Linux Performance Tools

SHARE 101 Technical Conference
August 10-15, 2003
Washington, DC
Session 9360

Oliver Benke
IBM Germany Lab
Email: benke@de.ibm.com

eServer Systems Management
Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM*  RACF*  DB2*
the IBM logo*  RMF  WebSphere*
OS/390*  zSeries  Domino
Parallel Sysplex*  Tivoli*  e business(logo)*
MVS  CICS  e(logo)server
z/OS*  IMS  e(logo)business

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

LINUX is a registered trademark of Linus Torvalds

* All other products may be trademarks or registered trademarks of their respective companies.
Agenda

1. Performance Management, zSeries Architecture, …
   Base concepts

2. Performance Tools with Usage Examples
Some basics

- Performance Management
- Resource Sharing, Overcommitted Resources, Virtualization
  - CPU Resources in a virtualized environment
- zSeries Mainframes: what’s different?
- Performance base concepts
  - Load Average
  - System/User CPU Consumption
- The /proc filesystem
Performance Management

- Online Monitoring, Problem drill-down; 1 day history (or 3 days for the weekend) needed
  - May be automated, using asynchronous events
  - Online performance data may be used by autonomic software components, like VMRM and IRD on zSeries

- Long-term monitoring and capacity planning
  - Understand whether growth of resource consumption is bug driven or business driven
  - Estimate by when you need to invest in new hardware
Mainframe Linux: Any Advantages?

- Leading-edge Virtualization
  - z/VM or LPAR virtualization technologies
  - Possibility to virtualize and share CPUs, Channels (=I/O) and probably Memory (z/VM only)

- Advanced Resource Sharing
  - Workload Management using *Intelligent Resource Director* (IRD) or *z/VM VMRM*

- Optimized for Server Workloads
  - Reliability – Availability – Scalability
  - Horizontal and vertical scaling
  - High I/O performance, high memory bandwidth

- Internal Networking Facilities
  - Memory-based networking using HiperSockets (LPAR) or GuestLAN (z/VM)

- Server consolidation
zSeries Resource Sharing Overview

Networking between LPARs on same CEC: HiperSockets

Workload Management - WLM / IRD

WAS
Linux

... ... ...

WAS
Linux

z/VM Virtualization Layer

Networking between guests: z/VM guest LAN

z/VM Virtualization Layer

IMS
z/OS

mySAP
Linux

DB2
z/OS

LPAR Virtualization Layer with PR/SM Hypervisor

Processor, Memory, Channels
Virtual Resources

β ... can be shared between several instances which do not even know about each other, like several companies hosted by the same data center
β ... can be over-committed to a certain degree. However, this does not mean there are no limits, performance of over-committed systems can be very unpleasant. The useful capacity limit of virtual resources depends on the given workload mix you are running
β ... can be created “out of nothing”, so as an example, you may go create a whole network infrastructure with router, switches, links, and servers – all virtual, all inside z/VM. No cabling, no hardware configuration changes, pure software. Virtual test floor.
Resource Sharing and Virtualization: Effects

- No idle resources if any virtual server has useful work to be executed
  
  This way, a mainframe can drive most resources to their capacity limits without penalties to the response times of critical business workloads.

- Different workload may compete for resources with each other, so performance tuning more challenging

- For severe over-commitment of resources, overall performance may degrade if no proper workload management and tuning is in place (like thrashing effects).

- Re-configuration of virtual data center very flexible; z/VM configuration changes instead of network cabling and hardware changes.
IRD

- Linux servers running in non-IFL LPARs can have their LPAR weights automatically managed by IRD. The policy is then defined in z/OS WLM.
- As WLM/IRD does not look inside the Linux LPAR, the LPARs are managed as a whole – in essence, the Linux LPAR is treated as one business application.
  In reality, this happens to be the case for normal Linux applications, as it follows the philosophy of client-server and dedicated servers.
  Linux on the mainframe allows for multiple dedicated virtual servers on one physical server.
Internal Virtual Networks

- HiperSockets: z900/z800 Hardware, can be used to communicate between different LPARs running z/VM, z/OS, Linux for zSeries, Linux under z/VM
- For TCP/IP socket-based applications, this is transparent.
- Alternative under z/VM 4.2 and higher: Guest LAN - HiperSockets simulated in software, useful for communication of several guests running inside the same z/VM
- Connect a “virtual network” (Guest LAN, HiperSockets) with a Linux router to the outside world; of course, this router could be a “hot spot”, so carefully watch it
- Older z/VM technologies: IUCV, vCTC
User-mode and kernel-mode CPU time consumption

- If UserModeTime / KernelModeTime is relatively high and IdleTimePercentage is near zero, this can be an indicator that the underlying z/VM has a contention for CPU.
- This happens because if Linux is constrained for CPU, it may only be able to execute the most important kernel daemons and at the time it would probably start doing some useful work, the CPU is taken away.
- If KernelModeTime is relatively high, the system overhead is high, and this is usually a bad sign.
- However, as always, it depends; there are some workloads which simply need high amount of KernelModeTime CPU, and for those workloads, high KernelModeTime values are just normal.
CPU Usage: Variable cost and Fixed cost

IdleTime

UserModeTime = Variable Cost Part 1

KernelModeTime dependent on UserModeTime = Variable Cost Part 2

Base KernelModeTime = fix cost
Resource Sharing of CPU resources: the zSeries way

---

**zSeries HW: N-way SMP**

<table>
<thead>
<tr>
<th>LPAR 1</th>
<th>LPAR 2</th>
<th>LPAR Hypervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined Capacity (Weighting, Capping, Dedicated, # logical CPUs, …) and Actual Capacity</td>
<td>Defined Capacity and Actual Capacity</td>
<td>*PHYSICAL Dispatch Time = Overhead for LPAR virtualization</td>
</tr>
<tr>
<td><strong>z/VM, even more flexible virtualization layer than zSeries LPAR</strong></td>
<td>Linux for zSeries or z/OS</td>
<td></td>
</tr>
<tr>
<td>LX1</td>
<td>LX2</td>
<td>LX3</td>
</tr>
<tr>
<td>User</td>
<td>K</td>
<td>User</td>
</tr>
</tbody>
</table>

---

Shared Memory; CPU, I/O “double-shared”

Shared CPU, Shared I/O
Idle time

In the last picture, idle is not shown. Depending on whether CPU resources are dedicated or not, idle time cannot be attributed to single operating systems, as the zSeries box is only idle if and only if all of the running operating systems are idle concurrently. So for a well used system, you may not see any idle time.

However, if a CPU is dedicated to one operating system, it is used completely by this operating system, so it would make sense to charge this idle time to the operating system which has the dedicated resources.
Load Average

- Average number of processes in the "run" queue
- A runnable process is one that is ready to consume CPU resources right now; a process waiting for I/O is not runnable
- A high load average value (in relation to the number of physical processors) is an indicator for latent demand for CPU. The processes waiting on the run queue are not waiting for I/O or other processes, they are waiting for CPU and they are otherwise ready to run.
- Load averages are available in various places; you may obtain it by typing
  
  \[ \text{cat} /\text{proc/loadavg} \]
  
  or using program like \text{xload}
Linux Page and Buffer Cache

The page cache contains pages of memory mapped files - page I/O related system calls like `generic_file_read`

It usually contains unnecessary files which can be freed, and the kernel actually discards those pages if it runs out of free memory.

Another important Linux kernel data structure is the so-called Buffer Cache which contains pages read from or written to physical devices like DASDs (block I/O related system calls)

Linux rarely has free space; everything not used is allocated for Page Cache and Buffer Cache, so even if Linux does not really need it all, it uses all available memory up to the last few percent up to now.

On Intel Linux or for Linux running in a LPAR, the page cache is always useful as the memory would have been wasted otherwise. But running under z/VM, it may cost valuable z/VM memory, leading to z/VM page activity.
Timer Interrupt and Jiffies

- Derived from PC timer interrupt (100 Hz)
- Every time a timer interrupt occurs (100 times per second), the jiffies variable is incremented by one; that’s one timer tick
- CPU usage is accounted on in jiffies
- If a process is running at the time the timer interrupt occurs, its CPU usage counter is incremented
- Accuracy (10 msec) might be enhanced in future Linux versions
- Jiffie-based performance measurement is wrong if running under z/VM
- Solution: correlate information from LPAR Hypervisor, z/VM and Linux
- On demand timer patch: for an idle Linux image running under z/VM, CPU resources are used up mainly for generating the jiffies. With this patch, jiffies are generated on demand.
Linux process memory: basic terms

- **SIZE**: size of the address space seen by the process, virtual size
- **RSS**: Resident Set Size
  actual amount of memory that the process is using in RAM
- **SHARE**: portion of the RSS that is shared with other processes, such as shared libraries
Processes and Threads

- In contrast to some commercial UNIX implementations, in Linux a thread is pretty much the same as a process, it just does not have an own address space.
- Even without real thread support, the Linux implementation is competitive; Linux process handling is very fast and offers great scalability on zSeries hardware.
- For the scheduler, a Posix thread is almost like a process.
- In the /proc filesystem (see below), there is no difference between a process and a thread; so if you are monitoring your system, your threads might appear like processes on first sight.
- As an alternative, user-space thread libraries are available today.
The /proc filesystem

- Virtual filesystem
- One of the interfaces between kernel space and user space; if the user gives a command like `cat /proc/stat`
  - the kernel executes some function to generate the needed "virtual file"
- Parts of the /proc filesystem are human readable
- Most performance measurement tools for Linux are based on /proc filesystem
/proc/stat Example

```bash
benke@lnxrmf:~$ more /proc/stat
cpu 220494 274647 1095518 701390830
cpu0 66125 77458 298850 233884730
cpu1 58940 102875 335467 233829881
cpu2 95429 94314 461201 233676219
page 17421389 12618473
swap 19506 22061
intr 0
disk_io: (94,0):(2894594,1601804,34839816,1292790,25236984)
ctxt 142638745
btime 1057071413
--More--(0x)
```
Linux Performance Tools

- Standard UNIX Tools for performance-related problem analysis: `top`, `ps`, `time`, `netstat`, `free`, `vmstat`, `iostat`, `strace`, `df`, `du`, `ping`, `traceroute`
- `sysstat` package (sar, sadc) for long-term data collection
- BSD accounting
- NET-SNMP
- SBLIM
- RMF for Linux, VM Performance Toolkit

... lots of useful point solutions for performance management
Advantages of good old UNIX standard tools

- Can be used in own (shell) programs, in order to automate systems management (considered dangerous by some installations)
- Very flexible
- Available on every UNIX system (but one needs to be careful if it should run on both e.g. AIX as well as on Linux)
- Usually quite fast and low impact on system performance
- Nice for people who like to code
- In any case, at least for problem drill-down analysis, you should know about the standard UNIX tools

Hard to learn, but everything is explained in man pages (well, almost everything ;-)}
top

- Nice option: in interactive mode, enter <f>, <u>, <return> to see what the process is waiting for
### ps - report process status

- **common set of parameters:**
  - `ps aux`
- **single out a user:**
  - `ps u --User apache`

```
# ps aux | more

USER   PID   %CPU  %MEM   VSS  RSS  TTY STAT START TIME COMMAND
root   1     0.0   0.1   1536 160   ?    S Jan22 0:12 init
root   2     0.0   0.0     0     0   ?    S Jan22 0:00 [kmccheck]
root   3     0.0   0.0     0     0   ?    S Jan22 0:00 [keventd]
root   4     0.0   0.0     0     0   ?    S Jan22 0:22 [kswapd]
root   5     0.0   0.0     0     0   ?    S Jan22 0:00 [kreclaimd]
root   6     0.0   0.0     0     0   ?    S Jan22 0:00 [kflush]
root   7     0.0   0.0     0     0   ?    S Jan22 1:05 [kupdated]
root  63     0.0   0.0     0     0   ?    S Jan22 0:00 [mdrecoveryd]
root  248     0.0   0.0     0     0   ?    S Jan22 0:00 [kventd]
root  310     0.0   0.2   1732 292   ?    S Jan22 0:12 syslogd -m 0
root  315     0.0   0.6   2088 768   ?    S Jan22 0:00 klogd -2
rpc    325     0.0   0.0   1732 120   ?    S Jan22 0:00 portmap
rpcuser 338     0.0   0.1   1844 140   ?    S Jan22 0:00 rpc.statd
root  385     0.0   0.6   3180 800   ?    S Jan22 0:00 /usr/sbin/sshd
root  401     0.0   0.4   2876 512   ?    S Jan22 0:00 xinetd
```
Show running processes as a tree
free

Give free memory; important is the second line, as buffer/cache memory is not really needed by Linux

```
[root@lnxbenk1 /root]# free
            total  used  free  shared  buffers  cached
Mem:    118092  116872  1220     0   4148  66124
-/+ buffers/cache:    46600   71492
Swap:       0      0     0     0
```
/proc/meminfo

- **MemShared**: 0 (available for compatibility reasons only)
- **SwapCached**: memory which is both in swap space (=on disk) as well as in main memory (=usable); it’s easier to page memory from the SwapCache out, as there is already a copy in the swap file
- **Active**: memory which was recently used
- **Buffers, Cached**: memory in buffers and in cache
- **Mem, Swap**: physical memory, swap space
mpstat

