option of dmesg.
Tools  
→ System Monitoring

- `lsof`
  (part of the lsof package)
- Install with `YaST` or `zypper in lsof`
- `lsof` shows a list of the open files
- Use the `-p <PID>` option to view files open for the process with the <PID> number
- As the output of the command `lsof` is quite huge, it is possible to use `grep` to limit it

```
tux@mercury:~> lsof | grep CHR
bash   3838   tux   0u   CHR  136,0                      2   /dev/pts/0
bash   3838   tux   1u   CHR  136,0                      2   /dev/pts/0
```
Tools

→ System Monitoring

• Example of lsof

tux@mercury:~> lsof -p $$

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PID</th>
<th>USER</th>
<th>FD</th>
<th>TYPE</th>
<th>DEVICE</th>
<th>SIZE/OFF</th>
<th>NODE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>cwd</td>
<td>DIR</td>
<td>3,3</td>
<td>1512</td>
<td>117619</td>
<td>/home/tux</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>rtd</td>
<td>DIR</td>
<td>3,3</td>
<td>584</td>
<td>2</td>
<td>/</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>txt</td>
<td>REG</td>
<td>3,3</td>
<td>498816</td>
<td>13047</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>mem</td>
<td>REG</td>
<td>0,0</td>
<td>0 [heap] (stat: No such</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>mem</td>
<td>REG</td>
<td>3,3</td>
<td>217016</td>
<td>115687</td>
<td>/var/run/nscd/passwd</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>mem</td>
<td>REG</td>
<td>3,3</td>
<td>208464</td>
<td>11867</td>
<td>/usr/lib/locale/en_GB.</td>
</tr>
<tr>
<td>[...]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>mem</td>
<td>REG</td>
<td>3,3</td>
<td>366</td>
<td>9720</td>
<td>/usr/lib/locale/en_GB.</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>mem</td>
<td>REG</td>
<td>3,3</td>
<td>97165</td>
<td>8828</td>
<td>/lib/ld-2.3.6.so</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>0u</td>
<td>CHR</td>
<td>136,5</td>
<td>7</td>
<td></td>
<td>/dev/pts/5</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>1u</td>
<td>CHR</td>
<td>136,5</td>
<td>7</td>
<td></td>
<td>/dev/pts/5</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>2u</td>
<td>CHR</td>
<td>136,5</td>
<td>7</td>
<td></td>
<td>/dev/pts/5</td>
</tr>
<tr>
<td>bash</td>
<td>5552</td>
<td>tux</td>
<td>255u</td>
<td>CHR</td>
<td>136,5</td>
<td>7</td>
<td></td>
<td>/dev/pts/5</td>
</tr>
</tbody>
</table>
Tools

→ System Monitoring

• top

• top provides a dynamic real-time view of a running system.

• It can display system summary information as well as a list of tasks currently being managed by the Linux kernel.

• Top can be used in `batch mode` operation with the `-b` option. This could be used for sending output to other programs or to a file.
Tools

→ System Monitoring

• Example of top

tux@mercury:~> top -n 1
top - 17:06:28 up  2:10,  5 users,  load average: 0.00, 0.00, 0.00
Tasks:  85 total,   1 running,  83 sleeping,   1 stopped,   0 zombie
Cpu(s):  5.5% us,  0.8% sy,  0.8% ni, 91.9% id,  1.0% wa,  0.0% hi,  0.0% si
Mem: 515584k total, 506468k used,  9116k free,  66324k buffers
Swap: 658656k total,  0k used, 658656k free,  353328k cached

PID USER      PR  NI  VIRT  RES  SHR  S %CPU %MEM    TIME+  COMMAND
  1 root      16   0   700  272  236  S  0.0  0.1   0:01.33 init
  2 root      34  19     0    0    0  S  0.0  0.0   0:00.00 ksoftirqd/0
  3 root      10 -5     0    0    0  S  0.0  0.0   0:00.27 events/0
  4 root      10 -5     0    0    0  S  0.0  0.0   0:00.01 khelper
  5 root      10 -5     0    0    0  S  0.0  0.0   0:00.00 kthread
 11 root      10 -5     0    0    0  S  0.0  0.0   0:00.05 kblockd/0
 12 root      20 -5     0    0    0  S  0.0  0.0   0:00.00 kacpid
 472 root     20  0     0    0    0  S  0.0  0.0   0:00.00 pdflush
 473 root     15  0     0    0    0  S  0.0  0.0   0:00.06 pdflush
 475 root     11 -5     0    0    0  S  0.0  0.0   0:00.00 aio/0
Tools

System Monitoring

• iotop

• Available as of SUSE Linux Enterprise Server 11

• Install iotop with `YaST` or `zypper in iotop`

• iotop watches the I/O usage information output by the linux kernel and displays a table of current I/O usage by processes or threads on the system.

