Linux on System z Performance Hints and Tips

Thomas Weber (tweber@de.ibm.com)
WAVV Conference 2008
Chattanooga, TN, April 18-22

Trademarks

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

DB2*
DB2 Connect
DB2 Universal Database
e-business logo
IBM*
IBM eServer
IBM logo*
Informix®

System z
Tivoli*
WebSphere*
z/VM*
zSeries*
z/OS*

ECKD
Enterprise Storage Server®
FICON
FICON Express
HiperSocket
OSA
OSA Express

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

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

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

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

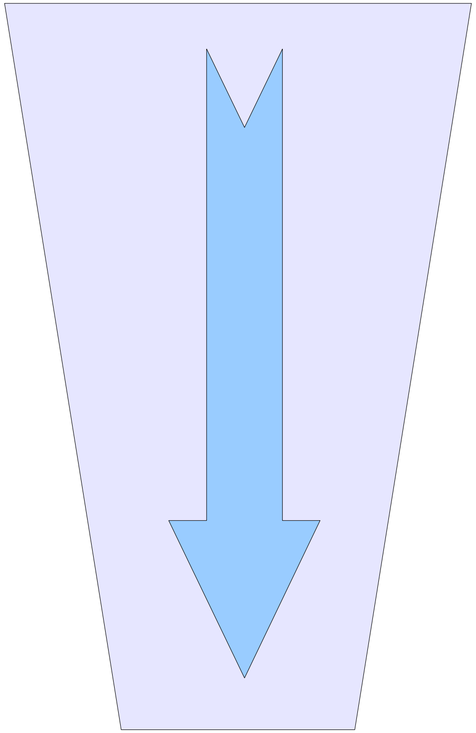
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- Tuning
  - Application
    - C/C++
  - Middleware
    - Java
  - Linux
    - Networking
  - Virtualization
  - Hardware / Setup
- Monitoring
  - Linux
  - z/VM

Optimize your stack in the right direction

- Diminishing effect of tuning efforts
  - Application design
  - Application implementation
  - Middleware
  - Operating system
  - Virtualization layer
  - Hardware



Optimizing C/C++ code with gcc

- **use -O3 optimization as default**
  - **no debugging options**

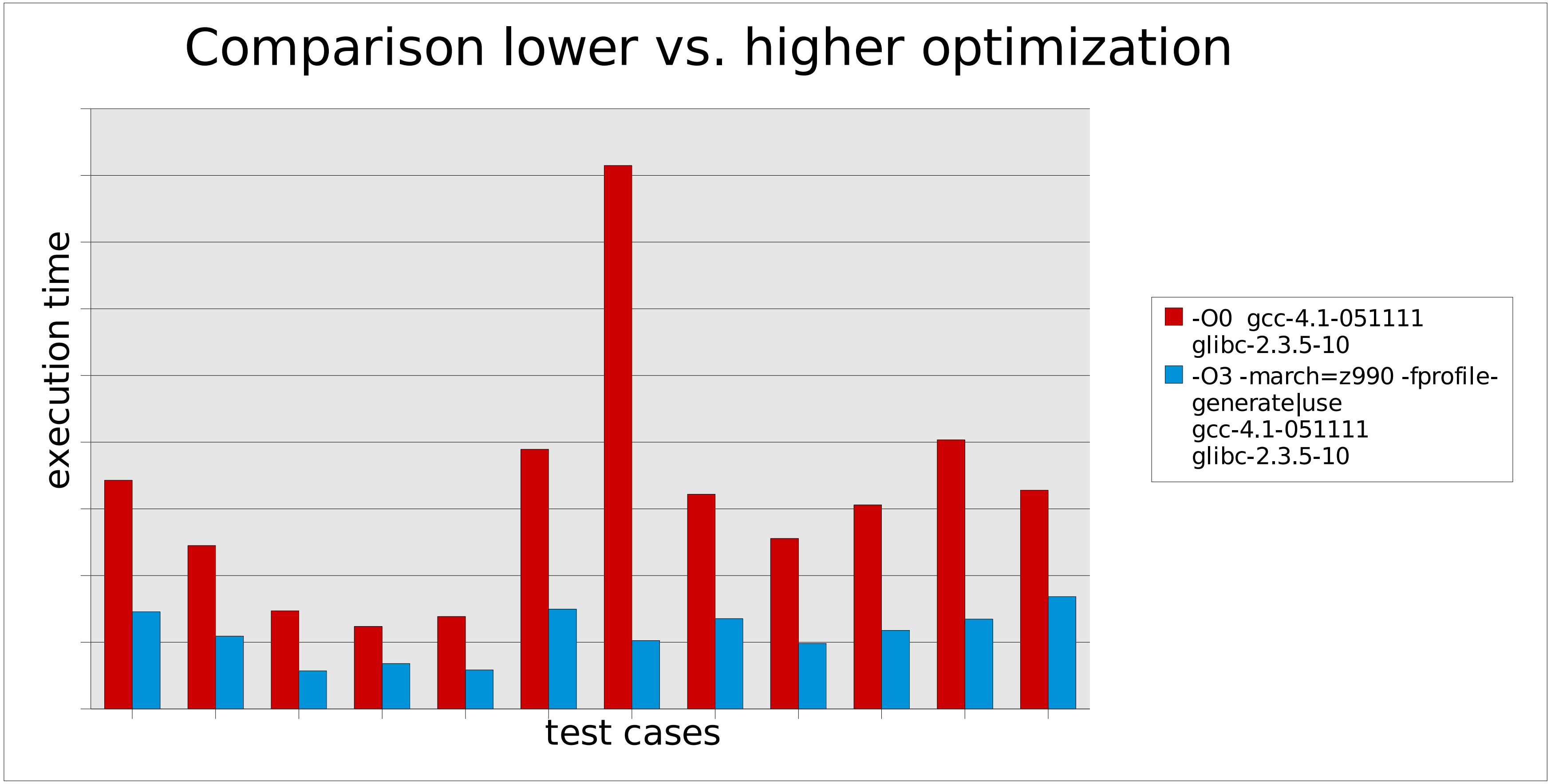
- **further optimization:**
  - **architecture dependent options**
    - **-march=values <G5,z900,z990> <z9-109 with gcc-4.1>**
    - **-mtune=values <G5,z900,z990> <z9-109 with gcc-4.1>**
  - **inline assembler functions**

- **next step: application design**
  - **dynamic or static linking**
  - **avoid –fPIC for executables**
  - **right use of inlined C / C++ functions**

- **fine tuning: additional general options on a file by file basis**
  - **-funroll-loops -ffast-math**

Results of changing compiler options

- **Using -O3 instead of no optimization shortens runtime to 50% or less**



Java basics

- try to use latest Java version
  - up to 60% release to release improvements
  - true as well for newer Service Releases (SR)

- make sure that you've got enabled JIT
  - 'java -version' tells “JIT enabled/disabled”

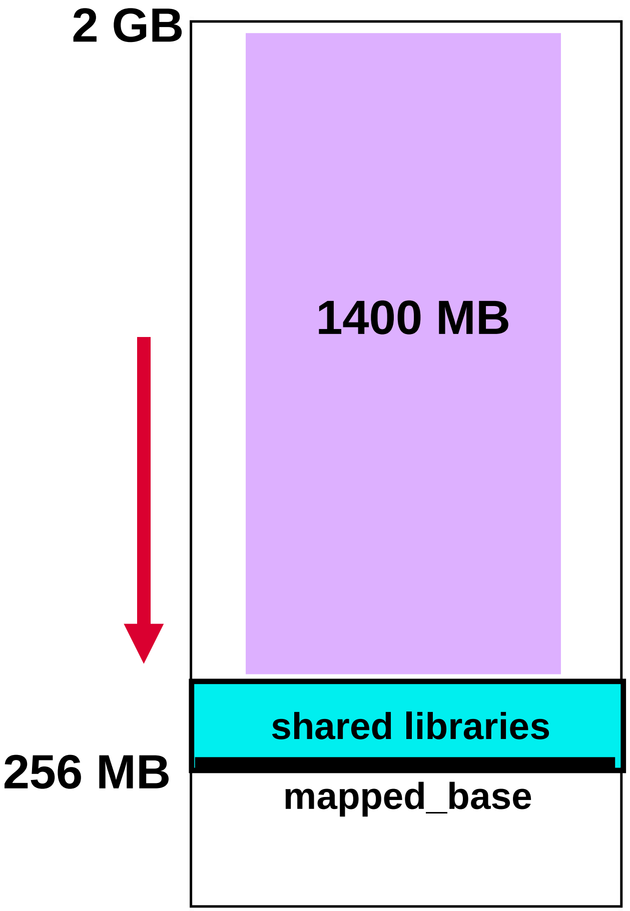
- don't use Java in batch mode:
  - if you do 100 calls “java MyProg”, you interpret MyProg 100 times
    - can take more CPU power than the program itself
    - the JIT compiler can't do such a optimization
  - instead pull the loop inside the Java program and call “java MyProg” once

