Help! My (Virtual) Penguin Is Sick!
Or
*Aptenodytes Patagonicus* Problems on z/VM

Phil Smith III
Levanta, Inc.
SHARE 103
August 2004
Session 9248

*King Penguin, of course!*
Copyright Information

SHARE Inc. is hereby granted a non-exclusive license to copy, reproduce or republish this presentation in whole or in part for SHARE activities only, and the further right to permit others to copy, reproduce, or republish this presentation in whole or in part, so long as such permission is consistent with SHARE's By-laws, Canons of Conduct, and other directives of the SHARE Board of Directors.
Why We’re Here

The difference between applications people and systems people:

Applications people worry about how it will work.

**Systems people worry about how it will fail.**

➢ If you support production, you’re a systems person!
Agenda

We’ll cover:

• Ways Linux can get sick
• Techniques to decide what’s wrong
• Debugging information you can gather

We won’t cover:

• Detailed use of debugging tools (gdb, et al.)
• Dump (core) analysis

➢ Paramedic/first response functionality, not ER surgery or pathology lab forensic reports!
Penguins and Bears, Oh My!

Penguin Diseases 101
The Modal Penguin Ailment

“Why isn’t my Linux virtual machine responding?”, aka:

1. Can I get from here to there?
2. If I can get there, is there a “there” there?
3. If there is a “there” there, is it open?

Problems with these correspond to:

- Networking problems
- Linux issues
- VM troubles
A Baseline is Useful!

- Linux guests vary widely
  - Networking configuration
  - Performance profile
  - Services provided
  - Etc., etc. etc.

- Keep *written* (and online) notes about production guests
  - IP address(es), network interfaces, routing, etc.
  - Typical/observed performance characteristics
  - Disk space usage
  - Etc., etc., etc.

- In a crisis, you need to know how things *should* look!
Network Issues

- **Is it a network issue:**
  - Between the user and VM?
  - Between the VM stack and the Linux virtual machine?
  - Within the Linux virtual machine?

- **If you can’t get to the machine, it sure won’t respond!**
VM Troubles

- Is the Linux virtual machine even logged on?
  - Someone might have logged it off, FORCEd it, etc.

- Is the virtual machine in a stopped state?
  - Users may disconnect from machines carelessly, leaving them stopped

- Is VM broken?
  - If VM is sick, Linux sure won’t run!

- Is VM letting the virtual machine run?
  - CP might not be giving it resource
Linux Issues

- Is it a kernel problem within the Linux virtual machine?
  - Even Linux can have problems — OOMs (Out-Of-Memory errors), loops, or Oopses (kernel errors)

- Is a specific service (ssh, ftp, etc.) broken?
  - If that’s the service the user is trying to use, Linux will appear to be down

- Is it resource exhaustion within Linux?
  - Insufficient disk space, or suffering from OOMs (Out-Of-Memory errors) can cause some/all Linux services to wait

- Is an application or service hogging resources within the Linux virtual machine?
  - Patience is a virtue…
Penguin Problem Identification

Taking Your Penguin’s Temperature and Pulse
Linux Diagnostic Tools

- Linux provides various commands that allow you to diagnose your system:
  - `ps` (Process Status)
  - `df` (Display Filesystems)
  - `free` (memory usage display)
  - etc...

- Many of these just display data from the `/proc` filesystem
  - `/proc` is a pseudo-filesystem that offers a direct reflection of various system settings, counters, etc.
  - Much better than running control blocks in memory!
  - Files in `/proc` are accessed like any other file: `cat`, etc.
  - Writing to `/proc` files is one way to change some system settings on-the-fly
Diagnosing Network Issues

• Try to **ping** Linux from user’s machine
  • If **ping** works, network and routing are complete between user and Linux
  • Helps if you know the Linux hostname/IP address
  • Also good to know whether Linux guest normally responds (some guests don’t; some firewalls don’t let ICMP echo through)

