Linux on System z performance update

Eberhard Pasch
epasch@de.ibm.com

Session 2590
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

- System z hardware
- z10 performance and support
  - File server
  - Compiler and DFP
  - Database
  - Java
- CPU hotplug
- Disk I/O
  - 4Gbps FICON and FCP
  - SCSI multipathing
  - Striped volumes
- Cryptographic support
  - CEX2A and CPACF
  - WebSeal
- Networking
  - Connection overview
  - Throughput / cost
IBM System z – system design comparison

System I/O Bandwidth

288 GB/sec

172.8 GB/sec

96 GB/sec

24 GB/sec

Memory

1.5 TBs

16-way

54-way

Balanced System CPU, nWay, Memory, I/O Bandwidth*

ITRs for 1-way

~900

~600

450

300

512 GB

256 GB

64 GB

32-way

54-way

64-way CPUs

z10

z9 EC

zSeries 990

zSeries 900

*Servers exploit a subset of its designed I/O capability
z10 Performance: DBench (file server workload)

- Improvement with z10 versus z9:
  - For 1 or 2 CPUs = 1.75x, for 8 CPUs = 1.5x (see below)
Compiler - System z features

- **System z9 109 and z9 ec | bc processor support (gcc-4.1)**
  - Exploit instructions provided by the extended immediate facility
  - Selected via `-march=z9-109 / -mtune=z9-109`

- **System z10 processor support (> gcc-4.3)**
  - Exploit instruction new to z10
  - Selected via `-march=z10 / -mtune=z10`

- **Overall integer performance enhancement on z9**
  - 8% comparing gcc-3.4 and gcc-4.1 on System z
  - 5.9% comparing gcc-4.1 and gcc-4.2 on System z
  - gcc-4.3 is work in progress

- **Decimal floating point support - DFP**
  - Software DFP support (gcc-4.2) for older machines without hardware DFP support
  - Hardware DFP support for newer machines support (gcc-4.3)
z10 performance: compiler workloads

- Overall improvement with z10 versus z9: 1.9x
- Work in progress with gcc-4.3 compiler using -march=z10 option
DFP - decimal floating point performance on z10

- Testcase: 1 million telephone bills
  - On z9: hardware DFP needs 1/5 of the runtime of software DFP
  - On z10: hardware DFP needs 1/9 of the runtime of software DFP
  - On z10 the test runs 2.3x/3.8x faster than on z9 (software DFP/hardware DFP)
z10 Performance: Java workload

- Overall improvement with z10 versus z9: 1.65x
z10 with Informix IDS 11 OLTP workload

- **Throughput improvements**
  - z9 to z10: 65% to 82%
  - x numbers of z10 CPUs can do the same work as 2x z9 CPUs

![Diagram showing throughput and scaling factor](image-url)

- Transactions
- Scaling factor

- Number of CPUs:
  - 1
  - 2
  - 4
  - 8
  - 12
  - 16

- Throughput:
  - z990
  - z9
  - z10

- Scaling factor:
  - 0.0
  - 2.0
  - 4.0
  - 6.0
  - 8.0
  - 10.0
  - 12.0
  - 14.0

© 2008 IBM Corporation
CPU hotplug

- Changes the number of used processors on the fly, depending on the current overall utilization
- The control information is stored at `/etc/sysconfig/cpuplugd`
- Minimum number of CPUs is set with `cpu_min="<number>"`
- Maximum number of CPUs is set with `cpu_max="<number>"`
- The update interval is set with `update="<value in seconds>"`
- The rule for increasing the number of CPUs is `HOTPLUG="(loadavg > onumcpus + 0.75) & (idle < 10.0)`
- The rule for decreasing the number of CPUs is `HOTUNPLUG="(loadavg < onumcpus - 0.25) | (idle > 50)`
Performance results with CPU hotplug

- Improvements in case where the default (high) number of CPUs is not needed
- Up to 40% more throughput, up to 40% CPU cost savings
Configuration for 4Gbps disk I/O measurements

2094

8 FICON
8 FCP

4 GBIT FICON/FCP SWITCH

8 FICON
8 FCP

DS8K

4 Gbit FICON Port
4 Gbit FCP Port
Disk I/O performance with 4Gbps links - FICON

- Strong throughput increase (average 1.6x)
- The best increase is with sequential read at 2x

Compare FICON 4 GBit - 2 GBit

Throughput in MB/sec

8 channels on 4 cards 8 DS8K3 ports on 8 cards 4GBit

8 channels on 4 cards 8 DS8K3 ports on 8 cards 2GBit
Disk I/O performance with 4Gbps links - FCP