• iotop can also be used in a batch mode (-b) and its output stored in a file for later analysis.
Tools

→ System Monitoring

- Example of iotop

```bash
tux@mercury:~> iotop --only
Total DISK READ: 50.61 K/s | Total DISK WRITE: 11.68 K/s

<table>
<thead>
<tr>
<th>TID</th>
<th>PRIO</th>
<th>USER</th>
<th>DISK READ</th>
<th>DISK WRITE</th>
<th>SWAPIN</th>
<th>IO&gt;</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>3416</td>
<td>be/4</td>
<td>ke</td>
<td>50.61 K/s</td>
<td>0.00 B/s</td>
<td>0.00 %</td>
<td>4.05 %</td>
<td>find /</td>
</tr>
<tr>
<td>275</td>
<td>be/3</td>
<td>root</td>
<td>0.00 B/s</td>
<td>3.89 K/s</td>
<td>0.00 %</td>
<td>2.34 %</td>
<td>[jbd2/sda2-8]</td>
</tr>
<tr>
<td>5055</td>
<td>be/4</td>
<td>ke</td>
<td>0.00 B/s</td>
<td>3.89 K/s</td>
<td>0.00 %</td>
<td>0.04 %</td>
<td>emacs</td>
</tr>
</tbody>
</table>
```
Tools

→ System Monitoring

- /proc/<$PID>/smaps
  - With the usual tools like top, ps etc … it is difficult to determine with precision how much memory a process is consuming. Thus, the smaps subsystem has been introduced since kernel 2.6.14 to get more precise data.
  - These data are found under /proc/<$PID>/smaps.
  - smaps shows the number of clean and dirty data memory pages the process is actually using.
  - smaps differentiates between private and shared data
Tools

→ System Monitoring

· Example of smaps output

```
08048000-080bc000 r-xp 00000000 03:02 13130 /bin/bash
Size: 464 kB
Rss: 424 kB
Shared_Clean: 424 kB
Shared_Dirty: 0 kB
Private_Clean: 0 kB
Private_Dirty: 0 kB
```

· Possibility to use the command “pmap” which does give information about the process' memory mappings, that is, stack, data segments, mapped files etc.
Tools

→ Kernel Monitoring

• oprofile

• oprofile is a profiling tool for Linux systems on almost all architectures (x86, x86_64, ppc, s390, arm, etc.)

• In the past, profiling an application was done only through recompilation with profiling enabled at compiler level.

• Nowadays, we have the possibility to use oprofile.

• Install it with `YaST` or zypper in oprofile
Tools

→ Kernel Monitoring

• oprofile

• oprofile consists of a kernel module and a running daemon that collects data from the system.

• It can profile all parts of a running system, from the kernel (including modules and interrupt handlers) to shared libraries and binaries.

• No special recompilations, wrapper libraries are necessary. Even compiling the application with -g is only needed in certain circumstances (e.g. annotated sources).
Tools

Kernel Monitoring

- oprofile

- This tool is capable of monitoring all hardware events by using the performance counters (PC) provided by the CPUs.

- PC are hardware registers that count events.

- Every time a certain number of events has occurred, the PC value is recorded.

- oprofile can collect profiles of code based on the number of these events. Example: Cache misses, CPU cycles, etc.
Tools

→ Kernel Monitoring

• oprofile

• This recorded information is then aggregated into profiles for each binary.