• Try **traceroute** to Linux from user’s machine
  • **traceroute** failure at last hop before Linux implicates Linux networking
  • Helps if you know normal routing, so you know what the “last hop” is!
  • Note Linux, Windows, VM all have **traceroute**, spelled varying ways

• If Linux networking appears broken, log onto guest virtual machine, and then into Linux as **root**
  • May not be possible if local **root** login disabled (may be able to login as another user and **su** to **root**)
Diagnosing Network Issues (continued)

- Use `ifconfig` and/or `netstat -i` to examine network configuration and status
  - Bouncing connection sometimes helps (`ifconfig down/ifconfig up`)

- Use `#CP QUERY VIRTUAL NIC` to see if virtual NICs on Guest LANs are connected

- Use `#CP QUERY LAN DETAILS` to see what the Guest LAN looks like, including IP addresses assigned
  - Use `#CP QUERY LAN DETAILS lanname` if many LANs

- Use `#CP QUERY VIRTUAL CTCA` to see if virtual CTCAs are connected
Diagnosing Network Issues (continued)

- Try `cat /proc/net/arp`
  - Shows cached hardware addresses
  - If none, that tells you network isn’t very happy

- If network is broadcast-capable (QDIO), ping the Bcast address shown by `ifconfig`:
  ```
  ping -b -c 1 10.3.2.255
  WARNING: pinging broadcast address
  PING 10.3.2.255 from 10.3.2.2 : 56(84) bytes of data.
  64 bytes from 10.3.2.2: icmp_seq=0 ttl=64 time=441 usec
  ```
  - On 3270, remember to use `ping -c 1`, or ping will run forever (no “<Cntrl>C” possible on 3270!)
  - If you get more than one response from your own IP address, then there’s a duplicate on the network

- Learn to use `tcpdump` (or equivalent tool)
  - Beyond the scope of this presentation, but **very** powerful!
Diagnosing VM Troubles

- Is VM broken?
  - Try to log onto another VM userid
  - If that doesn’t work, head for the machine room!

- Is network to/from VM healthy?
  - Try to ping and traceroute VM from your PC
  - Try to ping external host from VM
  - If you can get out but not back in, it’s a routing problem external to VM

- Is the Linux virtual machine even logged on?
  - Log onto any VM userid and issue #CP QUERY USER linuxid
  - If response is linuxid NOT LOGGED ON, there’s your problem!
(Digression) VM SPOOLOLed Consoles

- VM lets you keep a copy of all console activity for a virtual machine
  - Conceptually similar to having `root` logged on using a hardcopy terminal
- Files are saved in VM system SPOOL space
- Closed on demand or automatically at system shutdown or user logoff
- **Invaluable** resource for determining abnormal virtual machine events
  - Slightly less useful for Linux, since many services do not log to console
- Still always worth checking!
  - Oopses, OOMs, some segfaults are logged to console
How To SPOOL the Console

• **CP SPOOL** command turns on SPOOLing:
  
  CP SPOOL CONSOLE START
  CP TERMINAL TIMESTAMP ON useful: timestamps all output

• Various options control default destination userid, class, and filename/filetype

• Often useful to indicate date/time SPOOL started:
  
  CP SPOOL CONSOLE START TO * NAME yyyymmdd hh:mm:ss
  • Once file is closed, file timestamp will be **close** time, so this adds useful info

• May want to centralize console collection:

  CP SPOOL CONSOLE START TO CONSAVER NAME yyyymmdd hh:mm:ss
Finding (Open) VM SPOOLed Consoles

- To determine if a running virtual machine has its console SPOOLed:
  
  ```
  #CP QUERY PRT ALL linuxid
  ```
  
  - Look for open CON file:
    
    ORIGINID FILE CLASS RECORDS CPY HOLD DATE TIME NAME TYPE
    linuxid 6216 T CON nnnnnnnn 001 NONE OPEN- 0009 name type
    
    - Existence of file is useful data point

- To close the console and send it to yourself:
  
  ```
  #CP SEND CP linuxid CLOSE CONSOLE yourid
  ```
  
  (where `yourid` is your userid)
  
  - CP SEND requires privilege class C
Processing VM SPOOLed Consoles

- Result of previous command is message:
  
  RDR FILE nnnn SENT FROM linuxid CON WAS mmmm RECS rrrr ...

- Note the “nnnn” value — that’s the SPOOL file number in your virtual reader

- Issue CMS **PEEK** command to view the file:

  PEEK nnnn (FOR *
  
  * Places you in XEDIT session, viewing file contents
  * Large files may require some time to read, need larger virtual storage
  * Note: files may span days; HCPMID6001I appears each midnight

- Use CMS **RECEIVE** command to read file to disk

  * PF9 in **PEEK**, or:

  RECEIVE nnnn fn ft fm
Finding (Closed) VM SPOOLed Consoles

- To find SPOOLed consoles for non-running virtual machines (or to find consoles from previous logons):

  ```
  #CP QUERY RDR ALL linuxid
  #CP QUERY PRT ALL linuxid
  #CP QUERY RDR ALL XFER ALL linuxid
  ```

- First two commands show files in `linuxid`‘s virtual reader or printer
- Last command shows files sent or transferred to other virtual machines

- Use CP `TRANSFER` command to transfer files to your reader:

  ```
  TRANSFER ownerid RDR nnnn *
  ```

- Then follow same procedures as for newly closed file (`PEEK, RECEIVE`)
Notes About SPOOLED Consoles

• Consoles can get very large
  • For long-running guests with significant console activity, consider closing periodically to keep files manageable
  • Might close at midnight via simple WAKEUP-based service machine
  • Using EOF option causes automatic closure every 50,000 records (may or may not be desirable, depending on how you manage the files)

• Naming consoles rationally helps a lot
  • When SPOOLing them
  • RECEIVE them as “userid yyyymmdd”, perhaps

• Vendor products exist to aid console management
When and Why Was Linux Logged Off?

Examine operator’s console to see when and why it was logged off:

**User linuxid LOGOFF AS linuxid USERS = n**
- means it was logged off “normally”, either by a user command or by Linux itself after shutdown

**User linuxid LOGOFF AS linuxid USERS = n FORCED BY vmid**
- means it was logged off through a CP FORCE command issued by vmid

**User linuxid LOGOFF AS linuxid USERS = n FORCED BY SYSTEM**
- means it was logged off due to CP “timebomb” logoff, after being left in read for (usually) 15 minutes while disconnected

Examine Linux guest console, look at bottom for more nuggets
Diagnosing VM Troubles

• If Linux virtual machine is logged on, is it stopped because it’s in a **CP READ**?
  • Issue **CP SEND CP linuxid BEGIN** to start it (harmless at worst)
  • Also use **RUNNABLE EXEC** (see Resources) to check quickly

• How did it get there?
  • Linux virtual machine force disconnected (by system issue or because user closed emulator rather than issuing **#CP DISC**) when **RUN** is **OFF**
  • Linux virtual machine reconnected and left in **CP READ** (with **RUN OFF**)
  • **CP STOP** or **CP CPU ALL STOP** issued on Linux virtual machine

• Moral:
  • *Run your Linux virtual machines with **CP SET RUN ON***!!!
Diagnosing VM Troubles

- Is VM giving the virtual machine any service?
  - CP might not be giving it resource
  - Likely if Linux virtual machine reconnect shows **RUNNING** with no keyboard response
  - If it seems normal at reconnect, hit ENTER a couple of times, look for **VM READ**, Linux **login:** prompt
  - If no read, or significant delay before login prompt, VM may not be running the virtual machine

- Basic understanding of scheduling and dispatching useful
Scheduler and Dispatcher 101

- Virtual machines must be **runnable** to do work
  - CP must be willing to **schedule** the virtual machine
  - CP must be willing to **dispatch** the virtual machine

- A virtual machine is in one of three lists:
  - **Dormant list**: virtual machine is not trying to do any work
  - **Dispatch list**: virtual machine is active and CP is allowing it to run
  - **Eligible list**: virtual machine is active, but CP is not allowing it to run
Scheduler and Dispatcher 101

• The scheduler decides whether there are enough resources to give a virtual machine some service
  • If not enough resources are available, virtual machine does not get scheduled

• The dispatcher gives virtual machines access to CPUs
  • If multiple virtual machines are active, they take turns
  • VM is very good at this — supports tens of thousands of active users with excellent response time
Dispatch Classes

Class 1 virtual machines:

• When a virtual machine is first dispatched, it is **class 1**
  • Such users are usually referred to as “Q1 users”
  • CP waits one class 1 elapsed timeslice (C1ETS) to see if it goes idle voluntarily
  • If virtual machine does not go idle within that timeslice, it is preemptively stopped from execution (“queue dropped”) — sent back to the scheduler
  • C1ETS is dynamically calculated to keep a fixed % of users in class 1
  • C1ETS should be enough for short, interactive transactions (minor CMS commands)
Dispatch Classes

Class 2 virtual machines:

- If virtual machine does not go idle voluntarily in one Class 1 Elapsed Time Slice, it enters class 2
  - Such users are usually referred to as "Q2 users"
  - Next time CP runs it, it is given 8x C1ETS
  - If virtual machine does not leave the dispatch list within that amount of time, it is queue dropped (sent back to the scheduler)
  - Such users are presumed to be running a command, but not necessarily doing something “major”
Dispatch Classes

Class 3 virtual machines:

• If virtual machine does not go idle voluntarily within class 2 multiple of C1ETS, it enters class 3
  • Such users are usually referred to as “Q3 users”
  • Next time CP runs it, it is given 6x the class 2 timeslice (=48x C1ETS)
  • If virtual machine does not leave the dispatch list within that amount of time, it is queue dropped (sent back to the scheduler)
  • Such users are presumed to be running a long-running command
Dispatch Classes

Class 0 virtual machines:

- Users with `OPTION QUICKDSP` or `SET QUICKDSP ON` bypass some of the Q1/Q2/Q3 rules
  - Still subject to queue drops, but never get held in eligible list

- Interactive users (users who hit ENTER or PF/PA keys) also get a Q0 stay
  - Such users are called “hotshot” users
  - Still subject to queue drops, but “go to the head of the line” for a brief period
  - Return to their previous queue level after Q0 stay

- Users spinning on certain locks are also in Q0
  - Such “lockshot” users presumably are preventing other users from running
Leaving the Dispatch List

- Virtual machines leave the dispatch list because they:
  - Go idle voluntarily (load a wait PSW)
  - Hold execution waiting on a CP resource (paging, DIAGNOSE I/O)
  - Leave SIE emulation due to execution of a privileged instruction (privop)

- When virtual machine leaves dispatch list, a queue drop test timer is set (300ms)
  - If virtual machine resumes activity within that period, it is reinserted into previous place in queue — not into Q1 again (unless it came from there)!
  - Linux guests without the “notimer” patch never go idle long enough to get dropped from queue!
How This Plays Out…

- **CP analyzes real resources, makes scheduling decisions**
  - If not enough resource, virtual machines are held in Eligible list (E-list)
  - Assumption is that resource will become available soon
  - If resource not freed, E-listed virtual machines never get scheduled

- **Dispatched virtual machines “should” go idle**
  - Linux tends not to go idle (without “notimer” patch)
  - Linux virtual machines thus stay runnable all the time!

- **CP considers machines with outstanding I/O to be active**
  - Linux machines usually have a pending network I/O
  - Prior to APAR VM63282, this meant Linux machines never left queue!
  - With the APAR, network I/O is ignored for queue drop purposes
How Does This Go Wrong?