- Moderate throughput increase
- Best improvement with sequential read at 1.25x

Compare FCP 4 GBit - 2 GBit

Throughput in MB/sec

- SQ Write
- SQ RWrite
- SQ Read
- R Read
- R Write

8 channels on 4 cards 8 DS8K ports on 8 cards 4GBit
8 channels on 4 cards 8 DS8K ports on 8 cards 2GBit
Disk I/O performance with 4Gbps links – FICON / FCP

- Throughput for sequential write is similar
- FCP throughput for random I/O is 40% higher

Compare FICON to FCP - 4 GBit

![Bar chart comparing FICON and FCP throughput for different types of I/O operations.](image-url)
Disk I/O performance with 4Gbps links – FICON versus FCP / direct I/O

- Bypassing the Linux page cache improves throughput for FCP up to 2x, for FICON up to 1.6x.
- Read operations are much faster on FCP

Compare FICON to FCP - 4 GBit
FCP/SCSI single path versus multipath

- Use failover instead of multibus

Throughput for sequential readers

- Throughput in MB/s
- Number of processes
- 0 200 400 600 800 1000 1200 1400 1600
- 8 16 32 64

- single path LVM2
- failover LVM2
- multibus LVM2
- Costs for failover are about between 10% and 15%
- Costs for multibus vary

Relative CPU cost per transferred data
sequential read

<table>
<thead>
<tr>
<th>Number of processes</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative cost in %</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- single path LVM2
- failover LVM2
- multibus LVM2
Striped volumes (1)

LVM striped volume

Storage pool striped volume – DS8KR3 with 1GB stripe size
## Striped volumes (2)

<table>
<thead>
<tr>
<th>Striping is done in</th>
<th>LVM striping</th>
<th>DS8000 storage pool striping</th>
</tr>
</thead>
<tbody>
<tr>
<td>effort to construct the volume</td>
<td>take care of picking subsequent disks from different ranks</td>
<td>configure storage server</td>
</tr>
<tr>
<td>administrating disks within Linux</td>
<td>can be challenging, e.g. several hundred for a database</td>
<td>simple</td>
</tr>
<tr>
<td>volume extendable ?</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>maximum I/O request size</td>
<td>stripe size (e.g. 64KB)</td>
<td>maximum provided by the device driver (e.g. 512KB)</td>
</tr>
<tr>
<td>multipathing</td>
<td>SCSI: assign paths round robin to disks, multipath failover ECKD: path group</td>
<td>SCSI: multipath multibus, ECKD: path group</td>
</tr>
<tr>
<td>usual disk sizes</td>
<td>Lv = many disks SCSI 10GB to 20GB, ECKD mod9 or mod27</td>
<td>Volume = 1 disk, SCSI unlimited, e.g. 300GB, ECKD max. mod54</td>
</tr>
<tr>
<td>extent pool</td>
<td>1 rank</td>
<td>multiple ranks</td>
</tr>
<tr>
<td>maximum number of ranks for the constructed volume</td>
<td>total number of ranks</td>
<td>Total number of one server side (50%)</td>
</tr>
</tbody>
</table>
Striped volumes – results and recommendation

- **General pros / cons**
  - Storage pool striped volumes are as simple to set up and to administrate as a few large disks
  - Striping on the storage device lowers CPU consumption (LVM) on the Linux side
  - Stripe size is 1 GB
  - Rank failure will hit all disks

- **Results with ECKD disks**
  - Combined with HiperPAV reaches nearly the same performance as Linux solution
  - Without HiperPAV there can only be one IO outstanding per DASD, which limits the performance
  - FICON path groups doing the load balancing

- **Results with SCSI disks**
  - Linux striped logical volumes are faster but the logical volume manager takes more CPU cycles than e.g. the multipath daemon
  - For random workloads the multipath daemon used to distribute workload to the FCP channels needs improvements (work in progress)

- **If you don't use striping in Linux today, consider to enable it at least in the storage server – your performance won't become worse**
Cryptographic hardware support - SSL stack
Crypto Express2 accelerator (CEX2A) - SSL handshakes

- The number of handshakes is up to 4x higher with HW support
- In the 32 connections case we save about 50% of the CPU resources
Crypto Express2 Accelerator (CEX2A) and CPACF

- The use of both hardware features leads to 3.5x more throughput
- Using software encryption costs about 6x more CPU

![Bar chart showing normalized throughput and CPU cost per throughput with and without hardware features.]
Cryptographic hardware support for a WebSphere environment – using WebSEAL