Java heap size

- useful parameters
  - Setting heap size: -Xms (minimal), -Xmx (maximal), use min=max
  - -verbose:gc -- monitor GC
- max heap <= available memory
  - this avoids paging - Linux and z/VM
  - remember: heap memory will be used eventually!
- a larger heap size usually implies better performance
  - in 31bit SLES8, SLES9 & SLES10 use '/proc/<pid>/mapped_base' to define heaps greater than 1 GB
  - in 31-bit RHEL4 environments use flex-mmap mechanism and prelink
    - watch out for prelinked applications!
  - works also in 31-bit emulation mode on a 64-bit Linux

Mapped_base HOWTO

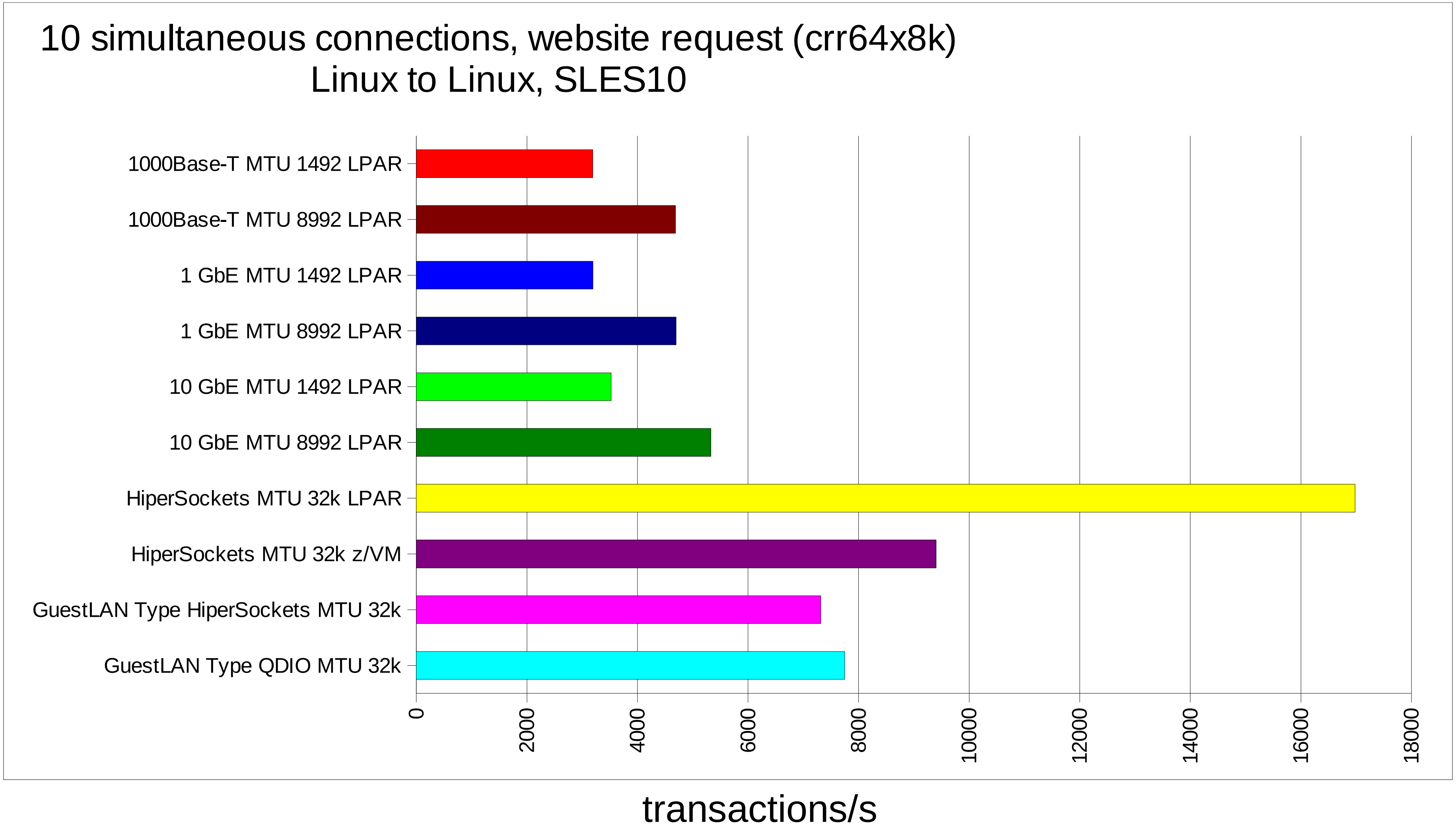
- applies only to Novell SLES8,9,10 distributions (31-bit)
- PID is the process ID of the process you want to change
  - $$ within bash gives you the current shell PID, you can use /proc/self/... as well
- display memory map of any PID by
  `cat /proc/<PID>/maps`
- check the mapped_base value by
  `cat /proc/<PID>/mapped_base`
- lower the value to e.g. 256 MB by
  `echo 268435456 >/proc/<PID>/mapped_base`



Networking performance

- external Connectivity:
  - use new 10 GbE cards with MTU 8992
  - attach OSA directly to Linux guest image
- internal Connectivity:
  - Hipersockets for LPAR-LPAR communication
  - VSWITCH, Guest LAN for VM guest-guest communication
- for really busy network devices consider to
  - use channel bonding – load balancing
  - increase the number of inbound buffers for the qeth driver
    - Device has to be offline
    - `# echo <number> > /sys/bus/ccwgroup/drivers/qeth/<device_bus_id>/buffer_count`
- channel bonding for High Availability(HA) creates only a small overhead
- choose your MTU size carefully
  - avoid fragmentation, lots of small packages increase CPU costs

Network throughput for various connection types



Networking throughput overview (SLES10)

	Online transaction (rr200x1000)	Database query (rr200x32k)	Website access (crr64x8k)	File transfer (strp, strg 20Mx20)
Advantage of large MTU size over default MTU size	equal	1.2x (1 GbE), 2.1x (10 GbE)	1.5x	3.4x (only 10 GbE)
Advantage of 10 GbE over 1 GbE	1.1x	1.9x (large MTU)	1.1x	3.3x (large MTU)
Advantage of GuestLAN over OSA	1.8x	3.4x (1 GbE), 1.7x (10 GbE)	1.4x	4.5x (1 GbE), 1.3x (10 GbE)
Fastest connection	HiperSockets LPAR	HiperSockets LPAR	HiperSockets LPAR	HiperSockets LPAR

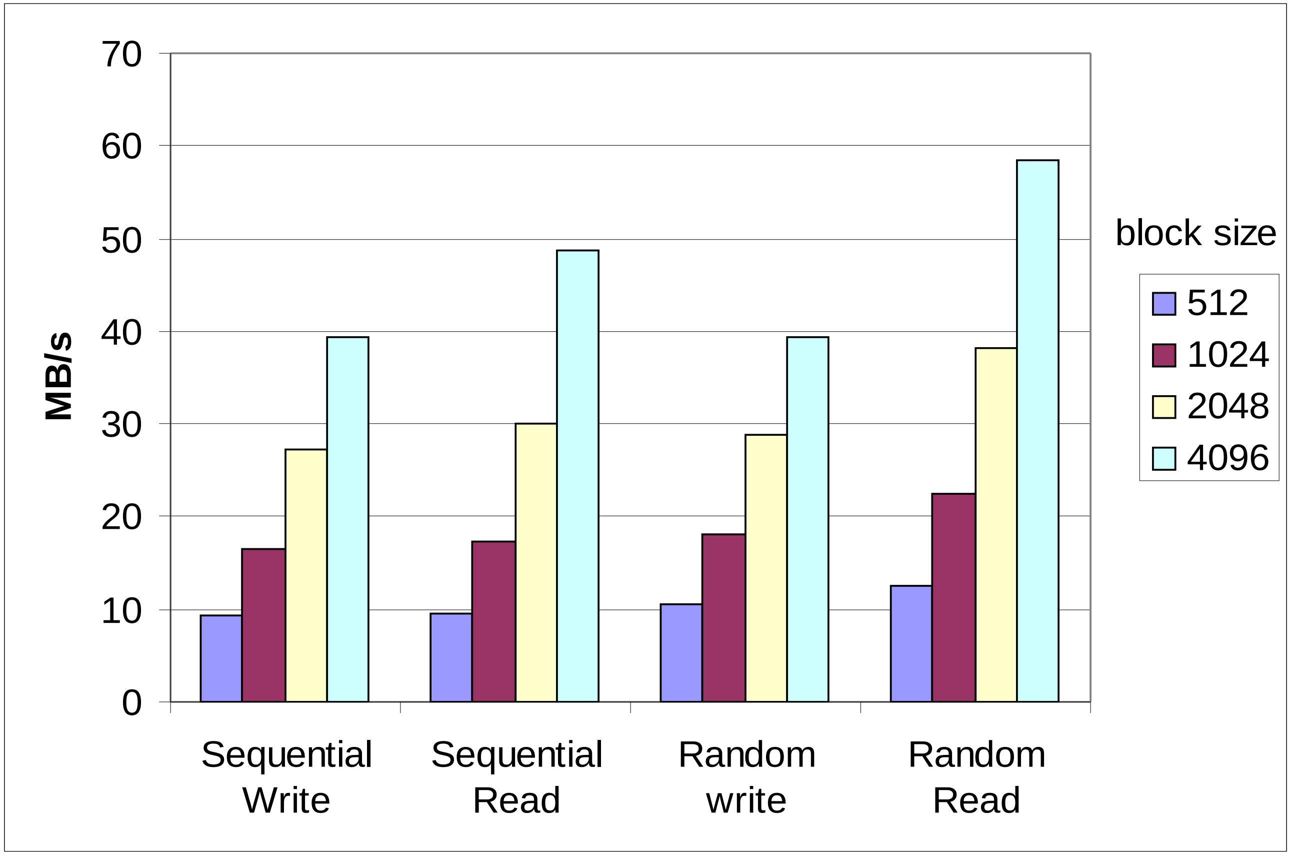
How to improve disk performance

- **Hardware choices**
  - **use SCSI instead of ECKD**
  - **use FICON instead of ESCON**
    - **4Gb FICON > 2Gb FICON > 1Gb FICON**
- **utilize your Hardware**
  - **use “striped” Logical Volumes(LV) from different storage server ranks**
  - **consider using PAV**
  - **carefully set up your storage system**

- **ww.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_optimizedisk.shtml**

Effect of dasdfmt block size on throughput and capacity

- **use 4k block size on ECKD DASDs whenever possible !**



dasdfmt blocksize	usable disk space
512b	3.5G
1024b	4.7G
2048b	6G
4096b	6.8G

z/VM 2 GB considerations

- Solution: upgrade z/VM to 5.2 or 5.3 level
- Read at
  - http://www.vm.ibm.com/perf/tips/2gstorag.html
  - http://www.vm.ibm.com/perf/reports/zvm/html/64bit.html
  - http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_fixe
- Old workarounds
  - Cooperative Memory Management
  - fixed I/O buffers with kernel 2.6 and ECKD
  - distribute your guests to multiple z/VMs
  - Move large guest to LPAR

“On Demand Timer” patch

- Linux uses a timer tick based interrupt mode (timer interrupts)
- Timer interrupts for idle guests create unnecessary overhead
- since SLES8: behaviour is configurable
  - /proc/sys/kernel/hz_timer (default 0; true for LPAR and z/VM)
  - 1 = timer interrupts occurring every 10 ms
  - 0 = timer interrupts generated on demand only
- included in SLES9, SLES10 and RHEL4, RHEL5 s390/s390x distributions

spin_retry

- **Problem:**
  - with many guests in z/VM it can happen that CP is busy executing diagnose instructions for the guest
- **What's behind it?**
  - in a so-called spin lock, Linux guests give their CPU share back to the hipervisor using a DIAG 44 instruction
  - Hipervisor can be overloaded
- **Solution:**
  - Linux tries to get a lock n-times before issuing a DIAG instruction
  - value of n is adjustable in /proc/sys/kernel/spin_retry (default 1000)
  - included in latest SLES9 + SLES10 + RHEL4 + RHEL5

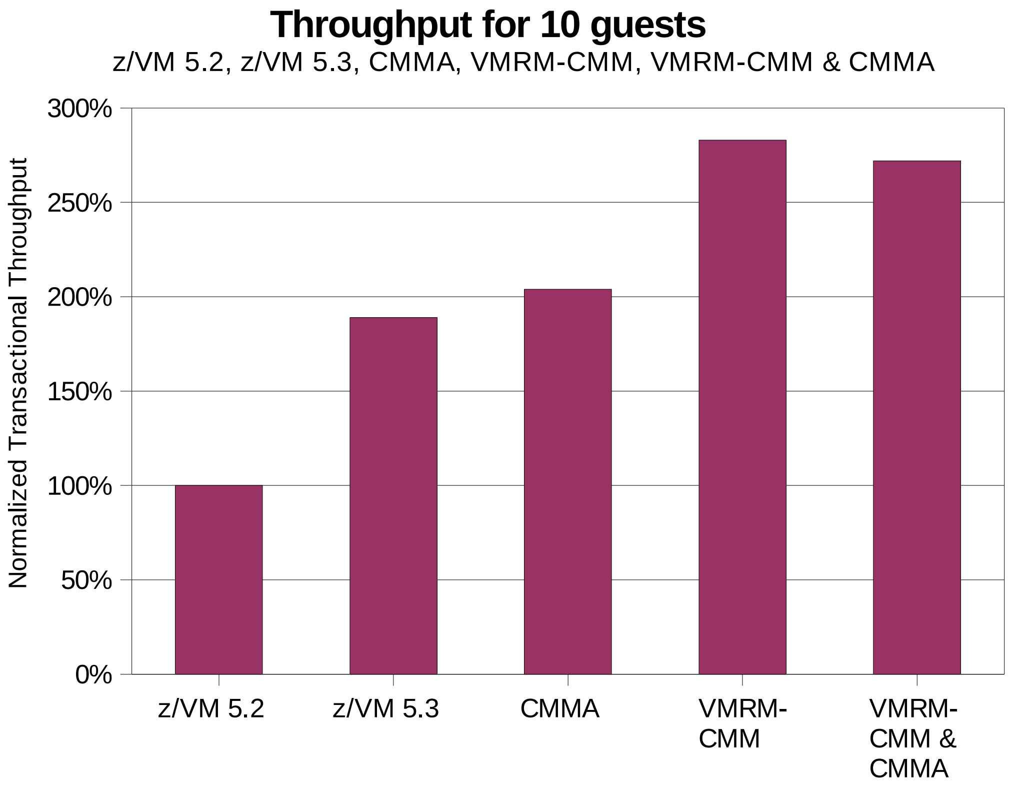
VM Resource Manager - CMM

- 2 methods available:
  - VMRM-CMM (VM Resource Manager – Cooperative Memory Management) aka CMM1
    - Resource manager controls the size of the guests
  - CMMA (Collaborative Memory Management Assist) aka CMM2
    - Linux indicates which pages don't need to be saved
- both methods show performance improvements when z/VM hits a system memory constraint