- Linux machines tend to:
  - Be quite large (virtual storage size)
  - Have a working set size close or equivalent to virtual storage size
  - Stay active (rarely/never go idle)
  - Not use shared pages (DCSS)

- Aggregate real storage requirements for Linux are thus much higher than the average CMS virtual machine

- If enough large Linux virtual machines are logged on, CP notices it will overcommit real storage
  - One or more such virtual machines “lose”, are E-listed — and stay there!
How Does This Manifest?

- **System is running along fine**
  - One guest too many is started
  - Things “just stop”!

- **Remember the queue drop timer:**
  - Guests never go idle (as far as CP can tell)
  - Never cycle out to scheduler, so E-listed guests stay there!
Detection

- **CP INDICATE QUEUES EXPANDED** command shows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Priority</th>
<th>Start/End</th>
<th>Deadline</th>
<th>Status</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINUX902</td>
<td>Q3</td>
<td>PS</td>
<td>00013577/00013567</td>
<td>....</td>
<td>-232.0</td>
<td>A00</td>
</tr>
<tr>
<td>LINUX901</td>
<td>Q3</td>
<td>PS</td>
<td>00030109/00030099</td>
<td>....</td>
<td>-231.7</td>
<td>A00</td>
</tr>
<tr>
<td>VSCS</td>
<td>Q1</td>
<td>R</td>
<td>00000128/00000106</td>
<td>.I..</td>
<td>-208.7</td>
<td>A00</td>
</tr>
<tr>
<td>LINUX201</td>
<td>Q3</td>
<td>PS</td>
<td>00031166/00029348</td>
<td>..D.</td>
<td>-1.072</td>
<td>A00</td>
</tr>
<tr>
<td>LINUX401</td>
<td>Q3</td>
<td>PS</td>
<td>00030214/00028874</td>
<td>..D.</td>
<td>-1.010</td>
<td>A00</td>
</tr>
<tr>
<td>VMLINUX3</td>
<td>Q3</td>
<td>IO</td>
<td>00052962/00051162</td>
<td>....</td>
<td>-.9398</td>
<td>A00</td>
</tr>
<tr>
<td>VMLINUX3 MP01</td>
<td>Q3</td>
<td>PS</td>
<td>00000000/00000000</td>
<td>....</td>
<td>.0612</td>
<td>A00</td>
</tr>
<tr>
<td>LINUX123</td>
<td>E3</td>
<td>R</td>
<td>00177823/00196608</td>
<td>....</td>
<td>5255.</td>
<td>A0</td>
</tr>
</tbody>
</table>

- **HELP INDICATE QUEUES** shows meaning of output
- CP privilege class E required
- **Note**: “deadline time” (sixth column) sometimes very large — and very bogus
- Virtual machine **LINUX123** is not going anywhere anytime soon…
Remediation

- Buy more storage ($10K/GB — cheap!)
- Make sure “notimer” patch is installed
  - Obviously only meaningful if guests are nominally idle
  - Remember cron and other things may wake them up anyway
- Log off some guests
- Tune guest storage sizes
  - Linux uses “extra” storage for file buffers
  - Back off virtual storage size until guests swap, then add a bit more (or not)
Diagnosing Kernel Problems

• Log onto Linux guest to see if it’s even alive:
  • Hit ENTER, look for `VM READ`, login prompt
  • No `VM READ` means Linux is “hung” (looping or otherwise busted)
  • No login prompt could just mean `login` isn’t running (again, it helps to know what normal behavior is!)
  • Look at SPOOLed console for Oops messages

• “What’s an Oops?”
  • A system ABEND, in VM terms: a kernel failure
  • Like VM, may or may not leave the system in a usable state
  • Examples: the kernel tries to access an invalid memory location
  • Doesn't necessarily indicate code bug — faulty hardware can cause an Oops
Basic Oops Analysis

- Utility **ksymoops** maps addresses in Oops output to kernel modules
  - Uses system map file, usually found in `/boot`

- Oops output used by **ksymoops** is in a file
  - Usually found in `/var/log/messages`
  - If `syslogd` not running, extract with `dmesg` utility (`dmesg > oops.log`)
  - If Linux not even that alive, cut & paste from console log, or type it back in!

- **Note**: If cascading Ooopses, only first is usually relevant
Diagnosing Kernel Loops

• Use \#CP INDICATE USER linuxid EXPANDED to watch guest CPU time
  • If increasing rapidly, guest may be looping (could just be busy, though)
  • Also note I/O counts, look for massive I/O load

• If loop suspected, log onto guest, use \texttt{CP TRACE}:
  • \#CP TRACE INST RUN NOTERM PRINT
  • Let it run for a bit; monitor with \#CP QUERY PRT * ALL
  • Then issue \#CP TRACE END and \#CP CLOSE PRT *, RECEIVE file
  • Analyze for repeated hits/patterns (or ask vendor to)
Diagnosing Broken Linux Services

- Use `ps aux` to show what services are running, pipe through `grep` to find target:
  
  ```
  # ps aux | grep ssh
  ```

  - Finds any lines that mention “ssh” (may find the `grep` itself, too)

- Restart service that’s not up and should be
  
  - Perhaps restart it anyway if it claims to be up but isn’t responding!
Diagnosing Broken Linux Services

- Look at system log files
  - `/var/log/messages` may contain something interesting

- The `dmesg` utility also shows recent kernel messages
  - Looks at “kernel ring buffer” — sort of like a trace table, but just messages

- Look at logs for service in question
  - Alas, location not predictable — try `/var/log/servicename` (prescribed by Linux Filesystem Hierarchy Standard, but not all applications behave)
  - Beware: Linux time and VM time may differ (timezone, drift)
  - Look in application directories, if you know what they are
  - Failing that, read the documentation (gasp) or code for the application

- Note: default logging levels often omit useful information
  - May need to set environment variable for next occurrence of problem
Diagnosing Resource Exhaustion

- If Linux runs short on a resource, results “may be unpredictable”
  - Well-behaved applications will fail in graceful ways
  - Severe and/or rapid resource depletion may prevent this

- Nothing unique about Linux resources that can be depleted:
  - Disk space
  - Memory
  - Page (swap) space
  - CPU
Diagnosing Disk Space Exhaustion