- **mpstat** is used to display CPU related statistics.
- **mpstat 0**: display statistics since system startup (IPL)
- **mpstat N**: display statistics with N second interval time

Btw the high %system values between 01:18:19 PM and 01:19:09 PM are no problem. I simply executed a file-system stress test, so there was lots of I/O and the operating system had lots to do…
vmstat

- Gives information about memory, swap usage, I/O activity and CPU usage. It really does a lot more than reporting virtual memory statistics ...
- Please note that the first line contains a summary line since system start (IPL).
- First parameter: interval time, second parameter: number of parameters.
vmstat fields explained

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>procs</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td>memory</td>
<td>swpd</td>
</tr>
<tr>
<td></td>
<td>free</td>
</tr>
<tr>
<td></td>
<td>buff</td>
</tr>
<tr>
<td></td>
<td>cache</td>
</tr>
<tr>
<td>swap</td>
<td>si</td>
</tr>
<tr>
<td></td>
<td>so</td>
</tr>
<tr>
<td>io</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>bo</td>
</tr>
<tr>
<td>system</td>
<td>in</td>
</tr>
<tr>
<td></td>
<td>cs</td>
</tr>
<tr>
<td>cpu</td>
<td>us</td>
</tr>
<tr>
<td></td>
<td>sy</td>
</tr>
<tr>
<td></td>
<td>id</td>
</tr>
</tbody>
</table>
iostat

`iostat` is used to report CPU statistics and disk I/O statistics. The first parameter is the interval time in seconds, the second is the number of intervals to run, so “`iostat 2 3`” gives 3 samples with 2 seconds interval.

As for `vmstat`, the first line reflects the summary of statistics since system IPL.

- **tps**: number of I/O requests to the device per seconds
- **Blk_read/s**: number of blocks (of indeterminate size) read per second
- **Blk_wrttn/s**: number of blocks written per second
/proc/dasd/statistics

- Only available in Linux for zSeries, kernel version 2.4
- Gathering of this information can be switched on and off, as it causes some overhead:
  - `echo set on > /proc/dasd/statistics`
  - `echo set off > /proc/dasd/statistics`
- Used in rmfpms to calculate the following metrics:
  - dasd io average response time per request (in msec)
  - dasd io average response time per sector (in msec)
  - dasd io requests per second
Displaying Network Interface Statistics Overview

Example use of the `netstat` command line tool:

```
benke@lnxrnf：“”> netstat -i
Kernel Interface table
  Iface  MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
  eth0  1492  0 1311984  0  0  0   684851  0  0  0 MRU
  lo   16436  0   1224  0  0  0    1224  0  0  0 LRU
benke@lnxrnf：“”>
```

- RX-OK, TX-OK: number of packets received/ transmitted without error
- RX-ERR, TX-ERR: transfer with error
- RX-DRP, TX-DRP: dropped packets
- RX-OVR, TX-OVR: packets dropped because of overrun conditions
- MTU, Met field: current MTU and Metric settings for this interface
  (Metric is used by the Routing Information Protocol RIP; MTU, Maximum Transmission Unit: max number of bytes transferred in one packet)
- Flg: status, properties of the interface (R: running, U: up, …)
- Iface: Name of the interface
Display Network Protocol Statistics

In contrast to “netstat –i”, which reports on network device level, “netstat –s” reports on network protocol level.

One advantage of this performance report is that it is less cryptic ;-) although there is a whole bunch on conditions gathered especially for the very important TCP protocol (not displayed here).
ICMP Exploiter Applications

- ICMP: Internet Control Message Protocol
- ping and traceroute are making use of the ICMP protocol in order to identify network problems.
- ping measures round-trip times between two hosts.
- traceroute – although a widely used UNIX command – is a hack, and so it does not always tell the truth. It tries to trace the way of packets through the network by sending around messages with short time to live (TTL) values.
- use “traceroute –q N” with N about 10 or higher if you want traceroute to sent more packets, in order to enhance precision of the reported numbers
ping and traceroute examples

benke@lnxrmf:~ $ ping www.uni-karlsruhe.de
PING www-uka.rz.uni-karlsruhe.de (129.13.64.69) from 9.152.81.228 : 56(84) bytes of data,
64 bytes from www-uka.rz.uni-karlsruhe.de (129.13.64.69) : icmp_seq=1 ttl=234 time=15.1 ms
64 bytes from www-uka.rz.uni-karlsruhe.de (129.13.64.69) : icmp_seq=2 ttl=234 time=14.0 ms
64 bytes from www-uka.rz.uni-karlsruhe.de (129.13.64.69) : icmp_seq=3 ttl=234 time=14.5 ms

--- www-uka.rz.uni-karlsruhe.de ping statistics ---
3 packets transmitted, 3 received, 0% loss. time 2034ms
rtt min/avg/max/mdev = 14.083/14.602/15.161/0.462 ms
benke@lnxrmf:~ $ /usr/sbin/traceroute www.uni-karlsruhe.de
traceroute to www.uni-karlsruhe.de (129.13.64.69), 30 hops max, 40 byte packets
  1 bp180002.boeblingen.de.ibm.com (9.152.80.2) 0.622 ms 0.583 ms 0.545 ms
  2 s2-60.boeblingen.de.ibm.com (9.152.94.9) 0.733 ms 1.135 ms 1.104 ms
  3 c1-15.boeblingen.de.ibm.com (9.152.120.41) 1.171 ms 1.145 ms 1.117 ms
  4 r2-18.boeblingen.de.ibm.com (9.152.120.58) 1.082 ms 1.055 ms 1.028 ms
  5 9.152.121.62 1.248 ms 0.976 ms 0.962 ms
  6 dei-bc6509-r-b-vl13.megacenter.de.ibm.com (9.149.250.13) 1.048 ms dei-bc6509-r-a-vl11.megacenter.de.ibm.com (9.149.250.5) 1.029 ms dei-bc6509-r-b-vl13.megacenter.de.ibm.com (9.149.250.13) 1.228 ms
  7 9.149.250.50 0.900 ms 9.149.250.58 0.864 ms 9.149.250.50 0.811 ms
  8 9.64.130.40 1.255 ms 1.216 ms 1.180 ms
  9 194.196.100.91 1.595 ms 1.581 ms 2.082 ms
 10 ehn1br2-2-0-1-1.eh.de.prsvr.net (152.158.3.138) 2.006 ms 2.410 ms 2.384 ms
 11 fran2br2.fr.de.prsvr.net (152.158.92.2) 17.437 ms 17.940 ms 18.072 ms
 12 dcix1nap-1-0-0.de.ip.att.net (152.158.33.237) 8.271 ms 8.210 ms 8.178 ms
 13 decix.Frankfurt1.belwue.de (80.81.192.175) 9.342 ms 9.305 ms 9.260 ms
 15 Stuttgart1.belwue.de (129.143.1.33) 13.873 ms 13.845 ms 13.817 ms
 16 Karlsruhe1.BelWue.DE (129.143.1.4) 15.466 ms 15.438 ms 15.412 ms
 17 BelWue-GU.uni-Karlsruhe.de (129.143.166.130) 14.446 ms 14.408 ms 14.310 ms
 18 www.uka.rz.uni-karlsruhe.de (129.13.64.69) 14.114 ms 14.274 ms 14.234 ms
Filesystem Usage

The `-h` option stands for human readable. Without `-h`, reported numbers are bytes …

The `df` command gives you a list of all mounted filesystems, corresponding to /dev/dasdxx devices.

Using `du` you can see the amount of disk storage used in various directories. If you want a sum, use `-s` option.
Inode Utilization

- In UNIX, an inode is a structure containing meta data about files and directories.
- The number of inodes is limited, can be changed at filesystem creation time.
- If you are running out of inodes, you can not store anything more on this filesystem.
- Check with "df -i" command:

```
benke@tux390:/projects/home/benke > df -i
Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/dasdb1 601312 59034 542278 10% /
/dev/dasdc1 300960 63886 237074 21% /projects
```
time

Find out how many CPU resources a command is using.

Example:
$ > time make dep
...
  72.52user 8.87system 2:03.72elapsed 65%CPU
(0avgtext+0avgdata 0maxresident)k 0inputs+0outputs
(131158major+106391minor) pagefaults 0swaps
$ >

elapsed: real time elapsed
user: time this command (and its children) have spent in user space
sys: time spent in kernel space
System Call Trace

