Linux on IBM z13: Performance Aspects of New Technology and Features

Mario Held (mario.held@de.ibm.com)
Linux on z Systems Performance Analyst
IBM Corporation

Session 17772
August 13, 2015
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.

All z13 numbers have been measured on pre GA hardware with pre GA software. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. All z13 numbers have been measured on pre GA hardware. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document or 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.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html ("AUT"). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

August 13, 2015
Agenda

IBM z13 characteristics
- From a performance point of view
- Improving the overall efficiency
- Compiler and libraries including SIMD
- What to expect when running on IBM z13
- Recommendations and outlook

SMT-2

Compiler

Experiences

What's next
IBM z Systems – processor roadmap

- **z10**: 2/2008
  - Leadership Single Thread, Enhanced Throughput
  - Out of Order Execution
  - Accelerator Integration
  - Water Cooling
  - PCIe I/O Fabric
  - RAIM
  - Enhanced Energy Management
  - Workload Consolidation and Integration Engine for CPU Intensive Workloads
    - Decimal FP
    - Infiniband
    - 64-CP Image
    - Large Pages
    - Shared Memory

- **z196**: 8/2012
  - Leadership System Capacity and Performance
  - Modularity & Scalability
  - Dynamic SMT
  - Supports two instruction threads
  - SIMD
  - PCIe attached accelerators
  - Business Analytics Optimized

- **zEC12**: 9/2012
  - Leadership Single Thread, Enhanced Throughput
  - Improved out-of-order Transactional Memory
  - Dynamic Optimization
  - 2 GB page support
  - