IBM Systems Group

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*
the IBM logo*
OS/390*
Parallel Sysplex*
MVS
z/OS*

RACF*
RMF
zSeries
Tivoli*
CICS
IMS

DB2*
WebSphere*
Domino
e business(logo)*
e(logo)server
e(logo)businss

Lotus*
Tivoli(logo)*
z/VM*
z/Architecture
zSeries*

* 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

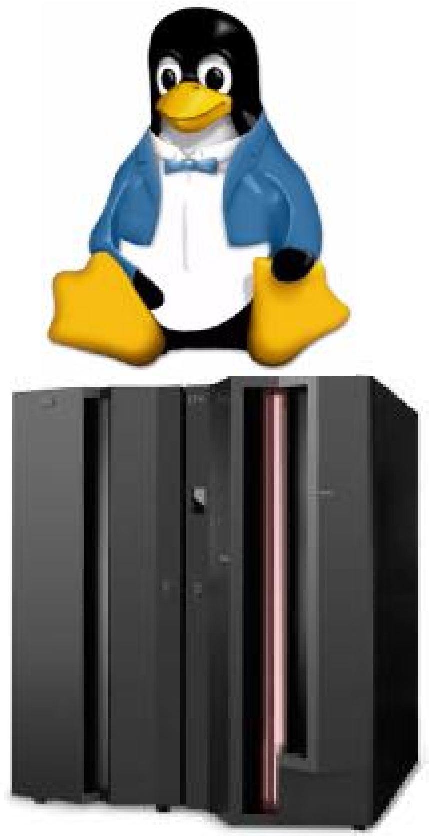
- Performance Management
- Resource Sharing, Overcommitted Resources, Virtualization
  - CPU Resources in a virtualized environment
- zSeries Mainframes: what’s different?
- Performance base concepts
  - Load Average
  - System/User CPU Consumption
- The /proc filesystem

Performance Management

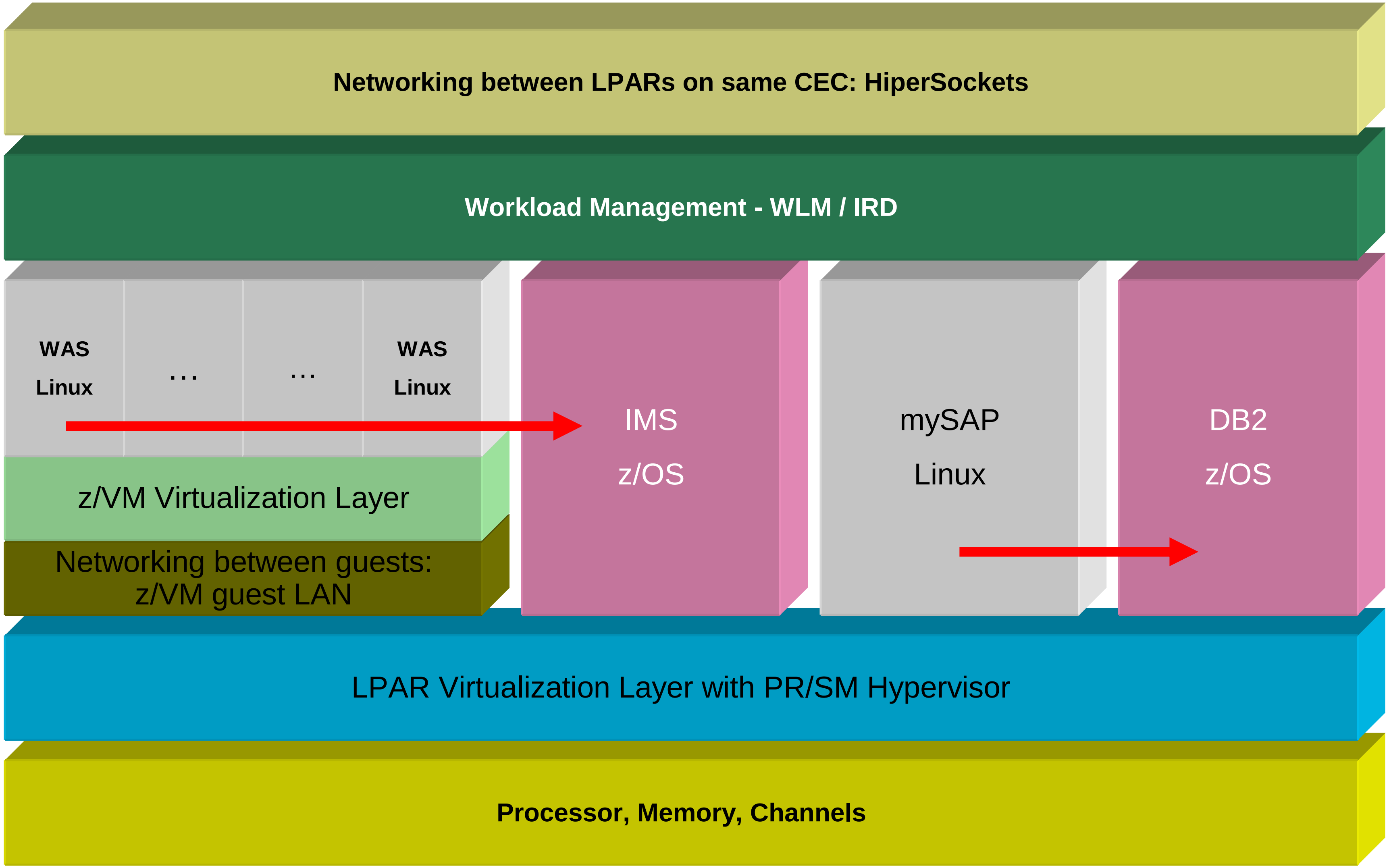
- Online Monitoring, Problem drill-down; 1 day history (or 3 days for the weekend) needed
  - May be automated, using asynchronous events
  - Online performance data may be used by autonomic software components, like VMRM and IRD on zSeries
- Long-term monitoring and capacity planning
  - Understand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



Virtual Resources

- … can be shared between several instances which do not even know about each other, like several companies hosted by the same data center
- … 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
- … 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.

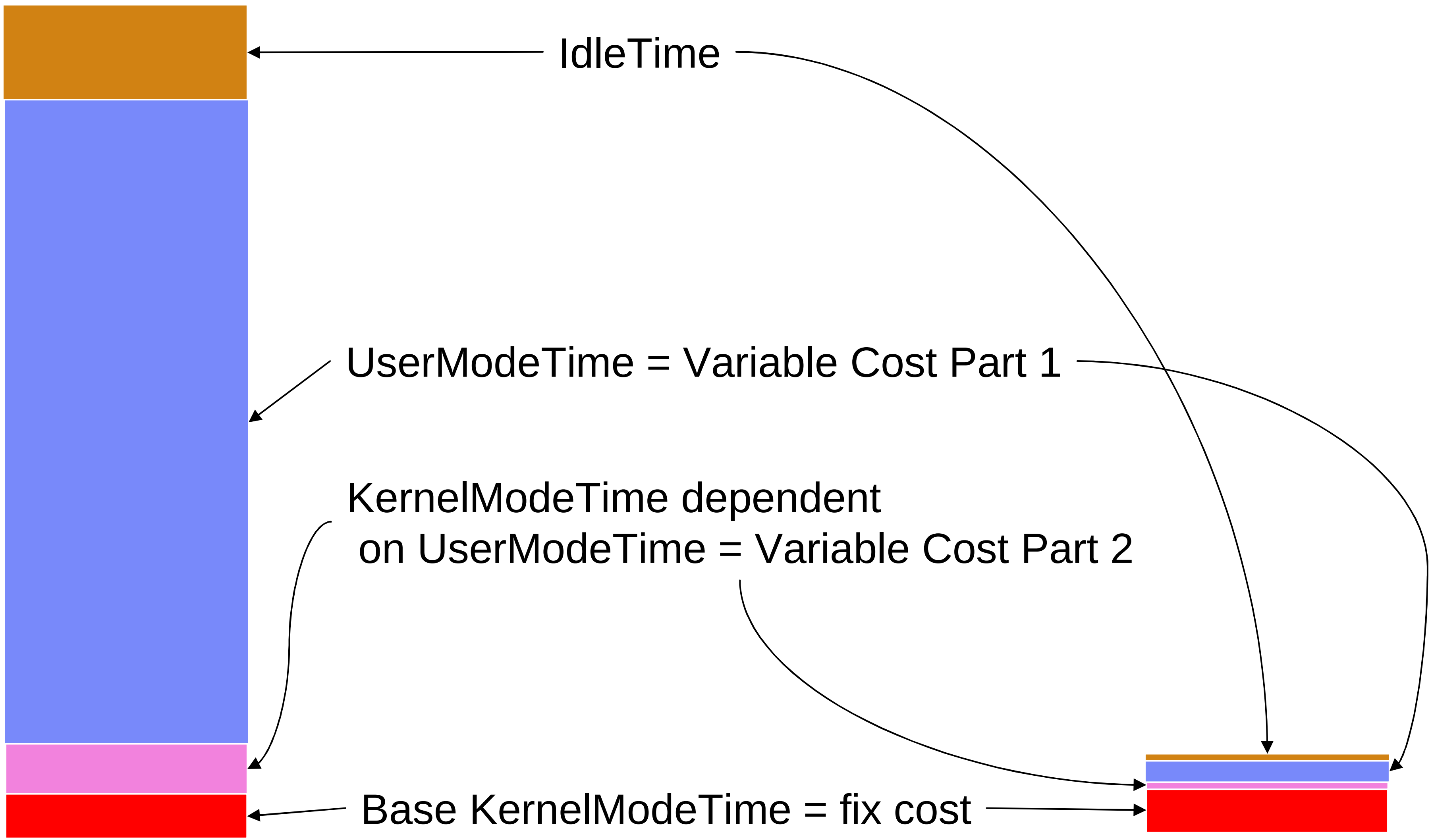
Internal Virtual Networks

- 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

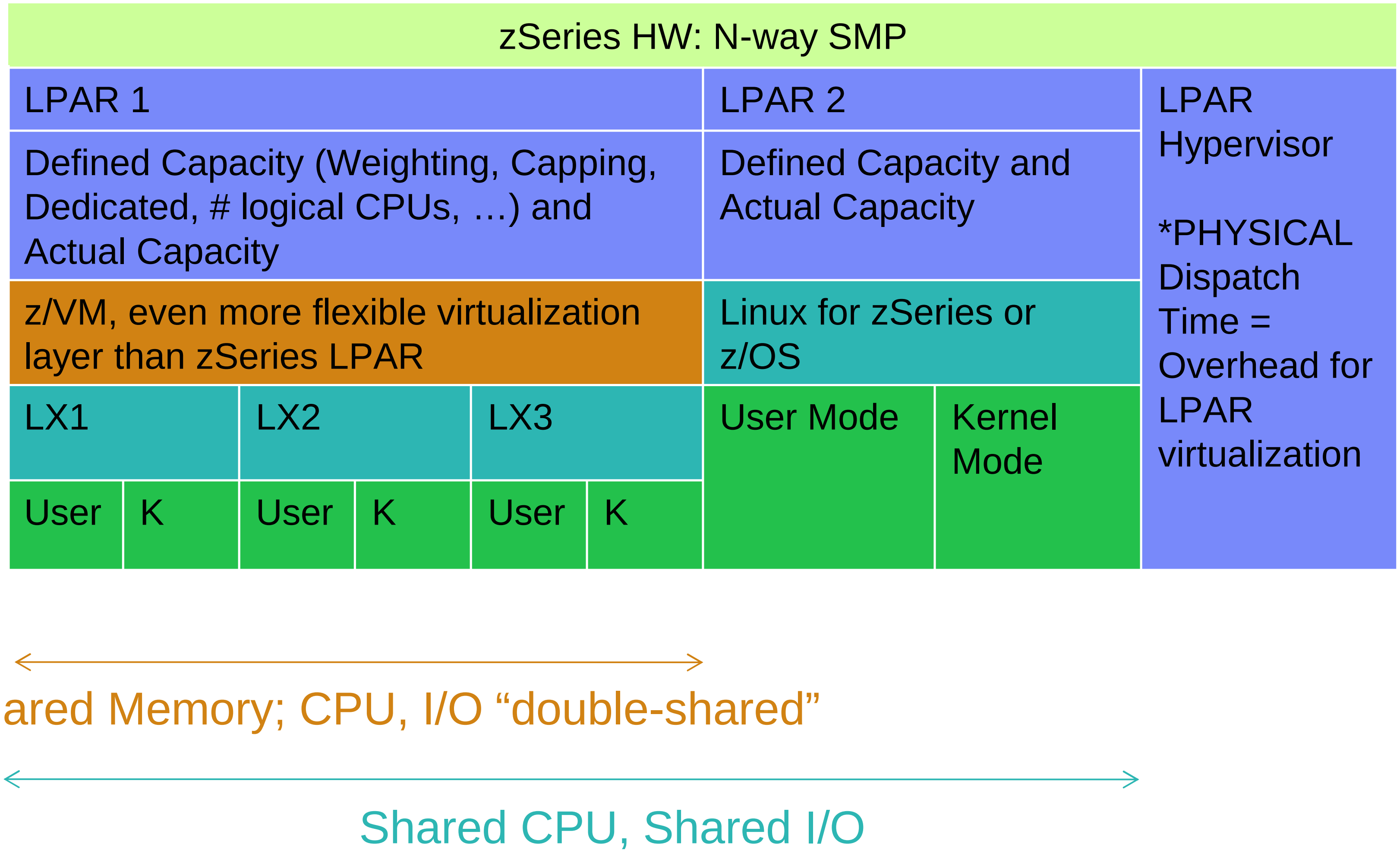
User-mode and kernel-mode CPU time consumption

- 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



Resource Sharing of CPU resources: the zSeries way



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

  cat /proc/loadavg

  or using program like *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

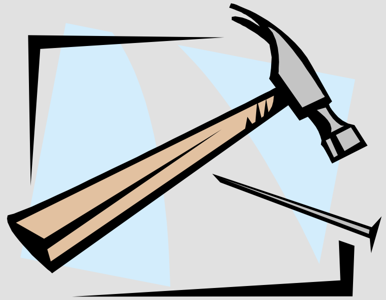
  cat /proc/stat

  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

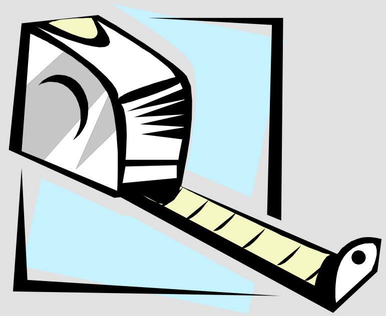
/proc/stat Example

```
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--(0%)
```

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**

```
xterm
 12:03pm  up 26 days, 19:06,  4 users,  load average: 0.59, 0.22, 0.13
61 processes: 59 sleeping, 2 running, 0 zombie, 0 stopped
CPU0 states:  0.0% user,  0.2% system,  0.0% nice, 99.3% idle
CPU1 states:  0.0% user,  0.0% system,  0.0% nice, 100.0% idle
CPU2 states: 98.3% user,  1.1% system,  0.0% nice,  0.0% idle
Mem:   123168K av,  117540K used,    5628K free,       0K shrd,     191
Swap:  503980K av,    7416K used,  496564K free                    1558

  PID USER     PRI  NI  SIZE  RSS SHARE WCHAN     STAT %CPU %MEM COMMA
27586 benke     25   0 29576  28M  3516           R    99.8 24.0 cc1pl
27546 benke     15   0  1644 1640  1368           R     0.3  1.3 top
    1 root      15   0    92   72    52 schedule_ S     0.0  0.0 init
    2 root      0K   0     0    0     0 migration SW    0.0  0.0 migra
    3 root      0K   0     0    0     0 migration SW    0.0  0.0 migra
    4 root      0K   0     0    0     0 migration SW    0.0  0.0 migra
    5 root      25   0     0    0     0 down_inte SW    0.0  0.0 kmche
    6 root      15   0     0    0     0 context_t SW    0.0  0.0 keven
    7 root      34  19     0    0     0 ksoftirqd SWN   0.0  0.0 ksoft
    8 root      34  19     0    0     0 ksoftirqd SWN   0.0  0.0 ksoft
    9 root      34  19     0    0     0 ksoftirqd SWN   0.0  0.0 ksoft
   10 root      15   0     0    0     0 kswapd    SW    0.0  0.0 kswap
   11 root      25   0     0    0     0 bdflush   SW    0.0  0.0 bdflu
   12 root      15   0     0    0     0 schedule_ SW    0.0  0.0 kupda
   13 root      16   0     0    0     0 kinoded   SW    0.0  0.0 kinod
```

```
xterm
   Current Field Order: AbcdgHIjklMnoTPlQRSUzYV{EFWX
Toggle fields with a-x, any other key to return:

* A: PID        = Process Id
  B: PPID       = Parent Process Id
  C: UID        = User Id
  D: USER       = User Name
* E: %CPU       = CPU Usage
* F: %MEM       = Memory Usage
  G: TTY        = Controlling tty
* H: PRI        = Priority
* I: NI         = Nice Value
  J: PAGEIN     = Page Fault Count
  K: TSIZE      = Code Size (kb)
  L: DSIZE      = Data+Stack Size (kb)
* M: SIZE       = Virtual Image Size (kb)
  N: TRS        = Resident Text Size (kb)
  O: SWAP       = Swapped kb
* P: SHARE      = Shared Pages (kb)
* Q: A          = Accessed Page count
* R: WP         = Write Protected Pages
* S: D          = Dirty Pages
* T: RSS        = Resident Set Size (kb)
* U: WCHAN      = Sleeping in Function
* V: STAT       = Process Status
* W: TIME       = CPU Time
* X: COMMAND    = Command
* Y: LC         = Last used CPU (expect this to change regularly)
  Z: FLAGS      = Task Flags (see linux/sched.h)
```

ps - report process status

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

```
bash-2.05# ps aux|more
USER       PID %CPU %MEM   VSZ  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 ?        SW   Jan22   0:00 [kmcheck]
root         3  0.0  0.0     0    0 ?        SW   Jan22   0:00 [keventd]
root         4  0.0  0.0     0    0 ?        SW   Jan22   0:22 [kswapd]
root         5  0.0  0.0     0    0 ?        SW   Jan22   0:00 [kreclaimd]
root         6  0.0  0.0     0    0 ?        SW   Jan22   0:00 [bdflush]
root         7  0.0  0.0     0    0 ?        SW   Jan22   1:05 [kupdated]
root        63  0.0  0.0     0    0 ?        SW<  Jan22   0:00 [mdrecoveryd]
root       248  0.0  0.0     0    0 ?        SW   Jan22   0:00 [keventd]
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

```
benke@lnxrmf:~/rmfpms/src> pstree
init-+-atd
     |-automount
     |-bdflush
     |-clustergat
     |-cron
     |-filegat
     |-gengat
     |-gpmddsrv---gpmddsrv---5*[gpmddsrv]
     |-keventd---qethsoftd0001
     |-kinoded
     |-kjournald
     |-klogd
     |-kmcheck
     |-ksoftirqd_CPU0
     |-ksoftirqd_CPU1
     |-ksoftirqd_CPU2
     |-kswapd
     |-kupdated
     |-lvm-mpd
     |-master-+-pickup
     |        `-qmgr
     |-mdrecoveryd
     |-migration_CPU0
     |-migration_CPU1
     |-migration_CPU2
     |-mingetty
     |-netgat
     |-nscd---nscd---5*[nscd]
     |-portmap
     |-procgat
     |-sshd---sshd---sshd---bash-+-3*[xterm---bash]
     |                           `-xterm---bash---pstree
     |-syslogd
     `-xdm
benke@lnxrmf:~/rmfpms/src>
```