- One of the commands more powerful than what we have for traditional mainframe operating systems, comes in very handy …
- strace allows to see the system calls a process is currently executing, so for example if you have the gut feeling a process with process ID PID 4711 is looping, you can execute 
  \[ \text{strace} -p 4711 \]
  in one terminal window; if it is a server process and it is not using any system calls but runs the CPU to 100% utilization, this is very suspicious, so you may think about killing this process
- strace is also useful as it can show you the sequence of system calls your favorite application is executing, so it may help you finding out how to tune the application. For example, good old UNIX philosophy is to search for files in various places if they are not where expected. This is goodness as it works, but badness as it costs some performance, so it is better to provide links to the files if this happens over and over again.
strace Example

```bash
benke@lnxrmf:~$ strace rmfpms/bin/rmfpms restart 2> straceoutput
Stopping performance gatherer backends ...
done!
Starting performance gatherer backends ...
 DRSRV: RMF-DRS-Server/Linux-Beta (Jul 28 2003) started.
 DRSRV: Functionality Level=1.950
 DRSRV: Reading exceptions from gpmexsys.ini and gpmexusr.ini.
 DRSRV: Server will now run as a daemon process.
done!
benke@lnxrmf:~$ more straceoutput
```
List open files (**lsof**)
**lsof explained**

- For UNIX, everything is a file. Directories, inter-process communication structures (like pipes), network sockets and regular files are all files. "lsof" can list all file usages.

- Some useful usage examples of lsof:
  
  List all files by processes with name “gpmddsrv”:
  ```
  lsof –c gpmddsrv
  ```

  List all TCP/IP v4 network connections to host “tux390.boeblingen.de.ibm.com”:
  ```
  lsof –i4tcp@tux390.boeblingen.de.ibm.com
  ```

  List all files using /var/log:
  ```
  lsof –t /var/log
  ```
Lock Contention

- /var/lock is the standard location to place lock files, so have a look what’s in it.
- The “ipcs” gives a summary on shared memory segments, semaphores and message queues the calling user has read access to. As “ipcs” only displays locks the calling user has read access to, you may run it as user root.
- You may also check “/proc/locks” if you suspect there is some locking problem. Unfortunately, Linux supports several ways of locking, and I don’t know a single place where all locks and lock contentions are displayed.
BSD Accounting

β Writes one accounting record per terminated process or thread (as threads are something like processes in Linux...)
β Information provided:
  – user ID, group ID, process name
  – CPU resource consumption
  – average memory usage, page faults, swap activity
β An alternative to accounting Linux "from the inside" is accounting it "from the outside", with the aid of z/VM or z/OS performance tools
“sysstat” package

- Contains sar and sadc, long term data collector
- Normally, it collects data about overall system activity like CPU usage, swapping; no data about processes
- start with
  - $ > sadc 60 /var/log/sa/sa25 &
- to let it generate one report every 60 seconds and write it in binary format to /var/log/sa/sa25
- http://freshmeat.net/projects/sysstat/
## sar: some options

<table>
<thead>
<tr>
<th>CPU</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sar -u</td>
<td>CPU Utilization Data: %user, %nice, %system, %idle</td>
</tr>
<tr>
<td></td>
<td>sar –U &lt;n&gt;</td>
<td>Like “sar –u”, but only for CPU number &lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td>sar –c</td>
<td>Process creation rate</td>
</tr>
<tr>
<td></td>
<td>sar –w</td>
<td>Context switch rate</td>
</tr>
<tr>
<td>Mem</td>
<td>sar –r</td>
<td>Memory and swap space utilization</td>
</tr>
<tr>
<td></td>
<td>sar –R</td>
<td>Memory usage statistics (buffer growth, ...)</td>
</tr>
<tr>
<td></td>
<td>sar -B</td>
<td>Paging statistics</td>
</tr>
<tr>
<td></td>
<td>sar –w</td>
<td>Swapping activity</td>
</tr>
<tr>
<td>I/O</td>
<td>sar –b</td>
<td>I/O and transfer rate statistics</td>
</tr>
<tr>
<td></td>
<td>sar –d</td>
<td>Block device statistics</td>
</tr>
<tr>
<td></td>
<td>sar –n DEV</td>
<td>Network device statistics</td>
</tr>
<tr>
<td></td>
<td>sar –n EDEV</td>
<td>Network device error rates</td>
</tr>
<tr>
<td></td>
<td>sar –n SOCK</td>
<td>Socket statistics</td>
</tr>
</tbody>
</table>
### sar: some examples