- "df" (Display Filesystems) shows filesystem status
  
  ```
  # df -a -h
  
  Filesystem   Size  Used  Avail  Use%  Mounted on
  none         592M  94M   464M   17%  /
  none         0     0     0     -   /proc
  none         0     0     0     -   /dev/pts
  /dev/dasd/0000/part1 485M  17M  468M   4%  /tmp
  
  Most interesting part is "Use%"
  - Filesystems above 90% may be suspect due to temporary file usage
  - Again, useful to know "normal" usage levels
Diagnosing Memory Exhaustion

- Some Linuxes take OOM (Out-Of-Memory) errors when insufficient “real” (virtual) memory is available
  - Applications can get OOMs
  - The kernel can get an OOM too (much more serious, usually game over!)

- OOMs are reported on Linux console:
  Out of Memory: Killed process *(processname)* (application OOM)
  Out of memory and no killable processes (kernel OOM)

- The *processname* is the name you would see from **ps**
  - May or may not be the process that was really using all the memory (may not be a real memory hog, just lots of small processes!)

- OOMs were removed as of kernel level 2.4.23
  - Now applications get individual memory allocation failures, must handle
Diagnosing Memory Exhaustion

- **The `free` command displays system memory use:**
  ```
  # free -t
  total     used     free    shared  buffers     cached
  Mem:  191092  185160   5932       0      13032     80548
  -/+ buffers/cache:  91580  99512
  Swap:  197176    2920  194256
  Total: 388268  188092  200176
  ```

- **“-/+ buffers/cache” line most interesting: shows usage without taking file buffers and cache into account**
  - Those pages are reclaimable for system use (DPA, in VM terms)
  - If Swap space is mostly/entirely in use, expect application OOMs/hangs!
Diagnosing CPU Exhaustion

- As on many operating systems, a single application can grab enough CPU to slow Linux
  - Mechanisms exist to control this, of course, but are not enabled by default

- `top` command is the “performance monitor” tool
  - `sar` is a popular free alternative (see `Resources`)
  - Vendor tools exist, of course (RMF PM, Velocity, Perfman — see `Resources`)
  - Beware: `top` can take a long time to start up on a loaded system!

- `uptime` displays 1-, 5-, 15-minute CPU load averages
  - Look for rising trend to show recent problem
  - Values > 1.00 mean CPU is fully loaded (work is waiting)
  - Cannot tell whether rising values mean this Linux or another machine is using more CPU — could mean higher fraction of less available CPU
top Command Output

4:26pm  up 5 days,  7:10,  2 users,  load average: 1.00, 1.00, 1.00
82 processes: 80 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  0.8% user, 14.0% system,  0.0% nice,  85.1% idle
Mem:  191092K av,  185808K used,  5284K free,  OK shrd,  12976K buff
Swap: 197176K av,  2920K used, 194256K free 80288K cached

PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
6250 root 17  0  1060  1060  844 R  5.9  0.5  0:01 top
6142 root  9  0  2320  2320 1828 S  0.3  1.2  0:02 sshd
 1 root  9  0   556   540  492 S  0.0  0.2  0:02 init
 2 root  9  0   0   0   0 SW  0.0  0.0  0:00 kmcheck
 3 root  9  0   0   0   0 SW  0.0  0.0  0:00 keventd

etc...

• Note that the, um, top command is top itself!
  • Shows up where it does because it’s sampling all current resources
  • Look at other candidates, note “heavy hitters”
  • “top d 5” will auto-refresh every five seconds, show some trends

• See man page to interpret values, especially STAT
  • Note “0.0% nice” — negative value would mean some tasks have priority
Other Performance Measurements

• **Look at** `/proc/loadavg`
  • Fourth value is #processors/#processes running (e.g., `2/81`)
  • Fifth value is number of total processes started since system booted
  • If fifth value is changing rapidly, something is going on

• **SNMP provides some data, depending on settings**
  • Must be enabled, of course, and SNMP collector operating somewhere!
  • Make sure **not** to leave default public/private strings (passwords) in place (sounds obvious, but empirical evidence shows far too many folks do)!

• **Linux I/O statistics may be useful**
  • Enable by `echo set on > /proc/dasd/statistics`
  • Must be enabled **before** problem, of course, to be useful!
  • Statistics generated in `/proc/dasd/statistics`
Other Performance Measurements

- `cat /proc/chandev` shows current state of devices
  - Useful if other evidence suggests a problem with a device

- Learn CP commands that tell about the virtual machine:
  - `QUERY VIRTUAL ALL` (lots of output!)
  - `QUERY VIRTUAL DASD` (show all virtual DASD)
  - `QUERY VIRTUAL xxxx` (show a specific device)
  - `QUERY MDISK` (show ownership of a virtual DASD)

- z/VM Performance Toolkit can provide some external performance measurement
  - Won’t tell you what’s going on inside Linux, but at least let you profile it

- `iostat` (partner to `sar`) also useful for I/O monitoring
VM Monitor Data

- z/VM generates monitor data on demand
  - Highly granular, very efficient mechanism

- Linux for zSeries can, too, as of recent IBM code drop
  - Still few (no?) tools to exploit it well
  - Stay tuned…
Penguin Forensics

Recording Evidence Before Burying the Body
First Failure Data Capture

- IBM has long promoted **First Failure Data Capture**:  
  - Collecting useful debugging information when a problem first occurs  
  - “Reboot and if it happens again let us know” is **not** FFDC!  
  - VM, MVS, AIX, DB2, even Tivoli push FFDC  
  - Even Windows XP Error Reporting can be considered FFDC

- **As Linux matures, FFDC concepts are seeping in**  
  - Logging, trace tables, memory leak/overlay traps, more dump capabilities…  
  - Still mostly not standard features, however — optional installs
Log Levels

- syslogd (syslog daemon) collects and writes messages from various services, applications
  - Of course, it has to be running to be useful!
  - Can centralize messages from multiple systems to single host

- Level of messages to be logged is configurable
  - Understanding logging levels for your services/applications is essential to ensuring FFDC

- Standard Linux syslogd isn’t very smart/flexible
  - Insufficiently granular in many cases
  - Uses UDP, which means messages can get lost due to network congestion
  - Alternatives exist, such as syslog-ng (see www.balabit.com)
Cores

- Traditional *ix dumps were “core files”
  - Created when applications did something blatantly illegal
  - Created in current working directory, either `core` or `core.pid` (controlled by `/proc/sys/kernel/core_uses_pid`)

- Modern distributions typically ship with cores disabled
  - Most Linux users wouldn’t know what to do with these large files!
  - May contain sensitive data from currently running application

- bash `ulimit -c size` enables for current login
  - `ulimit -c unlimited` means “dump everything”
  - `ulimit -c` displays current setting (value > 0 means enabled)
  - See `man bash` for details
As a VMer, I want to VMDUMP a wonky virtual machine:

```
#CP VMDUMP 0-END TO MAINT
```

- But none of the Linux dump tools want to read VMDUMPs!
- Still, the VM Dump Tool is programmable, can certainly handle Linux dumps

**LKCD (lcrash) — Linux Kernel Crash Dump**
- Must be installed *before* the problem occurs
- lcrash is the “IPCS” tool that helps you analyze the dump

**Standalone dump facilities available for zSeries Linux**
- Analyze dumps with lcrash
Linux Debugging Tools

- Kernel breakpoint tools:
  - DProbes (Dynamic KProbes):

- Kernel event (trace table) logging:
  - LTT (Linux Trace Toolkit): [wwwopersys.com/LTT/index.html](http://wwwopersys.com/LTT/index.html)
  - Strace (System call Trace): [www.liacs.nl/~wichert/strace](http://www.liacs.nl/~wichert/strace)

- Memory debuggers:
  - NJAMD (Not Just Another Malloc Debugger): [fscked.org/proj/njamd.shtml](http://fscked.org/proj/njamd.shtml)

- General debugger:
Learning Linux Debugging Techniques

• Zapping Linux bugs:
  www.eservercomputing.com/mainframe/articles/index.asp?id=675

• Mastering Linux debugging techniques:
FFDC: What To Save

• Linux data
  • System log files
  • Application log files
  • Any core files
  • Application configuration files

• VM data
  • VM console logs
  • CP command output
  • Trace files
  • Monitor data
  • Performance monitor reports
  • Any dumps
  • Guest directory entries
Conclusion
Summary

• To the VMer, Linux is obscure and opaque
• To the Linux expert, VM is the same!
• To provide proper support, learn to use the tools
  • Both VMers and Linux folks can learn from each other
• As always, use the community
  • LINUX-390@marist.edu: zSeries Linux mailing list
  • VMESA-L@listserv.uark.edu: z/VM mailing list

➤ z/VM and Linux — even better together!
Resources

- **Velocity Software (ESALPS)**: [www.velocitysoftware.com](http://www.velocitysoftware.com)
- **Perfman**: [www.perfman.com](http://www.perfman.com)
- **sar** (part of sysstat): [freshmeat.net/projects/sysstat/](http://freshmeat.net/projects/sysstat/)
- **ksymoops**: [www.gnu.org/directory/devel/debug/ksymoops.html](http://www.gnu.org/directory/devel/debug/ksymoops.html)
- **RUNNABLE EXEC**:
  - Display virtual machine status; email me for a copy
Contact Information and Credits

Contact Info
Phil Smith III
703.476.4511 (direct)
800.546.4878 (company)
psmith@levanta.com
www.levanta.com

Thanks To…
Alex “Puffin” deVries
Scott Loveland
Neale Ferguson
Len Reed
Christopher Neufeld
Rod Stewart