• Usually when profiling an application a small overhead is expected, depending on the load of the system.
Tools
→ Kernel Monitoring

• opprofile

• Getting started
  - minimum setup is to indicate where the vmlinux corresponding to the running kernel is:
    # opcontrol --vmlinux=/boot/vmlinux-`uname -r`
  - If it is not necessary to profile the kernel itself:
    # opcontrol --no-vmlinux
  - Shows events that can be monitored on the system:
    # opcontrol --list-events
Tools

→ Kernel Monitoring

- oprofile

- The default event is CPU_CLK_UNHALTED

- If other events have to be monitored, option “–event” has to be used the following way:

  # opcontrol –event=CPU-CLK_UNHALTED:10000 –event=MISALIGN_MEM_REF:500

- start profiling:

  # opcontrol --start
Tools

Kernel Monitoring

• oprofile

• Start oprofile

  # opcontrol --start

• Launch the application to be profiled if it is not already running.

• Two choices to dump the collected data:
  - either dump the already collected data with

    # opcontrol --dump

  - or just shutdown oprofile:

    # opcontrol --shutdown
lenovo-w500:~ # oprofile -t1
Using /var/lib/oprofile/samples/ for samples directory.
CPU: Core 2, speed 2.534e+06 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00 (Unhalted core cycles) count 10000
Counted MISALIGN_MEM_REF events (number of misaligned data memory references) with a unit mask of 0x00 (No unit mask) count 5000

<table>
<thead>
<tr>
<th>CPU_CLK_UNHALTED</th>
<th>MISALIGN_MEM_REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>%</td>
</tr>
</tbody>
</table>

2440653 65.9194  5261 10.3528 vmlinux-3.7.10-1.16-desktop
209604  5.6612  30730  60.4719 libc-2.17.so
138352  3.7367  9489 18.6729 libQtGui.so.4.8.4
122089  3.2975   989  1.9462 libzypp.so.1200.15.0
 98556  2.6619  65  0.1279 oprofile
 86561  2.1742  92  0.1810 libxul.so
 69970  1.8898  382  0.5943 libQtCore.so.4.8.4
 62135  1.6782 125  0.2460 libglib-2.0.so.0.3400.3
 62012  1.6749  51  0.1004 libpthread-2.17.so
 41666  1.1254  25  0.0492 firefox

CPU_CLK_UNHALTED | MISALIGN_MEM_REF |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>%</td>
</tr>
</tbody>
</table>

36495 87.5894  20  80.0000 [vdso] (tgid:1298 range:0x7fffffff5ff000-0x7fffffff600000)
 4085  9.8042  4  16.0000 firefox
  841  2.0184  1  4.0000 anon (tgid:1298 range:0x7f8fc29d2000-0x7f8fc29f2000)
  230  0.5520  0  0 anon (tgid:1298 range:0x7f8fc3bef000-0x7f8fc3bff000)
   15  0.0360  0  0 anon (tgid:1298 range:0x7f8fc2988000-0x7f8fc2998000)

lenovo-w500:~ #
Application Debugging
Overview
Application debugging →strace

- strace

- strace intercepts and records the system calls which are called by a process and the signals received by the process.

- strace can be launched together with a command, or attached to an already running process.
Application debugging  →strace

• Examples of strace usage

tux@mercury:~> strace ls
execve("/bin/ls", ["ls"], [/* 52 vars */]) = 0
brk(0) = 0x618000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) \
  = 0x7f9848667000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) \
  = 0x7f9848666000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT \n(No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=200411, ...}) = 0
mmap(NULL, 200411, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9848635000
close(3) = 0
open("/lib64/librt.so.1", O_RDONLY) = 3
[...]
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) \
  = 0x7fd780f79000
write(1, "Desktop\nDocuments\nbin\ninst-sys\n", 31) = 31
exit_group(0) = ?
Application debugging
  → strace

• strace -e trace=network ...
  The -e option is used in order to trace only network related system calls.

• strace -o <output_file>
  the option -o is used to direct the output to a file for further analysis.

• strace -f
  The option -f is used to trace the child processes.
Application debugging

→ valgrind

- valgrind is a suite of tools for debugging and profiling programs.

- It consists of a core that provides a virtual CPU and a set of tools which perform some kind of debugging, profiling, or similar tasks.

- The architecture is modular, thus, tools can be created without changing or disturbing the existing structure.
Application debugging
→ valgrind

- No need to recompile or modify executables to run valgrind on them.