<table>
<thead>
<tr>
<th>Interface</th>
<th>rxpk/s</th>
<th>txpk/s</th>
<th>rxbyt/s</th>
<th>txbyt/s</th>
<th>rxcmp/s</th>
<th>txcmp/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>0.04</td>
<td>0.04</td>
<td>2.00</td>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.66</td>
<td>0.13</td>
<td>213.35</td>
<td>22.63</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>lo</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.49</td>
<td>0.01</td>
<td>168.34</td>
<td>1.18</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>lo</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.54</td>
<td>0.01</td>
<td>171.53</td>
<td>1.08</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>lo</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.51</td>
<td>0.01</td>
<td>171.73</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>lo</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.50</td>
<td>0.01</td>
<td>170.58</td>
<td>1.08</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>lo</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>eth0</td>
<td>0.55</td>
<td>0.01</td>
<td>174.42</td>
<td>0.98</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Average</td>
<td>lo</td>
<td>0.01</td>
<td>0.01</td>
<td>0.56</td>
<td>0.56</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>sit0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>eth0</td>
<td>0.54</td>
<td>0.03</td>
<td>100.50</td>
<td>5.19</td>
<td>0.00</td>
</tr>
</tbody>
</table>

```bash
benke@lnormf:/var/lock> sar -n -R 10:00:00 -e 11:00:00
Linux 2.4.19-3suse-SP4 (Innormf)    07/28/2003

10:00:01 AM  IFACE  rxpk/s  txpk/s  rxbyt/s  txbyt/s  rxcmp/s
10:10:00 AM  lo    0.04    0.04    2.00    2.00    0.00
10:10:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
10:10:00 AM  eth0  0.66    0.13    213.35  22.63   0.00
10:20:00 AM  lo    0.00    0.00    0.00    0.00    0.00
10:20:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
10:20:00 AM  eth0  0.49    0.01    168.34  1.18    0.00
10:30:00 AM  lo    0.00    0.00    0.00    0.00    0.00
10:30:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
10:30:00 AM  eth0  0.54    0.01    171.53  1.08    0.00
10:40:00 AM  lo    0.00    0.00    0.00    0.00    0.00
10:40:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
10:40:00 AM  eth0  0.51    0.01    171.73  0.00    0.00
10:50:00 AM  lo    0.00    0.00    0.00    0.00    0.00
10:50:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
10:50:00 AM  eth0  0.50    0.01    170.58  1.08    0.00
11:00:00 AM  lo    0.00    0.00    0.00    0.00    0.00
11:00:00 AM  sit0  0.00    0.00    0.00    0.00    0.00
11:00:00 AM  eth0  0.55    0.01    174.42  0.98    0.00
Average:    lo    0.01    0.01    0.56    0.56    0.00
Average:    sit0  0.00    0.00    0.00    0.00    0.00
Average:    eth0  0.54    0.03    100.50  5.19    0.00
```
RMFPMS

- Long term data gathering
- XML over HTTP interface
- Independent from z/OS; with z/OS, you can also have an LDAP interface to Linux performance data
- Modular architecture
- zSeries specific information (like LPAR data) can be obtained using existing z/VM or z/OS code
- Integrated with z/OS RMF PM and z/VM FCON
  - If you have a mixed environment with z/OS and Linux or z/VM and FCON, you can have all relevant performance metrics in one application
  - Data reported by host tools like RMF (LPAR CPU performance data, iQDIO channel utilization, etc.) is very relevant for Linux; unfortunately, we cannot make all this data available for Linux currently

See

RMF PM Java Client
RMF PM Java Client: Features

- Positioned for online performance analysis and problem drill-down
- Can monitor multiple Linux server and multiple z/OS or OS/390 Sysplexes at the same time, in one application
- The performance analysis scenario can be saved
- Alternatively, you may use the web browser interface of the Distributed Data Server (DDS)
RMF PM: Spreadsheet Data
Enhanced RMFPMS Web Browser Interface

This metric measures the number of processes created per second. If this number is high, then a large number of processes are being started. Each time a process is created, there is some amount of overhead associated with this creation; this overhead can become a performance problem if the rate of process creation becomes large.
... you can now create your own customizable view even in a Web browser like Mozilla, Explorer, Netscape
Accessing Linux Performance Data: Concept
FCON configuration

File FCONX LINUXUSR

*******************************************************************************
** Initialization file with IP address definitions **
** for Linux systems that may have to be monitored. **
*******************************************************************************

* 
LINUX1  1.111.111.111:8803
LINUX2  2.222.222.222:8803
LINUX3  3.333.333.333:8803
...
...