```
benke@lnxrmf:~/rmfpms/src> pstree -almore
init)
  |-atd)
  |-automount) /netx file /etc/mount.xteam
  |-(bdflush)
  |-clustergat) 60
  |-cron)
  |-filegat) 60
  |-gengat) 60
  |-gpmddsrv)
  |   `-gpmddsrv)
  |       |-gpmddsrv)
  |       |-gpmddsrv)
  |       |-gpmddsrv)
  |       |-gpmddsrv)
  |       `-gpmddsrv)
  |-(keventd)
  |   `-(qethsoftd0001)
  |-(kinoded)
  |-(kjournald)
  |-klogd) -c 7 -2
  |-(kmcheck)
  |-(ksoftirqd_CPU0)
  |-(ksoftirqd_CPU1)
  |-(ksoftirqd_CPU2)
  |-(kswapd)
  |-(kupdated)
  |-(lvm-mpd)
  |-master)
  |   |-pickup) -l -t fifo -u
  |   `-qmgr) -l -t fifo -u
  |-(mdrecoveryd)
  |-(migration_CPU0)
  |-(migration_CPU1)
  |-(migration_CPU2)
  |-mingetty) /dev/ttyS0
  |-netgat) 60
  |-nscd)
  |   `-nscd)
  |       |-nscd)
  |       |-nscd)
  |       |-nscd)
  |       |-nscd)
  |       `-nscd)
  |-portmap)
  |-procgat) 60
--More--
```

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
```

/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

```
xterm
benke@lnxrmf:~> cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  126124032 119640064  6483968        0 10465280 57475072
Swap: 516075520 12390400 503685120
MemTotal:       123168 kB
MemFree:          6332 kB
MemShared:           0 kB
Buffers:         10220 kB
Cached:          51448 kB
SwapCached:       4680 kB
Active:          18064 kB
Inactive:        54368 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:       123168 kB
LowFree:          6332 kB
SwapTotal:      503980 kB
SwapFree:       491880 kB
benke@lnxrmf:~>
```

mpstat

- **mpstat** is used to display CPU relatded 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…

```
xterm
01:16:35 PM  CPU   %user   %nice %system   %idle    intr/s
01:16:35 PM  all    0.02    0.04    0.16   99.78      0.00
benke@lnxrmf:~> mpstat 10
Linux 2.4.19-3suse-SMP (lnxrmf)        07/28/2003

01:17:09 PM  CPU   %user   %nice %system   %idle    intr/s
01:17:19 PM  all   31.70    0.00    1.43   66.87      0.00
01:17:29 PM  all   32.40    0.00    0.97   66.63      0.00
01:17:39 PM  all   32.17    0.00    1.10   66.73      0.00
01:17:49 PM  all   23.57    0.00    0.87   75.57      0.00
01:17:59 PM  all    0.50    0.00    1.30   98.20      0.00
01:18:09 PM  all    0.37    0.00    4.10   95.53      0.00
01:18:19 PM  all    0.17    0.00    8.17   91.67      0.00
01:18:29 PM  all    0.70    0.00   12.27   87.03      0.00
01:18:39 PM  all    0.77    0.00   12.77   86.47      0.00
01:18:49 PM  all    0.53    0.00   13.50   85.97      0.00
01:18:59 PM  all    0.97    0.00   12.47   86.57      0.00
01:19:09 PM  all    0.90    0.00   13.20   85.90      0.00
01:19:19 PM  all    0.30    0.00    2.13   97.57      0.00
01:19:29 PM  all   19.33    0.00    2.73   77.93      0.00
01:19:39 PM  all   50.32    0.00    3.46   46.22      0.00
```

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.

xterm

```
benke@lnxrmf:~> vmstat 10 10
   procs                      memory    swap          io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
 0  0  0  14652  63732   2348  31064   0   0     2     2    0     2   0   0 100
 0  2  0  14392  44008   3196  24800 115   0  1264    20    0   236  11   2  87
 1  1  0  14232  24516   3204  61848  81   0  8684   141    0   589  32   5  63
 1  2  0  14192  26456   4040  54104  43   0  7371   186    0   859  32   4  63
 1  1  0  14192   2300   6112  53484  17   0  4731   286    0  1561  34   7  60
 1  2  1  14192   8496   8292  44140  14   0  4990   270    0  1394  31   7  62
 1  1  0  14192   2888   8796  30004  17   0  5047   294    0  1444  31   6  63
 1  1  0  14192   2352   6600  28744  17   0  4158   357    0  1393  32   6  62
 1  1  0  14264   2960   5708  29732  11  12  3554   345    0  1498  31   6  62
 2  1  0  14532   2364   4772  38244  14  20  4794   346    0  1195  30   6  64
benke@lnxrmf:~>
```

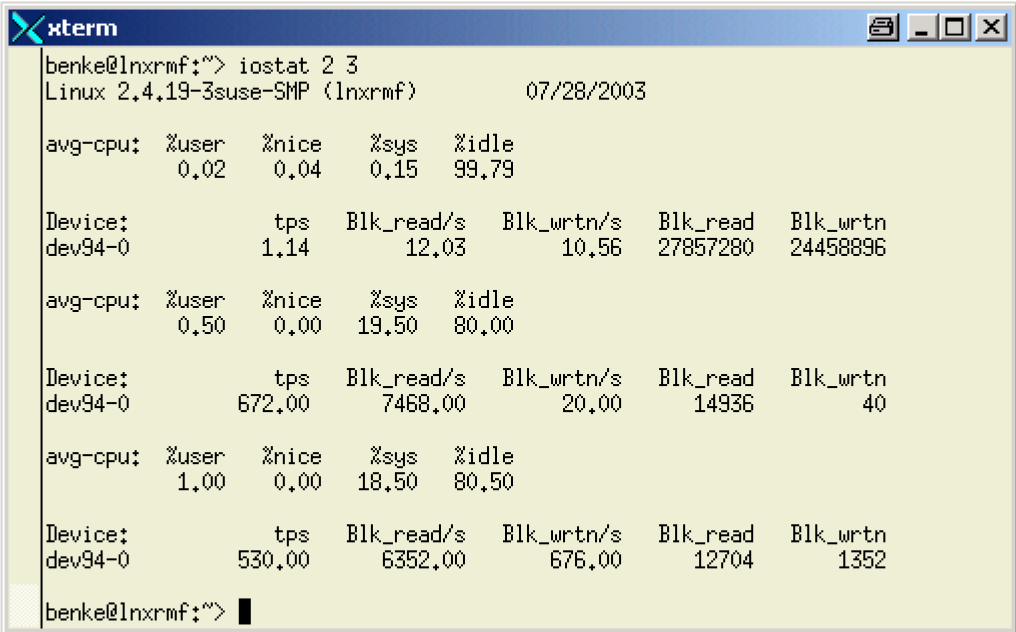
vmstat fields explained

procs	r	Number of Processes waiting for CPU, Ready to run
	b	Number of Processes blocked in uninterruptable wait (usually for I/O)
	w	Number of Processes swapped out but otherwise ready to run
memory	swpd	Memory used in swap space, in KB
	free	Real memory not used
	buff	Memory used for Buffers
	cache	Memory used for Cache
swap	si	Memory swapped in per second, in KB
	so	Memory swapped out per second, in KB
io	b	Blocks read from block devices per second
	bo	Blocks written to block device per second
system	in	Number of interrupts per second
	cs	Number of context switches per second
cpu	us	User time percentage of total CPU
	sy	System time percentage of total CPU
	id	Idle time percentage of total CPU

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_wrtn/s: number of blocks written per second



```
benke@lnxrmf:~> iostat 2 3
Linux 2.4.19-3suse-SMP (lnxrmf)        07/28/2003

avg-cpu:  %user   %nice    %sys   %idle
           0.02    0.04    0.15   99.79

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev94-0           1.14        12.03        10.56   27857280   24458896

avg-cpu:  %user   %nice    %sys   %idle
           0.50    0.00   19.50   80.00

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev94-0         672.00      7468.00        20.00      14936         40

avg-cpu:  %user   %nice    %sys   %idle
           1.00    0.00   18.50   80.50

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev94-0         530.00      6352.00       676.00      12704       1352

benke@lnxrmf:~>
```

/proc/dasd/statistics

- Only available in Linux for zSeries, kernel version 2.4
- Gathering of this information can be switched on and of, 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@lnxrmf:~> 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@lnxrmf:~>
```

- 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)

```
benke@lnxrmf:~> netstat -s|more
Ip:
    1314451 total packets received
    0 forwarded
    0 incoming packets discarded
    1205598 incoming packets delivered
    686873 requests sent out
    1867 reassemblies required
    805 packets reassembled ok
    108 fragments created
Icmp:
    3853 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 32
        echo requests: 3821
    3856 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 35
        echo replies: 3821
Tcp:
    52 active connections openings
    2404 passive connection openings
    0 failed connection attempts
    0 connection resets received
    3 connections established
    16493 segments received
    17316 segments send out
    4 segments retransmited
    0 bad segments received.
    229 resets sent
Udp:
    665606 packets received
    35 packets to unknown port received.
    0 packet receive errors
    665633 packets sent
```

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  bpl80002.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  c1-16.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  ehni1br2-2-0-1-1.eh.de.prserv.net (152.158.3.138)  2.006 ms   2.410 ms   2.384 ms
11  fran2br2.fr.de.prserv.net (152.158.92.2)  17.437 ms   17.940 ms   18.072 ms
12  dcix1nap-1-0-0.de.ip.att.net (152.158.93.237)  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
14  Stuttgart2.BelWue.DE (129.143.1.25)  14.016 ms   13.969 ms   13.910 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-GW.Uni-Karlsruhe.de (129.143.166.130)  14.446 ms   14.408 ms   14.910 ms
18  www-uka.rz.uni-karlsruhe.de (129.13.64.69)  14.114 ms   14.274 ms   14.234 ms
```