- Synopsis:
  valgrind [valgrind-options] my-program [my-program-options]

- valgrind is not shipped with standard SLES distribution but with SDK(SUSE Software Development Kit)

- Install it manually with `rpm -i ...` or `zypper in ...`
Application debugging
→ valgrind

• valgrind available tools:

  * Memcheck: a memory error detector
  * Cachegrind: a cache (time) profiler
  * Callgrind: a call-graph profiler
  * Massif: a heap(space) profiler
  * Helgrind: a thread error detector
Application debugging

→valgrind

• Most important option is --tool which tells valgrind which tool to run.

• memcheck is used per default if no tool is specified.

• memcheck detects:
  * use of uninitialized memory
  * memory leaks
  * reading or writing or memory after it has been freed
  * reading or writing inappropriate areas of stack
  * etc.
Application debugging → valgrind

```c
#include<stdio.h>
#include<stdlib.h>

void attrMem(void)
{
    int *p = malloc(10 * sizeof(int));
    p[10] = 5;  // 1-st problem: heap block overrun
    // 2-nd issue: the memory has not been free'd here
}

int main()
{
    attrMem();
    return 0;
}
```

1,17 All
Application debugging
→ valgrind

```
valgrind ./helloWorld
Invalid write of size 4
  at 0x400057A: attrMem (helloWorld.c:7)
  by 0x400058A: main (helloWorld.c:12)
Address 0x51e1068 is 0 bytes after a block of size 40 alloc'd
  at 0x42c2c7b: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
  by 0x40006d: attrMem (helloWorld.c:5)
  by 0x400058A: main (helloWorld.c:12)

HEAP SUMMARY:
  in use at exit: 40 bytes in 1 blocks
  total heap usage: 1 allocs, 0 frees, 40 bytes allocated

LEAK SUMMARY:
  definitely lost: 40 bytes in 1 blocks
  indirectly lost: 0 bytes in 0 blocks
  possibly lost: 0 bytes in 0 blocks
  still reachable: 0 bytes in 0 blocks
  suppressed: 0 bytes in 0 blocks
Rerun with --leak-check=full to see details of leaked memory
```

```
For counts of detected and suppressed errors, rerun with: -v
ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 2 from 2)
```
Crash Dump Analysis
Overview
Crash Dump Analysis
→ kexec/kdump

• Kexec is a system call that enables to load and boot into another kernel from the currently running one.

• The difference between a standard system boot and the kexec boot is that hardware initialization is not performed here.

• The second kernel (also called crash/capture kernel) is loaded at boot time, beside the working kernel.
Crash Dump Analysis → kexec/kdump

- The crash kernel runs in a reserved memory area
  - the old kernel is preserved and untouched

- Dump is done in the capture kernel context
  - devices are reinitialized to a sane state
Crash Dump Analysis

→ setup of kexec/kdump

- Installation of the following packages:
  * kexec-tools
  * kdump
  * kernel-$FLAVOR-debuginfo
  * crash
  * crosscrash (optional)

- with
  YaST, zypper in … or the command line
  `rpm -Uhv <package_name>.rpm`
Crash Dump Analysis
→ setup of kexec/kdump

- Edit /etc/sysconfig/kdump
  * KDUMP_KERNELVER="kdump"
  * KDUMP_COMMANDLINE=""
  * KEXEC_OPTIONS="--args-linux"
  * KDUMP_RUNLEVEL="1"
  * KDUMP_IMMEDIATE_REBOOT="yes"
  * KDUMP TRANSFER=""
  * KDUMP_SAVEDIR="/var/log/dump"
  * KDUMP_COMMANDLINE_APPEND=""
  * KDUMP_DUMPPDEV="/dev/sda3"
  * KDUMP_DUMPLEVEL="31"
  * KDUMP_DUMPFORMAT="compressed"
Crash Dump Analysis
→ setup of kexec/kdump

• Enable kdump init service:
  * via YaST, or
  * chkconfig kdump on

• Add the line crashkernel=128M@16M as boot option in /boot/grub/menu.lst.
  * 128 M is the reserved memory size for the capture kernel
  * 16 M is the offset of the capture kernel
Crash Dump Analysis
→ setup of kexec/kdump

• Set ENABLE_SYSRQ="yes" in /etc/sysconfig/sysctl
  The alternative on a running system is
  echo 1 > /proc/sys/kernel/sysrq

• Reboot the system or execute the command:

  # kexec -l /boot/vmlinuz --initrd=/boot/initrd \
  --append =`cat /proc/cmdline` \
  " crashkernel=256M-:64M@16M"

  # kexec -e
Crash Dump Analysis

→ setup of kexec/kdump

• !!! IMPORTANT!!!

