Linux on System z
Distribution Performance Update

Christian Ehrhardt
IBM Research and Development Germany

12th August 2011
Session 10015
Agenda

• Performance Evaluation Results
  • Environment
  • Improvements
  • Changes one should be aware of

• Performance evaluation Summary
  • Improvements and degradations per area
  • Summarized comparison
Environment

- Host – System z10
  - FICON 4 Gbps
  - FCP 4 Gbps
  - HiperSockets
  - OSA Express 3 1GbE + 10GbE

- Storage – DS8300 (2107-922)
  - FICON 4 Gbps
  - FCP 4 Gbps

- HW-Platform
  - Linux on LPAR
  - Linux in z/VM 5.4 guest

- Verified on
  - System z Enterprise
  - DS8800 with 8 Gbps connectivity
  - Linux in z/VM 6.1
Compared Versions

- Compared Set 1
  - RHEL5 U4 (2.6.18-194.el5)
  - RHEL6-GA (2.6.32-71.el6)
  - RHEL6-GA + tuning
    - our recommended tuning (in RH Tech Notes)
    - workarounds for known issues

- Compared Set 2
  - SLES10-SP3 (2.6.16.60-0.54.5-default)
  - SLES11-GMC (2.6.27.19-5-default)
  - SLES11-SP1-GMC (2.6.32.12-0.6-default)
    - workarounds for known issues

- Measurements
  - Base regression set covering most customer use cases as good as possible
  - Focus on areas where performance issues are more likely
  - Just the top level summary, based on thousands of comparisons
  - Special case studies for non-common features and setups

- Terminology
  - Throughput – “How much could I transfer once?”
  - Latency – “How long do I have to wait for event X?”
  - Normalized cpu consumption - “How much cpu per byte do I need?”
New process scheduler (CFS)

- Goals of CFS
  - Models “ideal, precise multi-tasking CPU”
  - Fair scheduling based on virtual runtime

- Changes you might notice when switching from O(1) to CFS
  - Lower response times for I/O, signals, …
  - Balanced distribution of process time-slices
  - Improved distribution across processors
  - Shorter consecutive time-slices
  - More context switches

- Improved balancing
  - Topology support can be activated via the topology=on kernel parameter
  - This makes the scheduler aware of the cpu hierarchy

- You really get something from fairness as well
  - Improved worst case latency and throughput
  - By that CFS can ease QoS commitments
Benchmark descriptions
File system / LVM / Scaling

- Filesystem benchmark dbench
  - Emulation of Netbench benchmark
  - Generates file system load on the Linux VFS
  - Does the same I/O calls like smbd server in Samba (without networking calls)

- Simulation
  - Workload simulates client and server (Emulation of Netbench benchmark)
  - Mainly memory operations for scaling
  - Low main memory and LVM setup for mixed I/O and LVM performance
  - Mixed file operations workload for each process: create, write, read, append, delete
  - 8 CPUs, 2 GiB memory and scaling from 4 to 62 processes (clients)
  - Measures throughput of transferred data
File system benchmark – process scaling

- Improved scalability
  - Especially improves large workloads
  - Lower cross process deviation improves QoS
- Increased CPU consumption due to
  - CFS is striving for better interactivity and fairness
  - Changes affecting the writeback of dirty pages
    - Rule of thumb – now about twice as aggressive
    - One might want to tune dirty ratios in `/proc/sys/vm/dirty_*`
- Comparison between SLES10 SP3 and SLES11 SP1 looks similar
Benchmark descriptions - Java

• evaluates server side Java
  • 3-tier system
    • Random input from user
    • Middle tier business logic implemented in Java
    • No explicit database --> emulated by Java objects
    • Scales warehouses

• stressed components
  • Java
    • Virtual Machine (VM)
    • Just-In-Time compiler (JIT)
    • Garbage Collection (GC)
  • Linux operating system
    • Threads
    • Scheduler
    • Caches and Memory
Java

- 64b – RHEL6-GA -4.8% vs RHEL5-U4; tuned at least only -3.8%
- This is caused by a bit of over-optimization for desktop latency in the new scheduler
- System z recommended tunables are not set by default in RHEL6, but part of the tech notes
- RHEL6.1 had our recommendation by default and CFS fixes, now almost equal to RHEL5-U4
- SLES11 SP1 showed always slightly better throughput versus SLES10 SP3
Benchmark descriptions - Webserving

- **Webserver Benchmark**
  - Static website content read
  - Variable number of connections
  - Measures throughput via network connection

- **Server side**
  - Apache
  - HTML content

- **Client side**
  - 3 clients connected to webserver
  - Number of active requests scaled from 1 to 20 connections per client
Webserving – example for improved CPU scaling

Simple Webserving Workload (60 Clients)

- Improved CPU scalability compared to RHEL5-U4
  - The recommended scheduler tuning adds further improvements
- Additional CFS effect
  - Lower worst-case response time
Benchmark descriptions - Network