Filesystem Usage

```
benke@lnxrmf:/usr> df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasdb1           6.8G  4.2G  2.3G  65% /
shmfs                  61M     0   61M   0% /dev/shm
benke@lnxrmf:/usr> du -h
120M    ./bin
68K     ./share/doc/packages/aide
20K     ./share/doc/packages/words
24K     ./share/doc/packages/man-pages
4.0K    ./share/doc/packages/aaa_base
20K     ./share/doc/packages/intlfnt
64K     ./share/doc/packages/gnome-mime-data
36K     ./share/doc/packages/libaio
60K     ./share/doc/packages/perl-DateManip
16K     ./share/doc/packages/perl-HTML-Tagset
```

- 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 elapse

  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

  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

```
benke@lnxrmf:~> strace rmfpms/bin/rmfpms restart 2> straceoutput
Stopping performance gatherer backends ...
done!
Starting performance gatherer backends ...
 DDSRV: RMF-DDS-Server/Linux-Beta (Jul 28 2003) started.
 DDSRV: Functionality Level=1.950
 DDSRV: Reading exceptions from gpmexsys.ini and gpmexusr.ini.
DDSRV: Server will now run as a daemon process.
done!
benke@lnxrmf:~> more straceoutput
execve("rmfpms/bin/rmfpms", ["rmfpms/bin/rmfpms", "restart"], [/* 49 vars */]) = 0
uname({sys="Linux", node="lnxrmf", ...}) = 0
brk(0)                                  = 0x8009afc8
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x10000018000
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=86342, ...}) = 0
mmap(NULL, 86342, PROT_READ, MAP_PRIVATE, 3, 0) = 0x10000019000
close(3)                                = 0
open("/lib64/libreadline.so.4", O_RDONLY) = 3
read(3, "\177ELF\2\2\1\0\0\0\0\0\0\0\0\0\3\0\26\0\0\0\1\0\0\0"..., 1024) = 1024
fstat(3, {st_mode=S_IFREG|0755, st_size=860670, ...}) = 0
mmap(NULL, 267440, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x1000002f000
```


List open files (*lsof*)

```
benke@lnxrmf:~> lsof -c gpmddsrv | more
COMMAND     PID  USER   FD   TYPE             DEVICE    SIZE    NODE NAME
gpmddsrv 29791 benke  cwd    DIR               94,5    4096       2 /
gpmddsrv 29791 benke  rtd    DIR               94,5    4096       2 /
gpmddsrv 29791 benke  txt    REG               94,5 3901056  412063 /home/benke/rmfpms/bin/gpmddsrv
gpmddsrv 29791 benke  mem    REG               94,5  104611   16287 /lib64/ld-2.2.5.so
gpmddsrv 29791 benke  mem    REG               94,5   20425   16301 /lib64/libnss_dns.so.2
gpmddsrv 29791 benke  mem    REG               94,5  141963   16308 /lib64/libpthread.so.0
gpmddsrv 29791 benke  mem    REG               94,5   90264   16309 /lib64/libresolv.so.2
gpmddsrv 29791 benke  mem    REG               94,5 1201943  646126 /usr/lib64/libstdc++.so.5.0.0
gpmddsrv 29791 benke  mem    REG               94,5  512359   16297 /lib64/libm.so.6
gpmddsrv 29791 benke  mem    REG               94,5   53628   16351 /lib64/libgcc_s.so.1
gpmddsrv 29791 benke  mem    REG               94,5 1506104   16292 /lib64/libc.so.6
gpmddsrv 29791 benke  mem    REG               94,5   60576   16303 /lib64/libnss_files.so.2
gpmddsrv 29791 benke    0r   CHR                1,3           65089 /dev/null
gpmddsrv 29791 benke    1u   REG               94,5     958  406186 /home/benke/rmfpms/.rmfpms/logs/ddsrv_log.txt
gpmddsrv 29791 benke    2u   REG               94,5      55  406187 /home/benke/rmfpms/.rmfpms/logs/ddsrv_trc.txt
gpmddsrv 29791 benke    3r  FIFO                0,6         6061871 pipe
gpmddsrv 29791 benke    4w  FIFO                0,6         6061871 pipe
gpmddsrv 29791 benke    5u  IPv4            6061877             TCP *:8803 (LISTEN)
gpmddsrv 29791 benke    6u  unix 0x0000000000c4cd00         6061876 socket
gpmddsrv 29792 benke  cwd    DIR               94,5    4096       2 /
gpmddsrv 29792 benke  rtd    DIR               94,5    4096       2 /
gpmddsrv 29792 benke  txt    REG               94,5 3901056  412063 /home/benke/rmfpms/bin/gpmddsrv
gpmddsrv 29792 benke  mem    REG               94,5  104611   16287 /lib64/ld-2.2.5.so
gpmddsrv 29792 benke  mem    REG               94,5   20425   16301 /lib64/libnss_dns.so.2
--More--
```

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”:

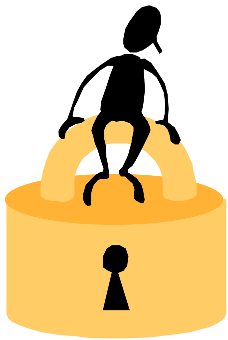
  lsof –i4tcp@tux390.boeblingen.de.ibm.com

  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

- Writes one accounting record per terminated process or thread (as threads are something like processes in Linux...)
- Information provided:
  - user ID, group ID, process name
  - CPU resource consumption
  - average memory usage, page faults, swap activity
- 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

CPU	sar -u	CPU Utilization Data: %user, %nice, %system, %idle
	sar –U <n>	Like “sar –u”, but only for CPU number <n>
	sar –c	Process creation rate
	sar –w	Context switch rate
Mem	sar –r	Memory and swap space utilization
	sar –R	Memory usage statistics (buffer growth, …)
	sar -B	Paging statistics
	sar –w	Swapping activity
I/O	sar –b	I/O and transfer rate statistics
	sar –d	Block device statistics
	sar –n DEV	Network device statistics
	sar –n EDEV	Network device error rates
	sar –n SOCK	Socket statistics

sar: some examples

```
xterm
benke@lnxrmf:/var/lock> sar -n DEV -s 10:00:00 -e 11:00:00
Linux 2.4.19-3suse-SMP (lnxrmf)          07/28/2003

10:00:01 AM     IFACE   rxpck/s   txpck/s   rxbyt/s   txbyt/s   rxcmp/s
10:10:00 AM        lo      0.04      0.04      2.80      2.80      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    219.95     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    168.84      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    171.63      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      0.00    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    170.38      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    180.50      5.19      0.00
benke@lnxrmf:/var/lock>
```

```
xterm
benke@lnxrmf:/var/lock> sar -b -s 10:00:00 -e 11:00:00
Linux 2.4.19-3suse-SMP (lnxrmf)          07/28/2003

10:00:01 AM       tps      rtps      wtps   bread/s   bw
10:10:00 AM      0.96      0.26      0.70      8.61
10:20:00 AM      0.66      0.00      0.66      0.04
10:30:00 AM      0.64      0.00      0.64      0.03
10:40:00 AM      0.66      0.00      0.66      0.03
10:50:00 AM      0.66      0.00      0.66      0.01
11:00:00 AM      0.66      0.00      0.65      0.01
Average:         0.72      0.05      0.66      1.74
benke@lnxrmf:/var/lock>
```

```
xterm
benke@lnxrmf:/var/lock> sar -u -s 10:00:00 -e 11:00:00
Linux 2.4.19-3suse-SMP (lnxrmf)          07/28/2003

10:00:01 AM       CPU     %user     %nice   %system     %idle
10:10:00 AM       all      0.02      0.00      0.14     99.84
10:20:00 AM       all      0.02      0.00      0.05     99.94
10:30:00 AM       all      0.01      0.00      0.05     99.94
10:40:00 AM       all      0.05      0.00      0.04     99.91
10:50:00 AM       all      0.02      0.00      0.05     99.94
11:00:00 AM       all      0.01      0.00      0.04     99.95
Average:          all      0.02      0.00      0.07     99.91
benke@lnxrmf:/var/lock>
```

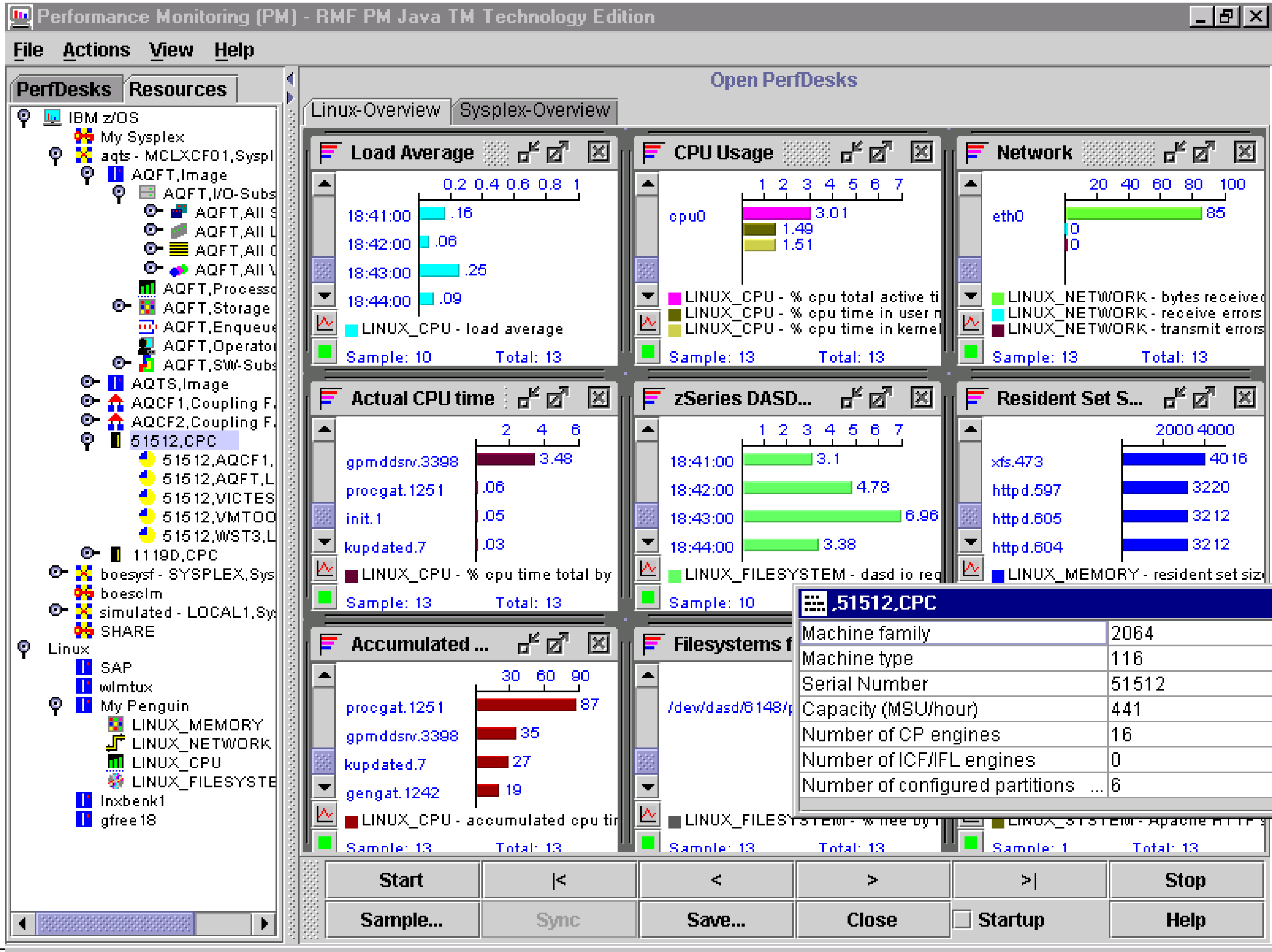
```
xterm
benke@lnxrmf:/var/lock> sar -W -s 10:00:00 -e 11:00:00
Linux 2.4.19-3suse-SMP (lnxrmf)          07/28/2003

10:00:01 AM  pswpin/s pswpout/s
10:10:00 AM      0.05      0.00
10:20:00 AM      0.00      0.00
10:30:00 AM      0.00      0.00
10:40:00 AM      0.00      0.00
10:50:00 AM      0.00      0.00
11:00:00 AM      0.00      0.00
Average:         0.01      0.00
benke@lnxrmf:/var/lock>
```