- WebSEAL provides an authentication and authorization mechanism
  - based on Tivoli Access Manager
  - enables an end-to-end Single Sign On (SSO) solution for secure transactions for WebSphere application servers residing on z/OS).
Crypto performance – WebSEAL SSL access

- The connection from the client to the WebSEAL server runs encrypted using SSL (AES-128)
- Scaling the size of the requested page
- Uses mostly CPACF
- Improvement up to factor 2.4 for hardware encryption versus software encryption

![Graph showing improvement by hardware crypto support](image-url)

- Throughput normalized to software encryption
- Page size:
  - 5.8KB
  - 12KB
  - 2.9MB

<table>
<thead>
<tr>
<th>Page size</th>
<th>Software encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8KB</td>
<td>100%</td>
</tr>
<tr>
<td>12KB</td>
<td>150%</td>
</tr>
<tr>
<td>2.9MB</td>
<td>300%</td>
</tr>
</tbody>
</table>
- SLES10 SP2 / z10
- 200 byte request
- 32k response
- 10 connections
- y axis is #trans
- larger is better

- Use large MTU size
- 10 GbE is there
- Hipersockets between LPARs
- VSwitch inside VM
- Direct OSA for outside connection of demanding guests

<table>
<thead>
<tr>
<th>Network Type</th>
<th>MTU Size</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 Base-T MTU 1492 LPAR</td>
<td>1492</td>
<td>1000 Base-T MTU 8992 LPAR</td>
</tr>
<tr>
<td>1 GbE MTU 1492 LPAR</td>
<td>1492</td>
<td>1 GbE MTU 8992 LPAR</td>
</tr>
<tr>
<td>10 GbE MTU 1492 LPAR</td>
<td>1492</td>
<td>10 GbE MTU 8992 LPAR</td>
</tr>
<tr>
<td>HiperSockets MTU 32k LPAR</td>
<td>32k</td>
<td>1000 Base-T MTU 1492 z/VM</td>
</tr>
<tr>
<td>1000 Base-T MTU 8992 z/VM</td>
<td>8992</td>
<td>1 GbE MTU 1492 z/VM</td>
</tr>
<tr>
<td>1000 Base-T MTU 8992 z/VM</td>
<td>8992</td>
<td>1 GbE MTU 8992 z/VM</td>
</tr>
<tr>
<td>1 GbE MTU 1492 z/VM</td>
<td>1492</td>
<td>1 GbE MTU 8992 z/VM</td>
</tr>
<tr>
<td>1 GbE MTU 1492 z/VM</td>
<td>1492</td>
<td>10 GbE MTU 1492 z/VM</td>
</tr>
<tr>
<td>10 GbE MTU 1492 z/VM</td>
<td>1492</td>
<td>10 GbE MTU 8992 z/VM</td>
</tr>
<tr>
<td>10 GbE MTU 8992 z/VM</td>
<td>8992</td>
<td>HiperSockets MTU 32k z/VM</td>
</tr>
</tbody>
</table>

- GuestLAN Type HiperSockets MTU 32k
- GuestLAN Type QDIO MTU 8992
- VSwitch MTU 1492
- VSwitch MTU 8992
- 1 GbE MTU 1492 z/VM->LPAR
- 1 GbE MTU 8992 z/VM->LPAR
- 10 GbE MTU 1492 z/VM->LPAR
- 10 GbE MTU 8992 z/VM->LPAR
- VSwitch 1 GbE MTU 1492 z/VM->LPAR
- VSwitch 1 GbE MTU 8992 z/VM->LPAR
- VSwitch 10 GbE MTU 1492 z/VM->LPAR
- VSwitch 10 GbE MTU 8992 z/VM->LPAR

- Use large MTU size
- 10 GbE is there
- Hipersockets between LPARs
- VSwitch inside VM
- Direct OSA for outside connection of demanding guests
- SLES10 SP2 / z10
- 200 byte request
- 32k response
- 10 connections
- y axis is server cost per transaction
- smaller is better

- Use large MTU size
- 10 GbE is there
- VSwitch inside VM
- VSwitch for outgoing connections has it's price
SLES10 SP2 / z10
200 byte request
1000 byte response
10 connections
y axis is #trans
larger is better

10 GE better than 1 GbE
Hipersockets between LPARs
VSwitch inside VM
VSwitch little bit slower for outside connections
SLES10 SP2 / z10

200 byte request

1000 byte response

10 connections

y axis is server cost per transaction

smaller is better

VSwitch best z/VM option for outside connection

Internal connections are expensive

MTU size makes no difference

VSwitch best z/VM option for outside connection
Visit us!

- Linux on System z: Tuning Hints & Tips
- Linux-VM Performance Website:
Questions