- Network Benchmark which simulates several workloads
- Transactional Workloads
  - 2 types
    - RR – A connection to the server is opened once for a 5 minute time frame
    - CRR – A connection is opened and closed for every request/response
  - 4 sizes
    - RR 1x1 – Simulating low latency keepalives
    - RR 200x1000 – Simulating online transactions
    - RR 200x32k – Simulating database query
    - CRR 64x8k – Simulating website access
- Streaming Workloads – 2 types
  - STRP/STRG – Simulating incoming/outgoing large file transfers (20mx20)
- All tests are done with 1, 10 and 50 simultaneous connections
- All that across on multiple connection types (different cards and MTU configurations)
Network Throughput

Gigabit Ethernet OSA Express3 MTU 8992

- Single connection Latency can be an issue, but it is much better than in SLES11
  - 1x1 is shown here as it forces max overhead and latency per transferred byte

- Connection scaling is good - parallel scenarios improved a lot
  - Workloads with larger transferred sizes benefit a bit more

- For HiperSockets even latency improved
Network CPU consumption

Gigabit Ethernet OSA Express 3 MTU 8992

HiperSockets MTU 32k

- CPU consumption increased for a lot of workloads
  - roughly 2/3 of the connection types we distinguish are affected
  - Part of the trade-offs for better performance
  - Also partially a scheduler/caching effect
- Some improvements for loads with large mtu's
  - Usually seen on the sender side
  - That implies it is beneficial for data sources not for data sinks
Benchmark descriptions - Disk I/O

- **Workload**
  - Threaded I/O benchmark
  - Each process writes or reads to a single file, volume or disk
  - Can be configured to run with and without page cache (direct I/O)
  - Operating modes: Sequential write/rewrite/read + Random write/read

- **Setup**
  - Main memory was restricted to 256 MiB
  - File size (overall): 2 GiB, Record size: 64KiB
  - Scaling over 1, 2, 4, 8, 16, 32, 64 processes
  - Sequential run: write, rewrite, read
  - Random run: write, read (with previous sequential write)
  - Once using bypassing the page cache
  - Sync and Drop Caches prior to every invocation
Page cache based disk I/O read issue

- Caused as corner case by memory management “improvements”
- Real World - Backups
  - It can hold a lot of data to scan by the backup software (→ a lot of seq. read)
  - A lot of data is usually split across many discs on s390 (→ concurrent access)
  - Overcommitment/ballooning effects or sized too small (→ memory constraint)

- Detection
  - Most workloads won't see the impact or more than that benefit from these changes
  - Check sysstat which should report a huge amount of pgscand/s
  - Run “sync; echo 3 > /proc/sys/vm/drop_caches”
    - Should hurt throughput, huge improvements mean you are probably affected

- Workarounds other than “more memory”
  - Drop caches if there is a single time this happens (e.g. on nightly backup)
  - Use direct I/O or shrink read ahead if applicable
  - Fix got upstream accepted in 2.6.37-rc1
    - will appear in both distributions next service update
Disk I/O – New FICON features

• HyperPAV
  • Avoid subchannel busy
  • Automatic management of subchannel assignment/usage
  • No need of multipath daemon
  • Especially useful for concurrent disk accesses

• Read-Write Track Data
  • Allows to read/write up to a full track in one command word
  • Especially useful for huge requests and streaming sequential loads

• High Performance Ficon
  • New metadata format reduces overhead
  • Especially useful for small requests
Disk I/O – FICON – HyperPAV

- Using 4 disks (4 ranks) with 3 aliases per rank
- Without PAV/HyperPAV
  - Access could become contented (subchannel busy)
  - Throughput stays constant >1 proc per disk
- Solution: multiple subchannels per device
  - PAV: Aliases for devices
  - HyperPAV: Pool of aliases defined per rank
  - Throughput increased up to 3.5 x in our scenario
→ Usage of HyperPAV can be highly recommended
Disk I/O – FICON – effect of RWTD/HPF – Throughput

Throughput for initial writers

Throughput for readers

- IOzone sequential write/read using direct I/O
  - Huge throughput improvements
    - Write throughput up to 26%
    - Read throughput up to 82%
  - Normalized I/O consumption stays about the same
    - despite the much larger throughput
- SLES11 SP1 to SLES10 SP3 comparison looks similar
Disk I/O – FICON – effect of RWTD/HPF – random workloads

• IOzone random write/read using direct I/O
  • Huge throughput improvements
    • Read throughput up to +81%
    • Write throughput up to +23%

• Where throughput isn't improved usually cpu consumption drops

• SLES11 SP1 to SLES10 SP3 comparison looks similar
Disk I/O – Multipathing

- In the Past: Required for RAS (failover)
- Now: recommendable for RAS+Perf (multibus)
  - Throughput
  - Lower latency
  - Utilize multiple Adapters
  - CPU consumption sane