- www.vm.ibm.com/sysman/vmrm/vmrmcmm.html

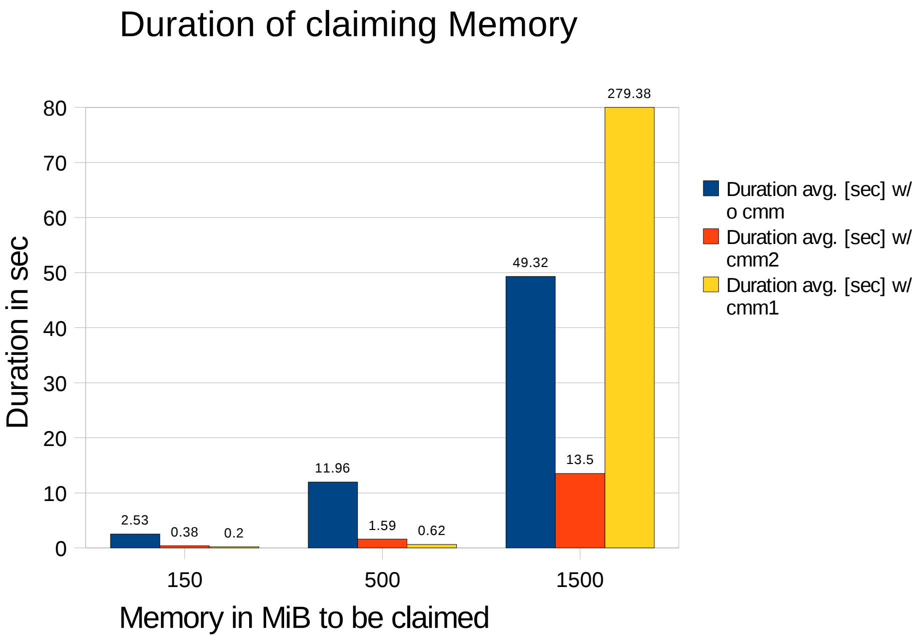
CMM1 scenario

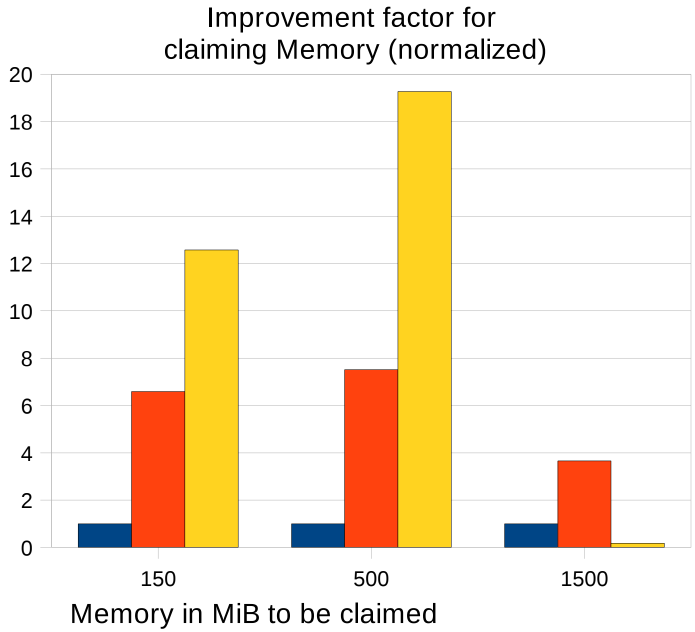
- **large Oracle guests, total used Linux memory = 2x of z/VM central storage, OLTP workload**
- **advantages with VMRM-CMM aka CMM1**
- **guests did not suffer from less page cache**



CMM2 scenario

- **Workload**
  - **15 guests, touching all their memory, all z/VM storage used. A guest orders now 150MB, 500MB, 1.5GB of memory. We measured the duration of this operation**
- **Result**
  - **in case of sudden memory claims CMM2 is the best choice**





# of CPUs per Linux image

- Use as few virtual CPUs as possible

- For LPAR definitions:
  - # all virtual CPUs : # real CPUs <= 4:1

- For z/VM:
  - #of guest CPUs <= #of CPUs for VM (LPAR)

- You don't get more done by defining more CPUs!

Agenda

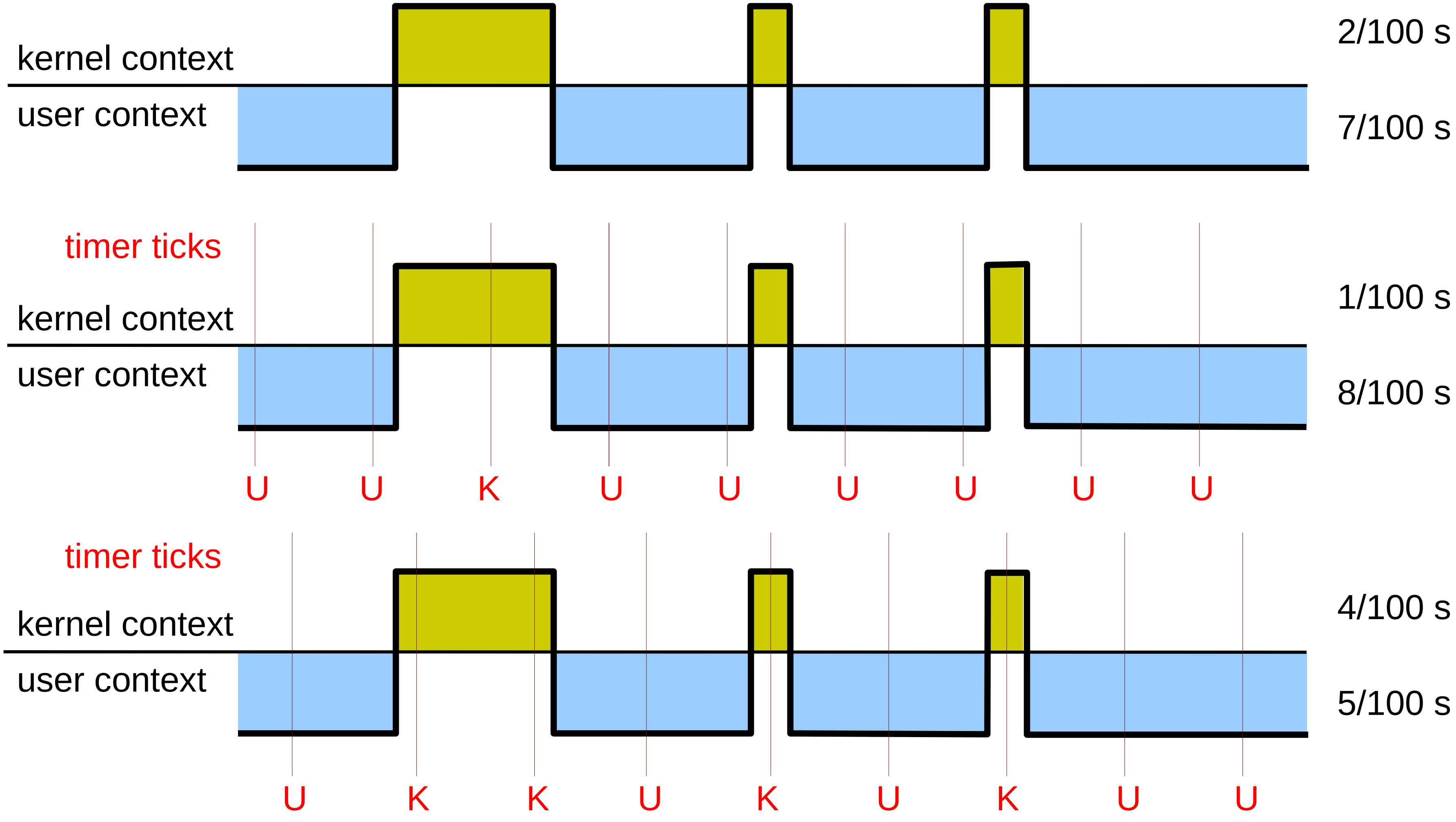
- Tuning
  - Application
    - C/C++
  - Middleware
    - Java
  - Linux
    - Disk I/O
    - Networking
  - Virtualization
  - Hardware / Setup
- **Monitoring**
  - **Linux**
  - **z/VM**