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

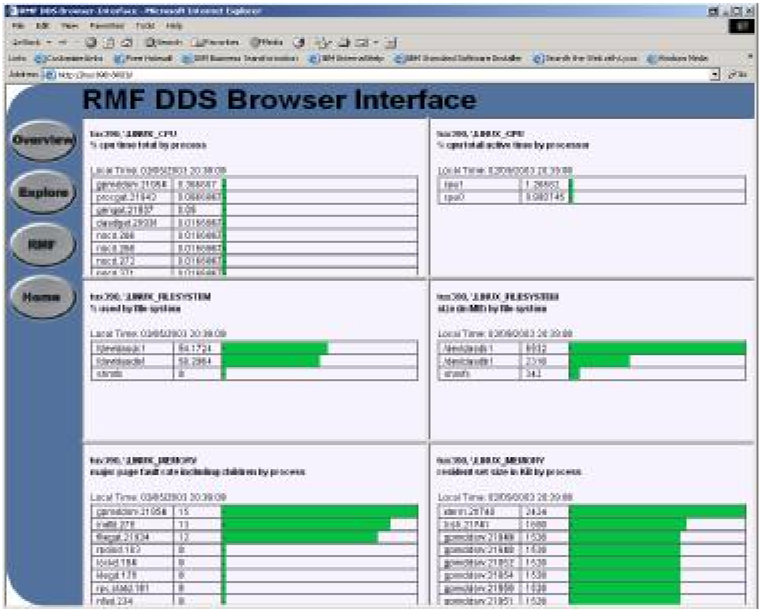
  http://www.ibm.com/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmlin.htm

RMF PM Java Client

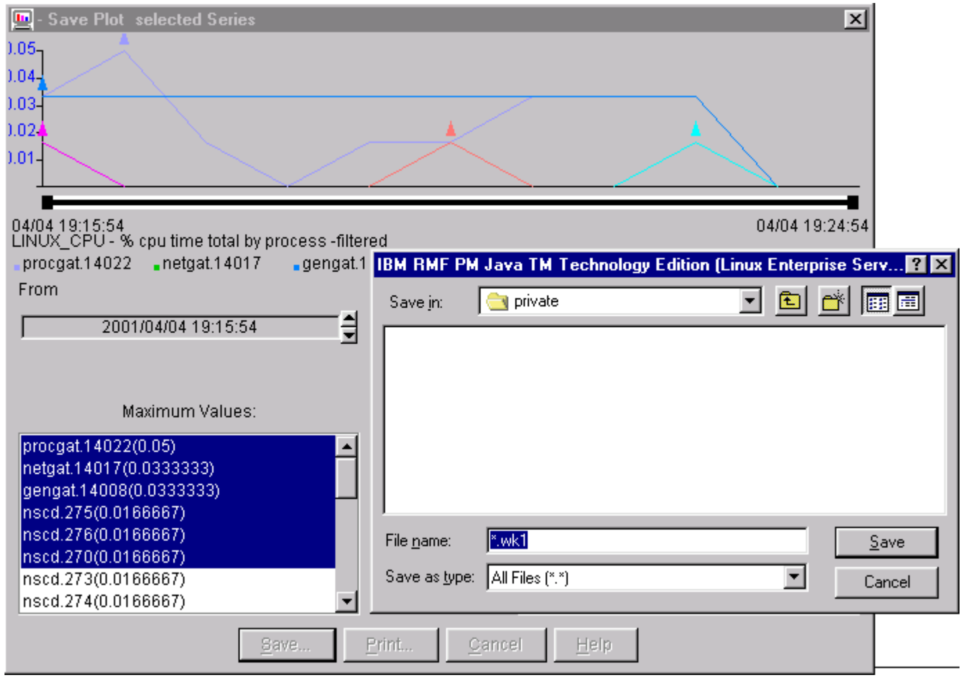


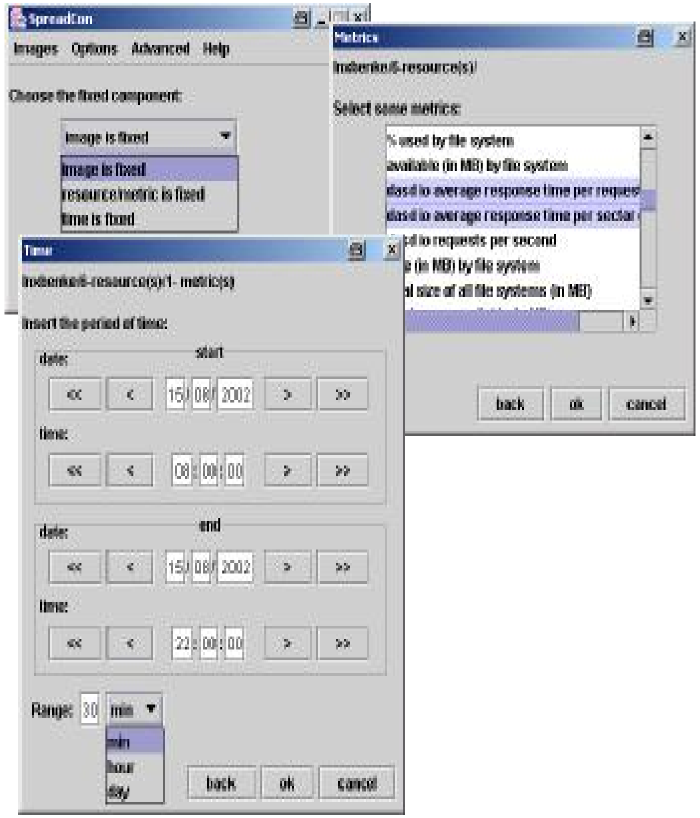
RMF PM Java Client: Features

- 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)

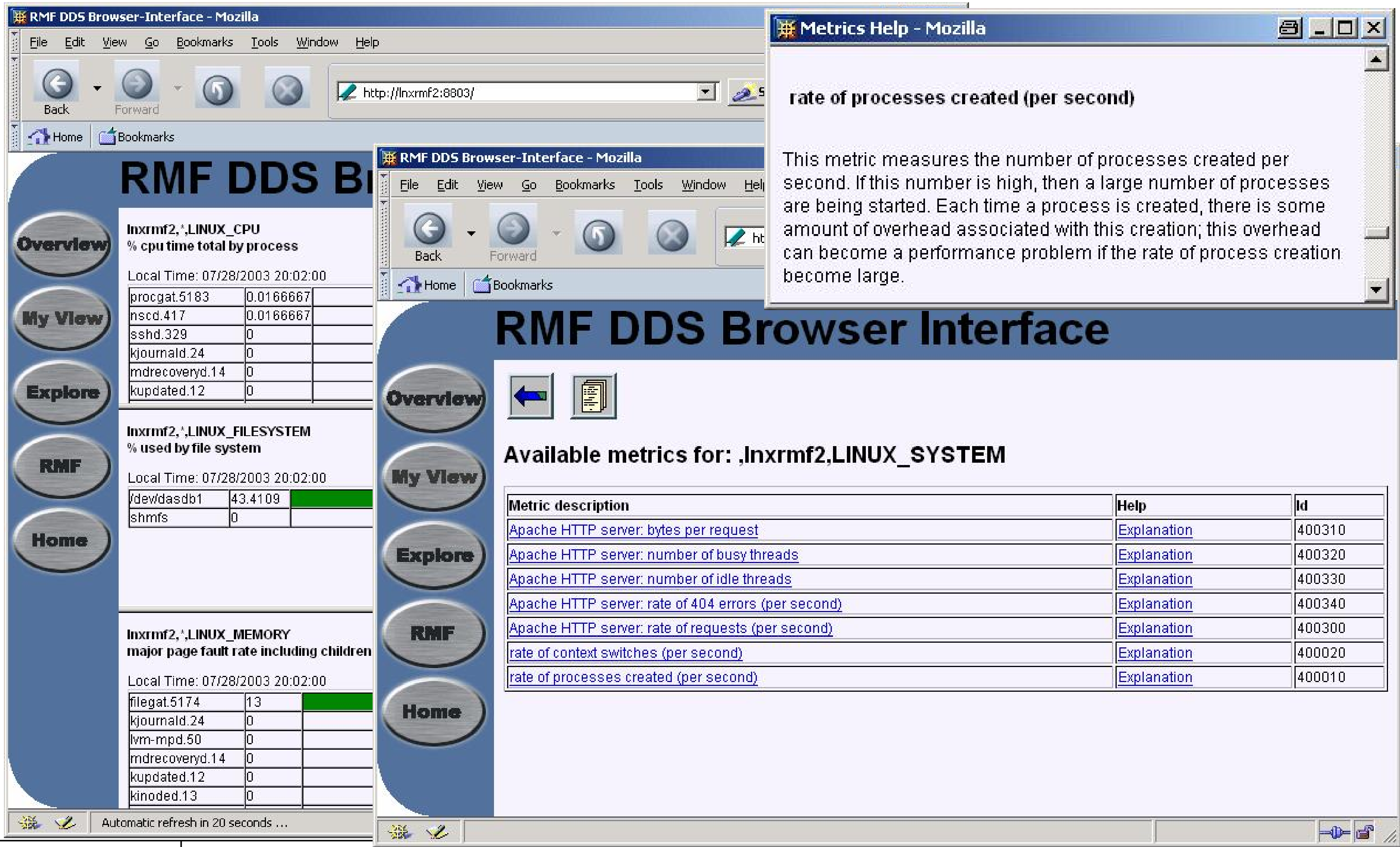


RMF PM: Spreadsheet Data

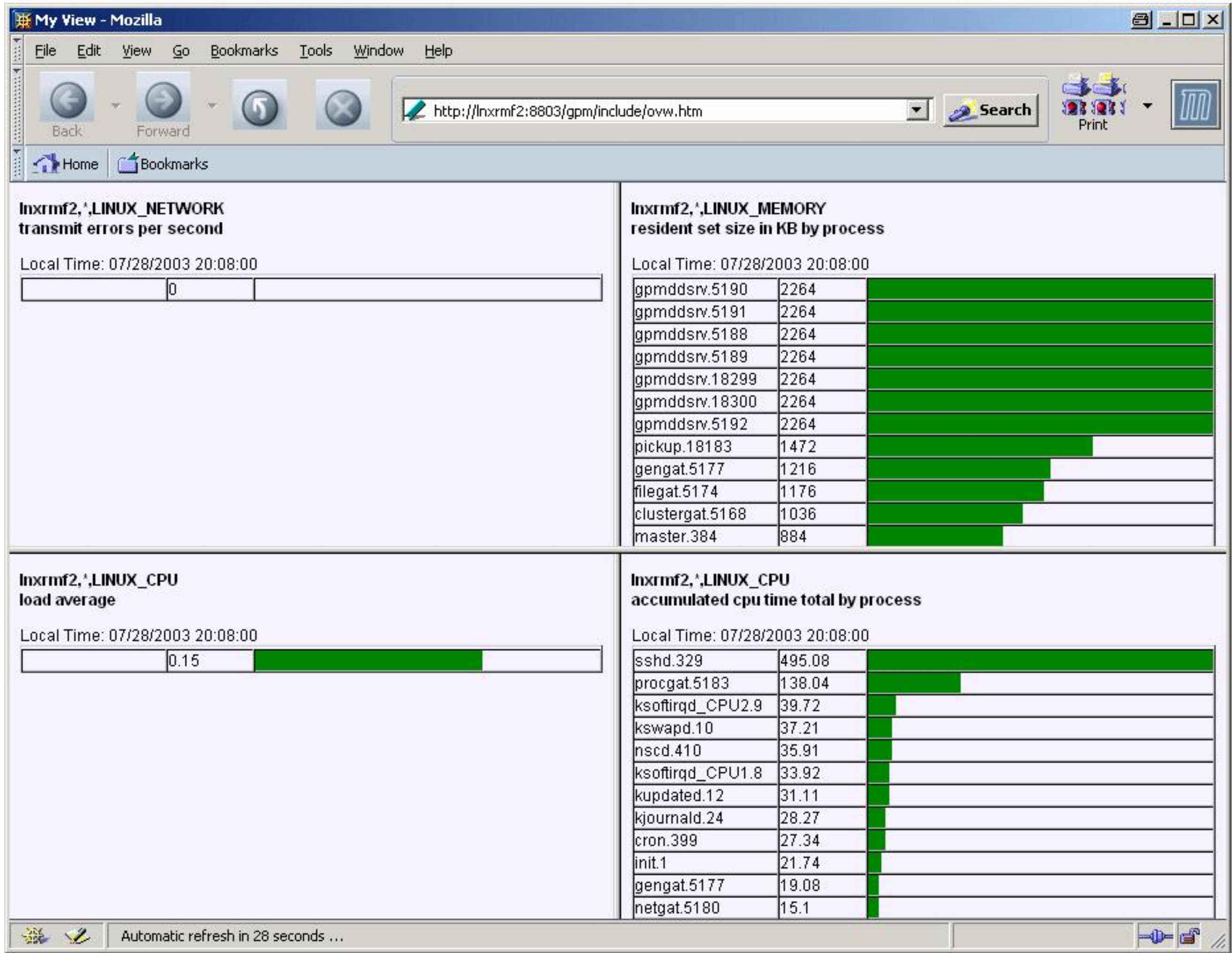




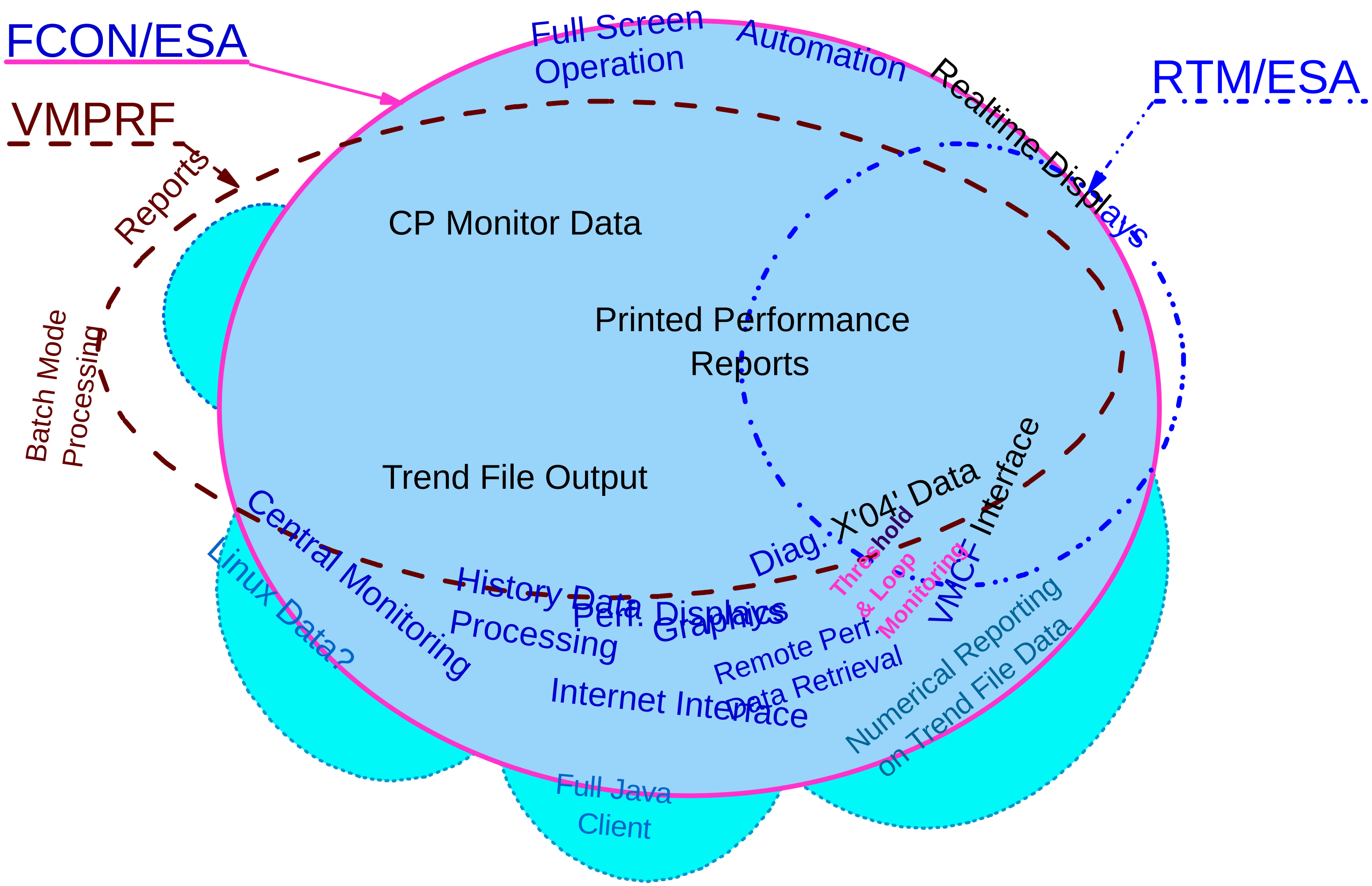
Enhanced RMFPMS Web Browser Interface



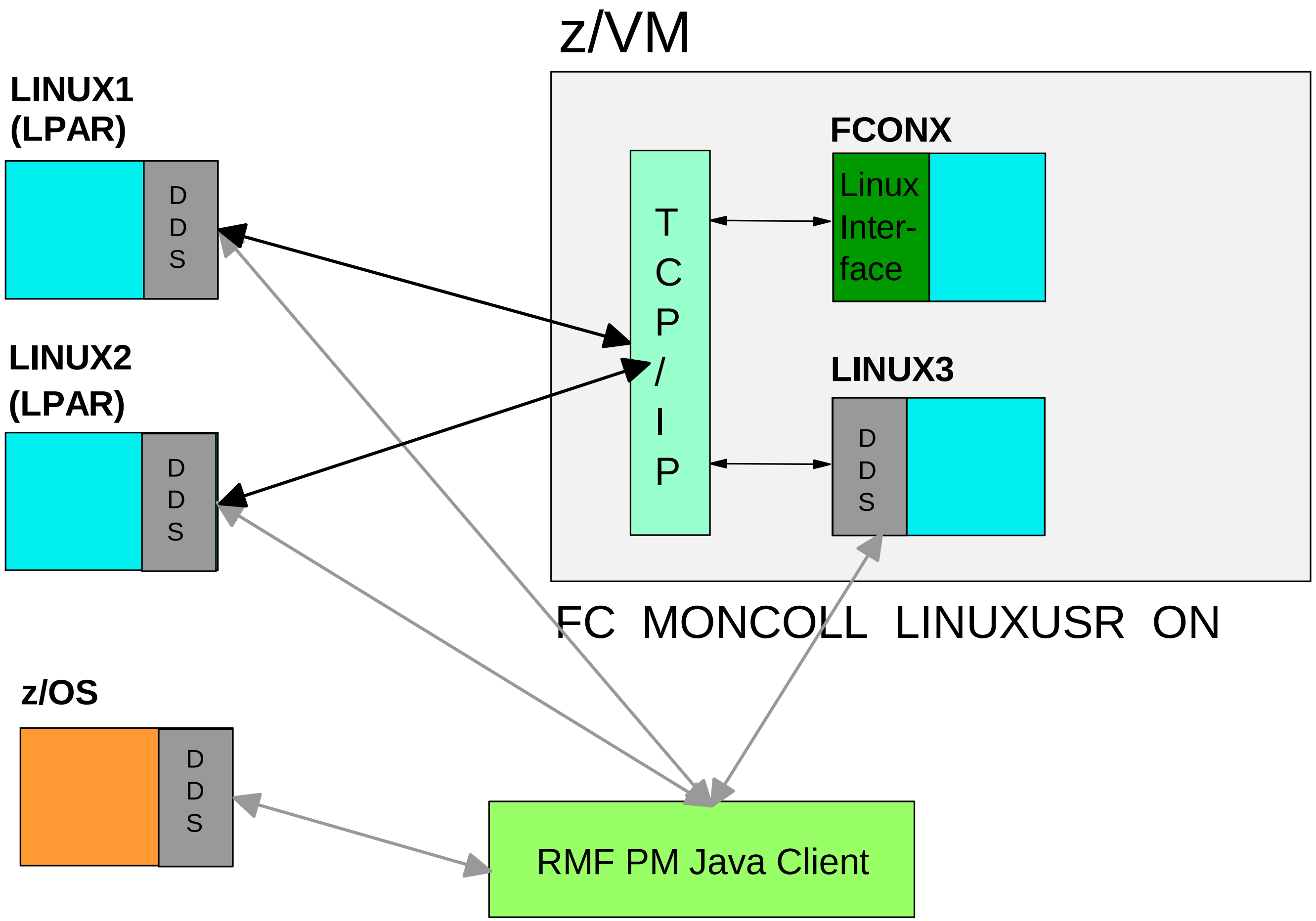
… you can now create your own customizable view even in a Web browser like Mozilla, Explorer, Netscape



z/VM FCON


FCON/ESA
VMPRF
RTM/ESA
Full Screen Operation
Automation
Realtime Displays
Reports
Batch Mode Processing
CP Monitor Data
Printed Performance Reports
Trend File Output
Central Monitoring
Linux Data?
History Data Processing
Perf. Displays
Graphics
Diag. X'04' Data
Threshold & Loop Monitoring
VMCF Interface
Remote Perf. Data Retrieval
Internet Interface
Numerical Reporting on Trend File Data
Full Java Client

Accessing Linux Performance Data: Concept



FCON configuration

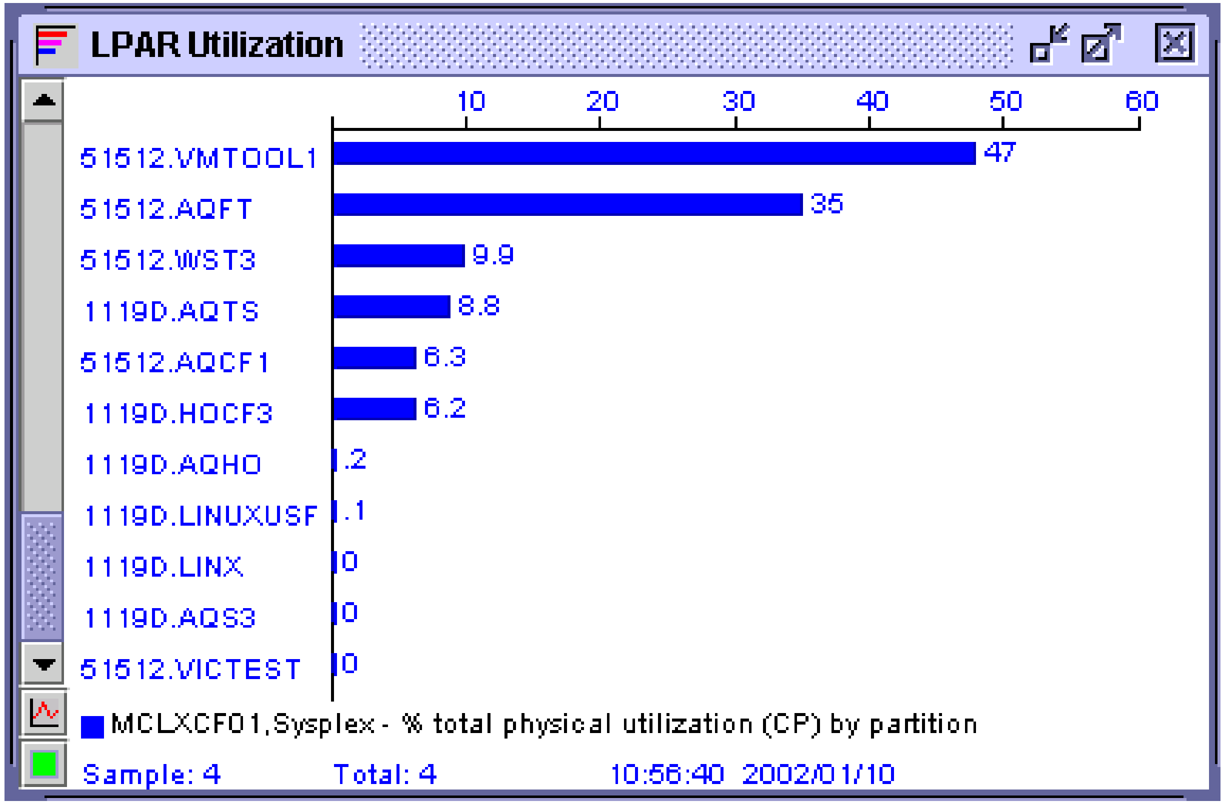
File **FCONX LINUXUSR**

```
*******************************************************
**  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

```
FCX231      CPU 2064   SER 51524  Interval 06:55:22 - 06:56:22    Perf. Monitor

_______            .         .         .       .         .         .        .
               <----------------- Hipersocket Activity/Sec. ----------------->
Channel        <--- Total for System ---> <--------- Own Partition --------->
Path           <-Transferred-->    Failed <-Transferred--> <--- Failed ---->
ID      Shrd   T_Msgs  T_DUnits  T_NoBuff L_Msgs  L_DUnits  L_NoBuff  L_Other
FB      No          0         0         0       0         0         0        0
FC      No          0         0         0       0         0         0        0
FD      No          0         0         0       0         0         0        0
FE      No          0         0         0       0         0         0        0
```

… and in z/OS RMF

```
                                         C H A N N E L   P A T H   A C T I V I T Y
                                                                                                                          PA
          z/OS V1R2                SYSTEM ID CB88            DATE 07/22/2001            INTERVAL 22.54.336
                                   RPT VERSION V1R2 RMF      TIME 15.37.05              CYCLE 1.000 SECONDS

IODF = 01   CR-DATE: 05/10/2000   CR-TIME: 21.00.01   ACT: POR          MODE: LPAR        CPMF: EXTENDED MODE

-------------------------------------------------------------------------------------------------------------------------
                                                   OVERVIEW FOR DCM-MANAGED CHANNELS
-------------------------------------------------------------------------------------------------------------------------

    CHANNEL         UTILIZATION(%)    READ(MB/SEC) WRITE(MB/SEC)
   GROUP  G NO    PART  TOTAL    BUS   PART  TOTAL   PART  TOTAL

   FC_SM  1  8   15.36  55.86   6.00  15.36  60.00  15.36  60.36
   FCV_M    12   30.00  45.00   5.00  45.00  50.00  45.00  50.00
   CNC_M     1   17.23  34.45

-------------------------------------------------------------------------------------------------------------------------
                                                      DETAILS FOR ALL CHANNELS
-------------------------------------------------------------------------------------------------------------------------

  CHANNEL PATH     UTILIZATION(%)    READ(MB/SEC) WRITE(MB/SEC)     CHANNEL PATH     UTILIZATION(%)    READ(MB/SEC) WRITE(
ID TYPE   G SHR  PART  TOTAL    BUS   PART  TOTAL   PART  TOTAL   ID TYPE   G SHR  PART  TOTAL    BUS   PART  TOTAL   PART

78 CVC_P        OFFLINE                                           80 CTC_S        OFFLINE
79 CNC_S        OFFLINE                                           81 CNC_S         0.04   0.04
7A FC     1  Y  20.00  30.00   5.00  20.00  30.00  20.00  50.00   82 FC        Y  20.00  30.00   6.00  20.00  30.00  20.00
7B FC_SM     Y  15.36  55.86   6.00  15.36  60.00  15.36  60.36   83 FC     1  Y  15.36  55.66   7.00  15.36  60.00  15.36
7C FCV       Y  10.00  30.00   5.00  10.00  50.00  10.00  50.00   84 FCV       Y  10.00  30.00   5.00  10.00  50.00  50.00
7D FCV_M     Y  30.00  45.00   5.00  45.00  50.00  45.00  50.00   85 FCV       Y  30.00  45.00   6.00  45.00  50.00  45.00
7E CNC_M        17.23  34.45                                      86 CNC_S         0.00   0.00
7F CNC_S        OFFLINE                                           8C CNC_S         0.00   0.00


    CHANNEL PATH      WRITE(B/SEC)     MESSAGE RATE     MESSAGE SIZE  SEND FAIL    RECEIVE FAIL
  ID TYPE   G SHR      PART   TOTAL     PART   TOTAL     PART   TOTAL     PART       PART   TOTAL

  AB IQD       Y     645.12M 2500.2G   850.23K 4.2K    760.12   779.56    12         85     120
```

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)
- "#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

- 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
  ...

- ... 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
  ...

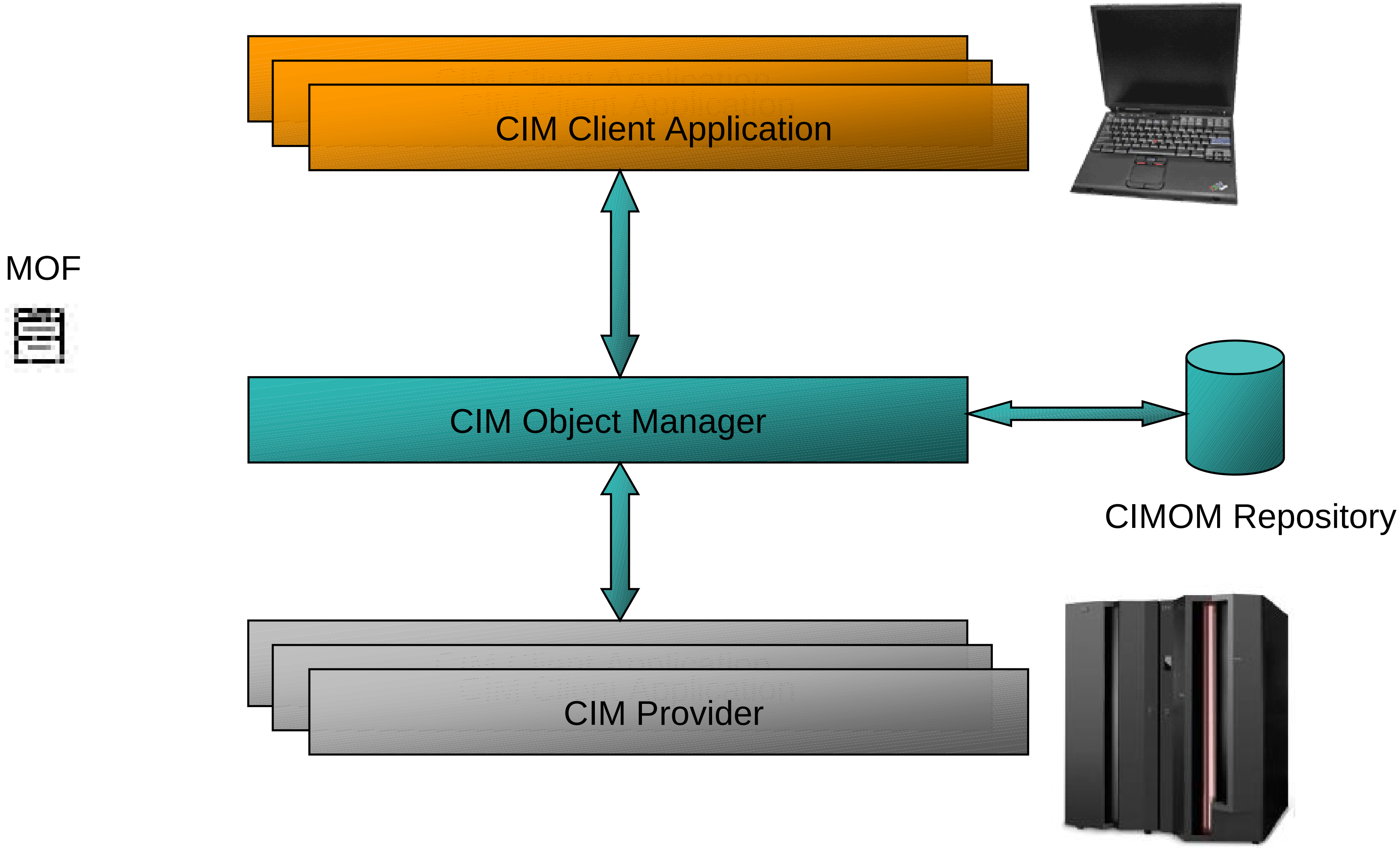
The NET-SNMP Project

- 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/*

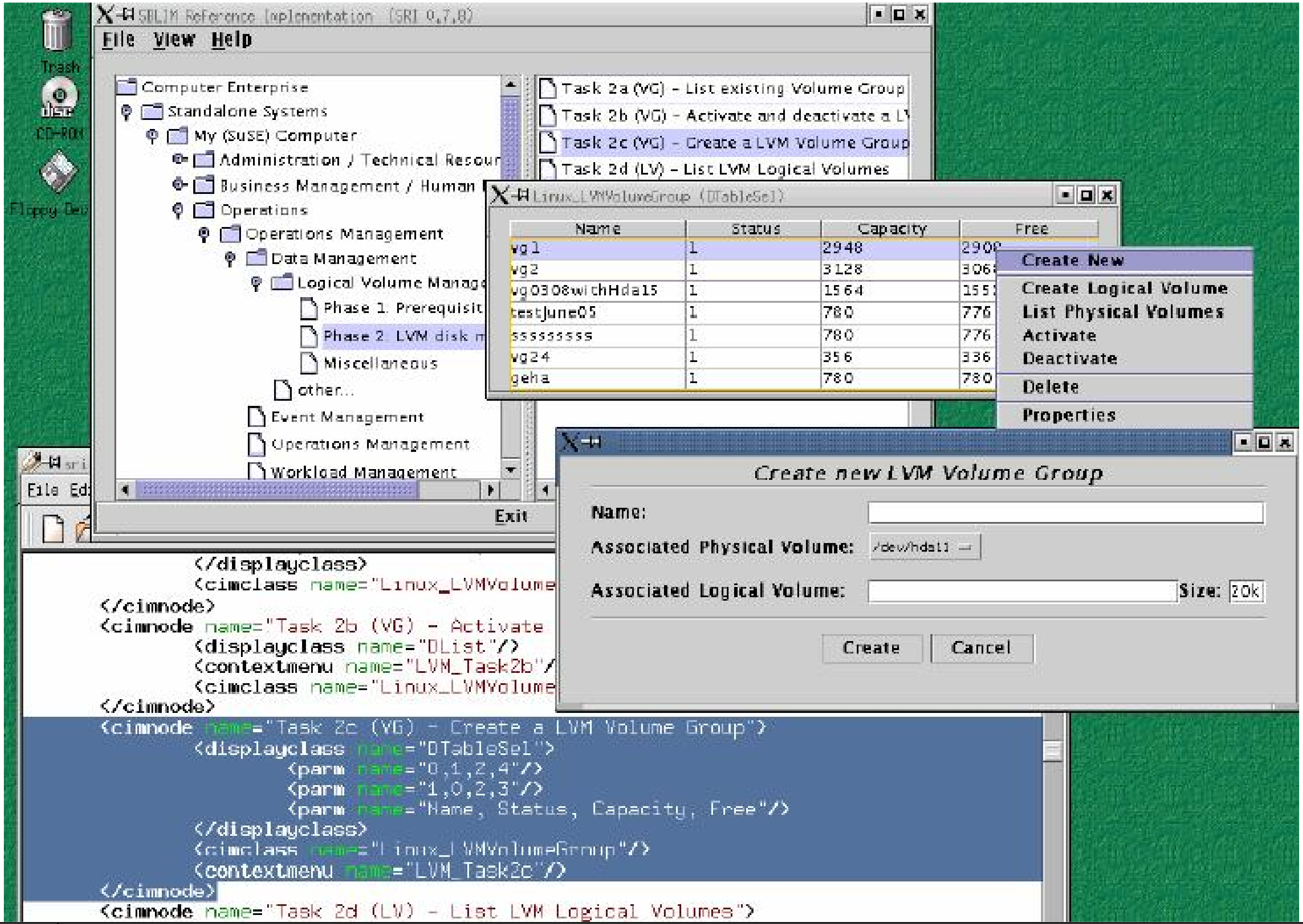
CIM Provider, Object Manager, and Client



SBLIM

- 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.
- SBLIM is an Open-Source WBEM instrumentation project; see *http://oss.software.ibm.com/developerworks/oss/sblim/*
- 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

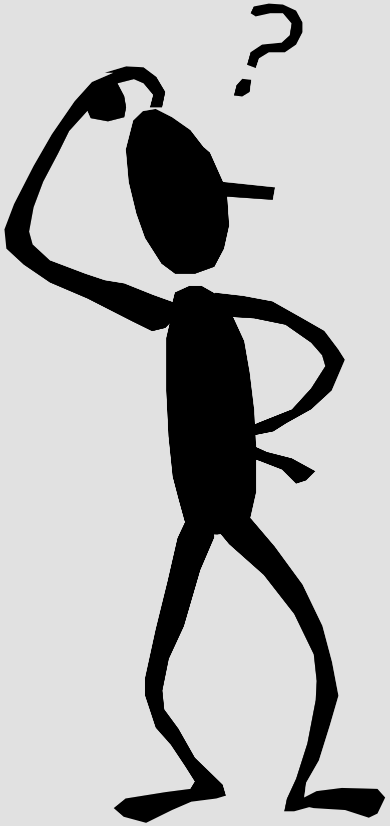
SBLIM Reference Implementation



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
- Jason R Fink & Matthew D Sherer: "Linux Performance Tuning and Capacity Planning", SAMS 2001, ISBN 0-672-32081-9
- *http://www.vm.ibm.com/perf/*
- *http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmlin.htm*

Questions?



Email:
benke@de.ibm.com