The whole kdump/kexec configuration can be easily done with YaST
Crash Dump Analysis

→ kexec/kdump setup verification

• First thing to check after reboot is the content of /proc/iomem; an area with “Crash” corresponding to the size of the capture kernel must be present.

• Trigger a dump with

  * echo c > /proc/sysrq-trigger

  * the magic SysRq key combination (ALTGr+SysRq+c)
Crash Dump Analysis
→ kexec/kdump setup verification

• Normally a dump is generated per default in /var/log/dump unless there is a configuration change.

• If not, trigger the dump from the console to see at which level it does hang.

• It is also possible to trigger the dump with the sysrq key combinations
Crash Dump Analysis → with crash

- crash is a tool to analyze linux crash dump data or a live system.

- Synopsis:

  crash [OPTION] <namelist> <mapfile> <memory-image>

* namelist is the kernel compiled with “-g” option
* mapfile is the kernel symbol table
* memory-image is a kernel core dump file created by kdump, netdump, kvmdump, lkcd and xendump facilities.
Crash Dump Analysis → with crash

• Get the <NAMELIST> file from the debuginfo file:
  * rpm2cpio kernel-$FLAVOR-debuginfo.rpm \
    | cpio -ivd
  
  * the namelist file is to be found under
    usr/lib/debug/vmlinux-<kernel-release>.debug

• The mapfile is often delivered with the memory dump, otherwise it is to be fetched from the corresponding kernel-base package.
Crash Dump Analysis → with crash

• For the memory-image (vmcore), it is important to get the md5sum of the uploaded file to make sure that the dump has been completely transferred.

Example:

# md5sum vmcore
6a185ABE587b11902e9120040644abe vmcore
Crash Dump Analysis

→ example of crash output

tux@mercury:~> crash /boot/vmlinux-2.6.32.8-0.1-default.gz
/var/crash/2010-04-23-11:17/vmcore
crash 4.0-7.6

    KERNEL: /boot/vmlinux-2.6.32.8-0.1-default.gz
DEBUGINFO: /usr/lib/debug/boot/vmlinux-2.6.32.8-0.1-default.debug
         CPUS: 2
         DATE: Thu Apr 23 13:17:01 2010
         UPTIME: 00:10:41
LOAD AVERAGE: 0.01, 0.09, 0.09
         TASKS: 42
NODENAME: eros
RELEASE: 2.6.32.8-0.1-default
VERSION: #1 SMP 2010-03-31 14:50:44 +0200
MACHINE: x86_64 (2999 Mhz)
MEMORY: 1 GB
    PANIC: "SysRq : Trigger a crashdump"
        PID: 9446
COMMAND: "bash"
    TASK: ffff88003a57c3c0 [THREAD_INFO: ffff880037168000]
    CPU: 1
    STATE: TASK_RUNNING (SYSRQ)
crash>
Crash Dump Analysis

→ with crash

• Important commands:
  * help <command> to get details on a command
  * bt
  * log
  * whatis <function_name>
  * struct
  * runq
  * print
Crash Dump Analysis

→ kernel oops

• When the kernel dereferences an invalid pointer, this is called an oops. Kernel oops is not a kernel panic; however it does show a certain inconsistency in the system and might lead, for ex., to the killing of the current task.

• An oops might lead to a kernel panic if a very important part of the system is destroyed.