Linux command 'top' – the real time view

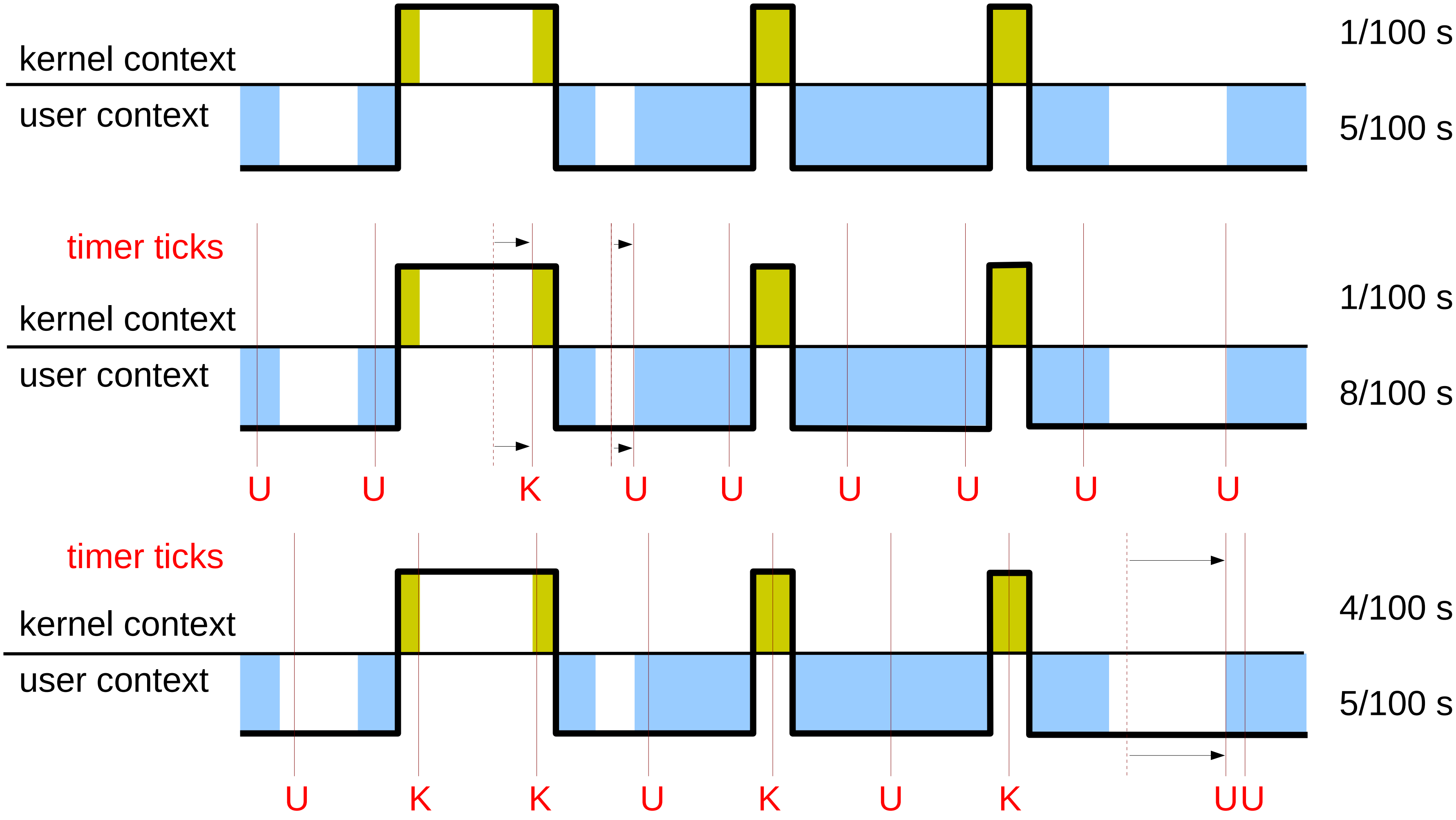
- adds new field “CPU steal time”
  - the time Linux wanted to run on a CPU, but the hipervisor was not able to schedule CPU
  - true for LPAR and z/VM hipervisor
  - included in SLES10 and RHEL5

```
top - 09:50:20 up 11 min,  3 users,  load average: 8.94, 7.17, 3.82
Tasks:  78 total,   8 running,  70 sleeping,   0 stopped,   0 zombie
 Cpu0 : 38.7%us,  4.2%sy,  0.0%ni,  0.0%id,  2.4%wa,  1.8%hi,  0.0%si, 53.0%st
 Cpu1 : 38.5%us,  0.6%sy,  0.0%ni,  5.1%id,  1.3%wa,  1.9%hi,  0.0%si, 52.6%st
 Cpu2 : 54.0%us,  0.6%sy,  0.0%ni,  0.6%id,  4.9%wa,  1.2%hi,  0.0%si, 38.7%st
 Cpu3 : 49.1%us,  0.6%sy,  0.0%ni,  1.2%id,  0.0%wa,  0.0%hi,  0.0%si, 49.1%st
 Cpu4 : 35.9%us,  1.2%sy,  0.0%ni, 15.0%id,  0.6%wa,  1.8%hi,  0.0%si, 45.5%st
 Cpu5 : 43.0%us,  2.1%sy,  0.7%ni,  0.0%id,  4.2%wa,  1.4%hi,  0.0%si, 48.6%st
Mem:    251832k total,   155448k used,    96384k free,     1212k buffers
Swap:   524248k total,    17716k used,   506532k free,    18096k cached
```

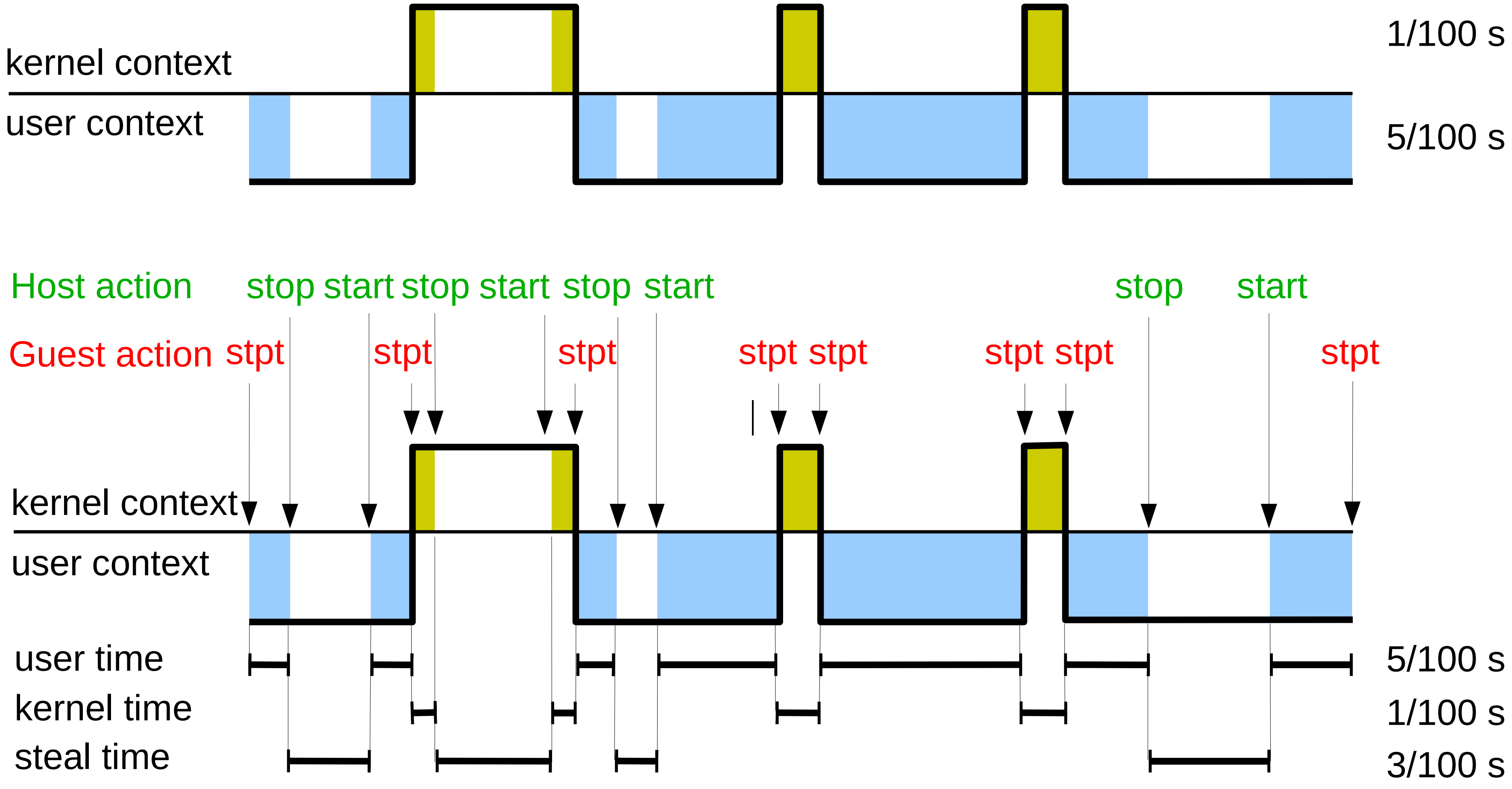
Tick based CPU Time inaccuracy (1)