Defines IP addresses of Linux systems from which performance data
may have to be retrieved.
You can only monitor systems defined in this file!
LPAR partition data from z/OS RMF
HiperSockets display in z/VM FCON

<table>
<thead>
<tr>
<th>Channel Path</th>
<th>CPU 2064</th>
<th>SER 51524</th>
<th>Interval 06:55:22 - 06:56:22</th>
<th>Perf. Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shrd</td>
<td>T_Msgs</td>
<td>T_DUnits</td>
<td>T_NoBuff</td>
<td>L_Msgs</td>
</tr>
<tr>
<td>FB</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FC</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FD</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FE</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
… and in z/OS RMF

<table>
<thead>
<tr>
<th>CHANNEL PATH ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Utilization (%)</td>
</tr>
<tr>
<td>Group G No</td>
</tr>
<tr>
<td>FC_SM 1</td>
</tr>
<tr>
<td>FCV_M</td>
</tr>
<tr>
<td>CNC_M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DETAILS FOR ALL CHANNELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Path</td>
</tr>
<tr>
<td>ID Type</td>
</tr>
<tr>
<td>CVC_P</td>
</tr>
<tr>
<td>FC</td>
</tr>
<tr>
<td>FC_SM</td>
</tr>
<tr>
<td>FCV_M</td>
</tr>
<tr>
<td>CNC_M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESSAGE RATE</th>
<th>MESSAGE SIZE</th>
<th>SEND FAIL</th>
<th>RECEIVE FAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB IQD</td>
<td>645.12M</td>
<td>2500.2G</td>
<td>85.023K</td>
</tr>
</tbody>
</table>
CP IND interface in Linux

- Interface between Linux kernel and z/VM CP
- CP device driver, developed by Neale Ferguson; interface between Linux and z/VM
- http://penguinvm.princeton.edu/programs (cpint.tar.gz)
- "$\texttt{#cp ind user}$" in Linux console:
  
  CP IND
  AVGPROC-069% 07
  XSTORE-000037/SEC MIGRATE-0000/SEC
  MDC READS-000001/SEC WRITES-000000/SEC HIT RATIO-094%
  STORAGE-024% PAGING-0000/SEC STEAL-000%
  Q0-00071 Q1-00000 Q2-00000 EXPAN-001 Q3-00000 EXPAN-001
Example scenario

β The following Linux image may be completely idle:

$ > top 12:30pm
up 4 min, 2 users, load average: 0.02, 0.07, 0.03
24 processes: 23 sleeping, 1 running, 0 zombie, 0 stopped
CPU0 states: 0.1% user, 19.1% system, 0.0% nice, 80.8% idle
CPU1 states: 0.0% user, 23.2% system, 0.0% nice, 76.8% idle
...

β ... as z/VM is heavily loaded and does not give Linux many resources, so even for simple tasks, Linux needs about 20% of its CPU resources just to do almost nothing:

$ > #CP IND
AVGPROC-099% 07
...
The NET-SNMP Project

SNMP (Simple Network Management Protocol) is a standard for performance data interchange. It is especially strong in TCP/IP network management. It is standardized by the IETF (Internet Engineering Task Force).

- SNMP has a simple Manager-Agent architecture. Standard protocol used is UDP (connectionless, delivery not guaranteed).
- NET-SNMP provides a free SNMP implementation, also usable for Linux for zSeries. The OSA adapter provides some performance information using SNMP.
- See http://net-snmp.sourceforge.net/
What is CIM?

- CIM is a systems management standard provided by the DMTF (Distributed Management Task Force), a sub group of The Open Group. It is the dominant standard in SAN management, but also applicable to all other areas of systems management. It provides bridges to SNMP, e.g. for TCP/IP network management.

- One of the strength of CIM is the rich conceptual data model with about 1000 classes for major resources needed in the management of heterogeneous, distributed servers.

- OpenPegasus, “C++ CIM/WBEM Manageability Services Broker”, is the DMTF reference implementation of a CIMOM. It is published under the liberal MIT license in open source. See http://www.openpegasus.org/
CIM Provider, Object Manager, and Client
The goal of **WBEM (Web-based Enterprise Management)** is to provide interoperable technology based on the CIM standard. This standard is also driven by the DMTF.


It currently uses **XML over HTTP** protocol, but this may change into **ASN.1 over HTTP** for performance reasons.

**CMPI (Common Manageability Programming Interface)** instrumentation interface (standardized API with CIM compliant semantics and operations) to make provider independent from CIMOM technology.
SBLIM Reference Implementation
References

- “Linux on IBM eServer zSeries and S/390: Performance Measurement and Tuning” Redbook, SG24-6926
- “Linux on zSeries and S/390: Systems Management Redbook, SG24-6820
- “Linux for IBM eServer zSeries and S/390: ISP/ASP Solutions” Redbook, SG24-6299
Questions?

Email: benke@de.ibm.com