• During an oops, the system will print a stack trace and the content of the EIP/RIP registers.
Crash Dump Analysis
→ kernel panic

• During a panic, the system cannot continue working. It is completely dead and has to be rebooted.
KERNEL: vmlinuz-2.6.32.54-0.3-default.gz
DEBUGINFO: vmlinuz-2.6.32.54-0.3-default.debug
DUMPFILE: vmcore [PARTIAL DUMP]
CPUS: 8
DATE: Fri Jun 29 02:11:42 2012
UPTIME: 5 days, 23:17:52
LOAD AVERAGE: 6.76, 8.41, 9.76
TASKS: 1214
NODENAME: misu570a01
RELEASE: 2.6.32.54-0.3-default
VERSION: #1 SMP 2012-01-27 17:38:56 +0100
MACHINE: x86_64 (1866 Mhz)
MEMORY: 48 GB
PANIC: "[515350.113849] kernel BUG at /usr/src/packages/BUILD/kernel-default-2.6.32.54/linux-2.6.32/mm/slab.c:3006!"
PID: 4267
COMMAND: "nsrmd"
TASK: ffff8807a17be040 [THREAD_INFO: ffff8807c8cb4000]
CPU: 1
STATE: TASK_RUNNING (PANIC)
crash> bt
PID: 4267  TASK: ffff8807a17be040  CPU: 1  COMMAND: "nsrmd"
#0 [fff8807c8cb57b0] machine_kexec at fffffff81020a62
#1 [fff8807c8cb5800] crash_kexec at fffffff81088780
#2 [fff8807c8cb58d0] oops_end at fffffff8139e0
#3 [fff8807c8cb58f0] do_invalid_op at fffffff81004b74
#4 [fff8807c8cb5990] invalid_op at fffffff81003d55
    [exception RIP: cache_alloc_refill+506]
RIP: fffffff810f09fa  RSP: ffff8807c8cb5a48  RFLAGS: 00010016
RAX: 0000000000000001  RBX: ffffffff88054f87000  RCX: 0000000000000000
RDX: 0000000000000000  RSI: ffff880631987550  RDI: ffff880631987560
RBP: ffff88063c3af800  R8: ffff880631987560  R9: ffff880270f4e0e0
R10: ffff8807c8cb5a58  R11: dead00000100100  R12: ffff880c1a81280
R13: ffff880631987540  R14: 0000000000000000  R15: ffff88062b5ab000
ORIG RAX: ffffffffffffffff  CS: 0010  SS: 0018
#5 [fff8807c8cb5a40] cache_alloc_refill at fffffff810f0885
#6 [fff8807c8cb5a90] kmem_cache_alloc at fffffff810f11c6
#7 [fff8807c8cb5b10] nfs_writepage_setup at ffffffffa0cc0814 [nfs]
#8 [fff8807c8cb5b50] nfs_updatepage at ffffffffa0cc0957 [nfs]
#9 [fff8807c8cb5b90] nfs_write_end at ffffffffa0caffa4 [nfs]
#10 [fff8807c8cb5bd0] generic_perform_write at fffffff810b5e2
#11 [fff8807c8cb5c50] generic_file_buffered_write at fffffff810b7a3e
#12 [fff8807c8cb5cb0] __generic_file_aio_write at fffffff810b8432
#13 [fff8807c8cb5d60] generic_file_aio_write at fffffff810b8675
#14 [fff8807c8cb5da0] nfs_file_write at ffffffffa0caf9e6 [nfs]
#15 [fff8807c8cb5de0] do_sync_write at fffffff810f1b3
#16 [fff8807c8cb5f10] vfs_write at fffffff810f7ce
#17 [fff8807c8cb5f40] sys_write at fffffff810f943
#18 [fff8807c8cb5f80] system_call_fastpath at fffffff81002f7b
RIP: 00007f61a36a500  RSP: 00007f555c7c4628  RFLAGS: 000000202
RAX: 0000000000000001  RBX: fffffff81002f7b  RCX: 0000000000000000
RDX: 0000000000000000  RSI: 00007f555c7c4780  RDI: 0000000000000000
RBP: 0000000000000000  R8: 00007f555c7c4628  R9: 0000000000000000
R10: 0000000000000001  R11: 0000000000000246  R12: 0000000000000000
R13: 0000000000000000  R14: 0000000000000000  R15: 0000000000000000
ORIG RAX: 0000000000000000  CS: 0033  SS: 002b

\"\"
Documentation

• http://oprofile.sourceforge.net/
• http://www.x86-64.org/documentation/abi.pdf
• http://www.x86-64.org/documentation/assembly.html
• http://people.redhat.com/anderson/crash_whitepaper
QUESTIONS?

Thank you.
Unpublished Work of SUSE. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.