- Example of a single Database I/O case
  - 32 processes doing random 8KiB writes
  - From worst to best setup throughput 13 times faster

- Huge topic
  - two presentations just about these setups
  - check out our webcasts
Hints - General

• Cgroup memory support
  • This is a feature coming with newer kernels
  • Recommended by some management tools to enforce very customizable memory constraints
  • Has a rather large footprint by consuming 1% of the memory
  • Activated by default
  • In a consolidation environment it is actually 1% multiplied by your virtual/real ratio
  • Not pageable by Linux, but fortunately by z/VM
  • This can be overridden with a kernel parameter (reboot):
    \texttt{cgroup\_disable=memory}
### Improvements and Degradations of RHEL6 per area

<table>
<thead>
<tr>
<th>Improvements</th>
<th>Degradations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FICON I/O</td>
<td>CPU consumption*</td>
</tr>
<tr>
<td>Process scaling</td>
<td>OSA single C. Latency</td>
</tr>
<tr>
<td>CPU scaling</td>
<td>I/O corner cases via page cache*</td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
</tr>
<tr>
<td>Multiconn. Networking</td>
<td></td>
</tr>
<tr>
<td>Disk I/O via page cache</td>
<td></td>
</tr>
</tbody>
</table>

- Improvements in almost every area
  - Especially for large workloads
- Degradations for corner cases and cpu consumption
  - * = Partially or completely avoidable due to tunings/workarounds
Summary for RHEL6

- RHEL 6 performance is good
  - With some trade-offs roughly equal to RHEL5-U4
    - A common trade-off is increased cpu consumption for better scalability
  - Our recommended tunings/workarounds help in some known cases
    - Upcoming RHEL6.1 will further reduce the amount of manual tuning needed
  - Almost generally recommendable
    - An exception are very cpu consumption sensitive environments
      - *Here upgrades have to be considered carefully*

- Improvements and degradations

<table>
<thead>
<tr>
<th>Base</th>
<th>New</th>
<th>Improved</th>
<th>No difference or Trade-off</th>
<th>Degraded</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH5U4</td>
<td>RH6</td>
<td>27</td>
<td>22</td>
<td>33</td>
</tr>
<tr>
<td>RH5U4</td>
<td>RH6 tune &amp; w.</td>
<td>34</td>
<td>48</td>
<td>0</td>
</tr>
</tbody>
</table>
# Improvements and Degradations of SLES11 SP1 per area

## vs. SLES10SP3

<table>
<thead>
<tr>
<th>Improvements</th>
<th>Degradations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FICON I/O</td>
<td>CPU consumption*</td>
</tr>
<tr>
<td>Process scaling</td>
<td>OSA single C. Latency</td>
</tr>
<tr>
<td>CPU scaling</td>
<td>I/O corner cases via page cache*</td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
</tr>
<tr>
<td>Multiconn. Networking</td>
<td></td>
</tr>
<tr>
<td>Disk I/O via page cache</td>
<td></td>
</tr>
</tbody>
</table>

## vs. SLES11

<table>
<thead>
<tr>
<th>Improvements</th>
<th>Degradations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk I/O</td>
<td>Only corner cases</td>
</tr>
<tr>
<td>Process scaling</td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
</tr>
<tr>
<td>Latency</td>
<td></td>
</tr>
<tr>
<td>CPU consumption</td>
<td></td>
</tr>
<tr>
<td>Multiconn. Networking</td>
<td></td>
</tr>
</tbody>
</table>

- **Improvements in almost every area**
  - Especially for large workloads
- **Degradations for corner cases and cpu consumption**
  - * = Partially or completely avoidable due to tunings/workarounds
Summary for SLES11 SP1

- SLES11-SP1 performance is good
  - With some trade-offs roughly equal to SLES10-SP3
    - A common trade-off is increased cpu consumption for better scalability
  - Almost generally recommendable
    - Especially Systems with SLES11 or with heavy FICON I/O
    - An exception are very cpu consumption sensitive environments
      - Here upgrades have to be considered carefully

- Improvements and degradations

<table>
<thead>
<tr>
<th>Base</th>
<th>New</th>
<th>Improved</th>
<th>No difference or Trade-off</th>
<th>Degraded</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLES10</td>
<td>SLES11</td>
<td>5</td>
<td>31</td>
<td>46</td>
</tr>
<tr>
<td>SLES10</td>
<td>SLES11-SP1</td>
<td>33</td>
<td>37</td>
<td>12</td>
</tr>
<tr>
<td>SLES10</td>
<td>SLES11-SP1 tune + w</td>
<td>36</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>SLES11</td>
<td>SLES11-SP1</td>
<td>53</td>
<td>29</td>
<td>0</td>
</tr>
</tbody>
</table>
Questions

• Further information is at
  • Linux on System z – Tuning hints and tips
  • Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/
  • Red Hat Enterprise Linux 6 Tech Notes
IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.