Tick based CPU accounting on virtual systems (2)



New Virtual CPU time accounting (3)



stpt = Store CPU Timer

Sysstat – the 'long' term data collection

- contains four main tools
  - sadc: data gatherer - stores data in binary file
  - sar: reporting tool - reads binary file and converts it to readable output
  - mpstat: processor utilization
  - iostat: I/O utilization
- “steal time” included starting version 7.0.0
- install the sysstat package and configure it depending on your distribution (e.g. crontab)
  - by default data is collected in /var/log/sa
- more info at: http://perso.orange.fr/sebastien.godard
  and with “man sar” on your system

OProfile – the Open Source sampling tool (1)

- OProfile offers profiling of all running code on Linux systems, providing a variety of statistics.
  - by default, kernel mode and user mode information is gathered for configurable events
- System z hardware currently does not have support for hardware performance counters, instead timer interrupt is used
  - Enable the hz_timer(!)
- the timer is set to whatever the jiffy rate is and is not user-settable
- Novell / SUSE: OProfile is on the SDK CDs
- More info at:
  - http://oprofile.sourceforge.net/docs/
  - http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-gu

OProfile – short HOWTO (2)

- **Example from** **http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_tools**

```
sysctl -w kernel.hz_timer=1

gunzip /boot/vmlinux-2.6.5-7.201-s390x.gz


opcontrol --vmlinux=/boot/vmlinux-2.6.5-7.201-s390x

opcontrol --start


<DO TEST>


opcontrol --shutdown

opreport
```

OProfile – output example (3)

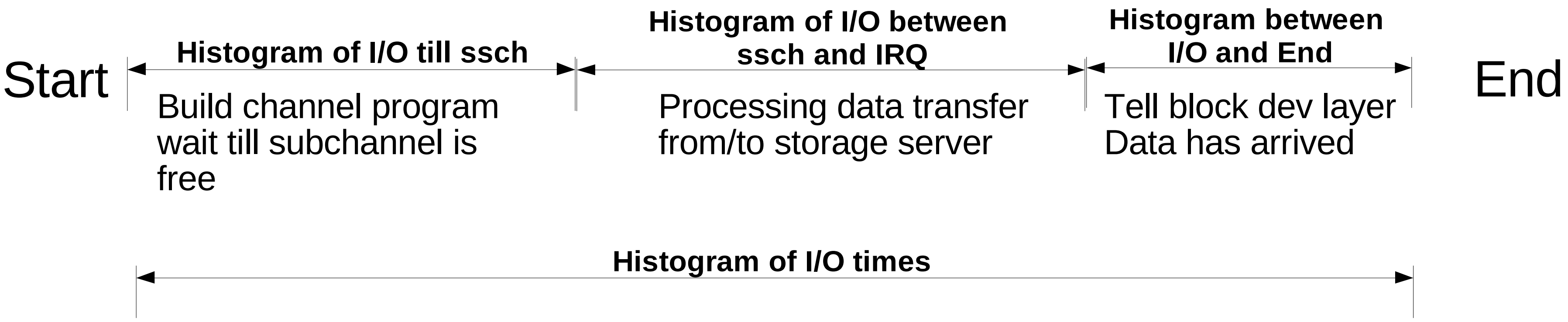
```
CPU: CPU with timer interrupt, speed 0 MHz (estimated)
Profiling through timer interrupt
vma      samples %        app name           symbol name
80002840 5862     34.8970  mcf_base.z_Linux    price_out_impl
800012c8 5221     31.0811  mcf_base.z_Linux    refresh_potential
80003cb4 4398     26.1817  mcf_base.z_Linux    primal_bea_mpp
80003b60 408       2.4289  mcf_base.z_Linux    sort_basket
0001a67c 345       2.0538  vmlinux             default_idle
800013d8 138       0.8215  mcf_base.z_Linux    flow_cost
800033bc 98        0.5834  mcf_base.z_Linux    update_tree
800020f8 88        0.5239  mcf_base.z_Linux    dual_feasible
800036a4 72        0.4286  mcf_base.z_Linux    primal_iminus
8000323c 40        0.2381  mcf_base.z_Linux    write_circulations
80002720 24        0.1429  mcf_base.z_Linux    insert_new_arc
```

/proc/dasd/statistics (1)

- Linux can collect performance statistics on DASD activity as seen by Linux(!)
- turn on with
  ```
  echo on > /proc/dasd/statistics
  ```
- turn off with
  ```
  echo off > /proc/dasd/statistics
  ```
- to reset: turn off and then on again
- can be read for the whole system by
  ```
  cat /proc/dasd/statistics
  ```
- can be read for individual DASDs by
  ```
  tunedasd -P /dev/dasda
  ```

/proc/dasd/statistics (2)

- **collects statistics (mostly processing times) of IO operations**
- **each line represents a histogram of times for a certain operation**
- **operations split up into the following :**


Start
Histogram of I/O till ssch
Build channel program wait till subchannel is free
Histogram of I/O between ssch and IRQ
Processing data transfer from/to storage server
Histogram between I/O and End
Tell block dev layer Data has arrived
End
Histogram of I/O times

http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_tools_dasd.html

/proc/dasd/statistics (3)

```
Tue Jan 18 20:52:50 EST 2005
21155901 dasd I/O requests
with 433275376 sectors(512B each)
   __<4    ___8    __16    __32    __64    _128    _256    _512    __1k    __2k    __4k    __8k    _16k    _32k    _64k    128k
   _256    _512    __1M    __2M    __4M    __8M    _16M    _32M    _64M    128M    256M    512M    __1G    __2G    __4G    _>4G
Histogram of sizes (512B secs)
      0       0 3774298  838941  352193  232188   43222   30563   16163    1403       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O times (microseconds)
      0       0       0       0       0       0       0       2 3005329  352056  726353  671293  355198  147238   29245    2201
     51       3       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O times per sector
      0       0   24686  204678  524222 2803252  500319  537993  249088  316175  111592   15932    1005      26       3       0
      0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O time till ssch
3498191   51615   86168   21601    2756    1927    4348   22793  177758  138465  955964  214188   61200   42284    9075     621
     14       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O time between ssch and irq
      0       0       0       0       0       0       0       4 4252115  408592   78374  122000  309317  108290    9848     416
     13       3       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O time between ssch and irq per sector
      0       0   41819  517428  890743 3323127   21897   23329  103966  280533   79777    6056     282      10       2       0
      0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Histogram of I/O time between irq and end
4531949  633301   75411   41903    4984     791     516      48      40       3       3      20       0       0       0       0
      0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
# of req in chanq at enqueuing (1..32)
      0 3658672  277906  128989   97542 1125789      27       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0       0       0       0       0      0       0
```

SCSI statistics (1)

- **in SLES9 and SLES10 SCSI statistics can be collected**
- **the parameter `CONFIG_STATISTICS=y` must be set in the kernel config file**
- **if debugfs is mounted at `/sys/kernel/debug/`, all the statistics data collected can be found at `/sys/kernel/debug/statistics/`**
- **the names of these subdirectories consist of `zfcp-<device-bus-id>` for an adapter and `zfcp-<device-bus-id>-<WWPN>-<LUN>` for a LUN**

http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_dasd_scsiIO.html

SCSI statistics (2)

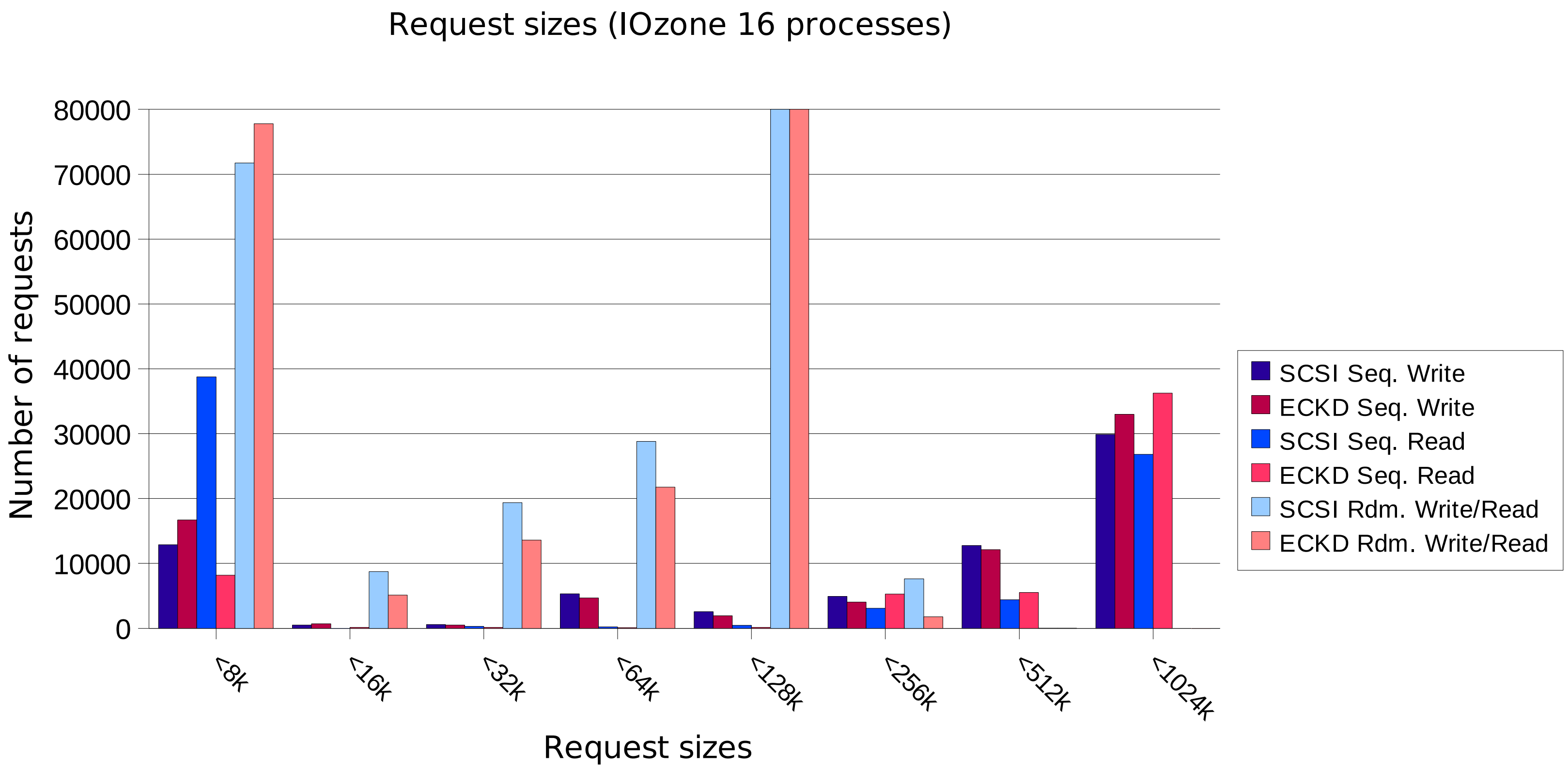
- each subdirectory contains two files, a data and a definition file
- using
  ```
  echo on=1 > definition
  ```
  the data gathering can be switched on for each device,
- with
  ```
  echo on=0 > definition
  ```
  the gathering is switched off again. It defaults to data gathering being turned off.
- the command
  ```
  echo data=reset > definition
  ```
  enables you to reset the collected data to 0

SCSI statistics example (3)

```
cat /sys/kernel/debug/statistics/zfcp-0.0.1700-0x5005076303010482-0x4014400500000000/data
...
request_sizes_scsi_read 0x1000 1163
request_sizes_scsi_read 0x80000 805
request_sizes_scsi_read 0x54000 47
request_sizes_scsi_read 0x2d000 44
request_sizes_scsi_read 0x2a000 26
request_sizes_scsi_read 0x57000 25
request_sizes_scsi_read 0x1e000 25
request_sizes_scsi_read 0x63000 24
request_sizes_scsi_read 0x6f000 19
request_sizes_scsi_read 0x12000 19
...
latencies_scsi_read <=1 1076
latencies_scsi_read <=2 205
latencies_scsi_read <=4 575
latencies_scsi_read <=8 368
latencies_scsi_read <=16 0
...
channel_latency_read <=16000 0
channel_latency_read <=32000 983
channel_latency_read <=64000 99
channel_latency_read <=128000 115
channel_latency_read <=256000 753
channel_latency_read <=512000 106
channel_latency_read <=1024000 141
channel_latency_read <=2048000 27
channel_latency_read <=4096000 0
...
fabric_latency_read <=1000000 1238
fabric_latency_read <=2000000 328
fabric_latency_read <=4000000 522
fabric_latency_read <=8000000 136
fabric_latency_read <=16000000 0
...
```

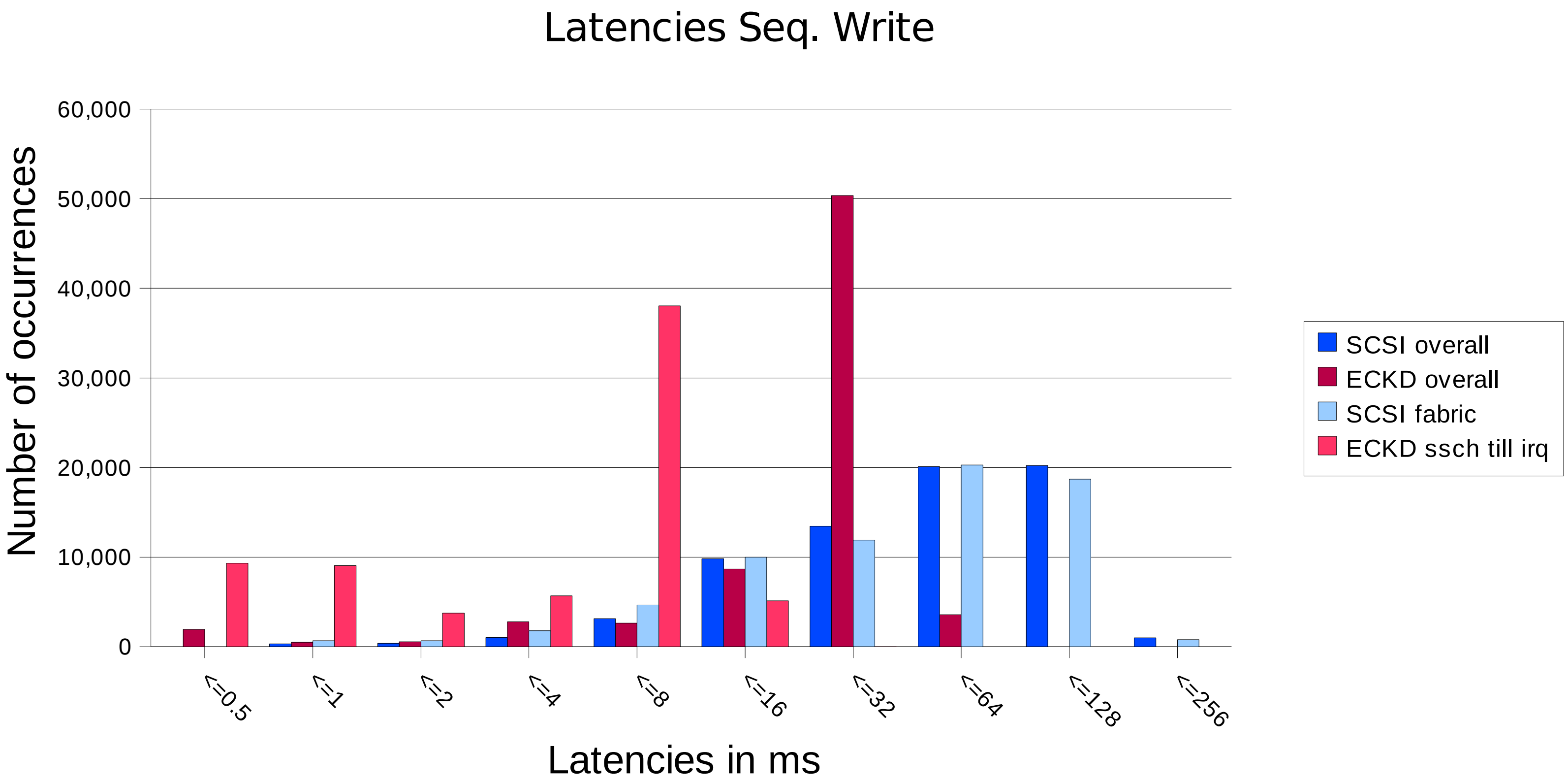
Comparing SCSI and ECKD request sizes

- **similar request sizes for sequential and random I/O**



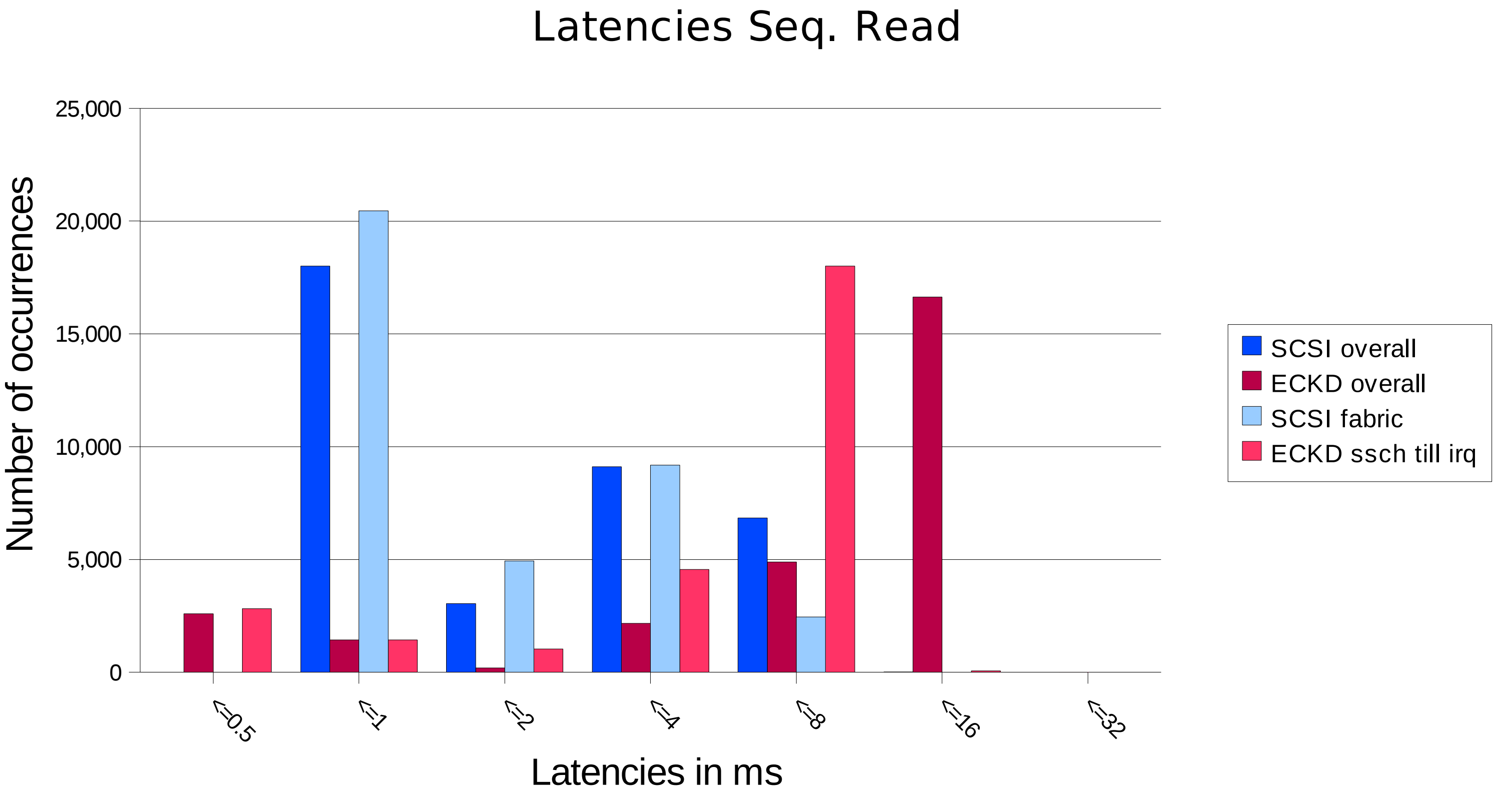
Comparing SCSI and ECKD latencies (1)

- **SCSI sequential write latencies are longer**



Comparing SCSI and ECKD latencies (2)

- **SCSI sequential read latencies are shorter**



How to collect z/VM monitor data

- **Cheat Sheet at: http://www.vm.ibm.com/perf/tips/collect.html**

- **5 basic steps**
  - **create monitor DCSS**
  - **setup userid to issue monwrite command**
  - **start and configure monitor**
  - **start monwrite**
  - **stop monwrite and save data**

How to insert Linux data in z/VM monitor stream

- enable your guest for inserting data into the monitor stream
  - set APPLMON option to user direct
- insert Linux modules
  - `modprobe appldata_mem`
  - `modprobe appdata_os`
  - `modprobe appldata_net_sum`
- turn on monitoring
  - `echo 1 > /proc/sys/appldata/timer`
  - `echo 1 > /proc/sys/appldata/mem`
  - `echo 1 > /proc/sys/appldata/os`
  - `echo 1 > /proc/sys/appldata/net_sum`
- details can be found in chapter 15 of Device Drivers, Features, and Commands (SC33-8281-02)
  http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26bdd02.pdf

Visit us !

- **Linux on zSeries Tuning Hints and Tips**
  - **http://www.ibm.com/developerworks/linux/linux390/perf/**
- **Linux-VM Performance Website:**
  - **http://www.vm.ibm.com/perf/tips/linuxper.html**

Questions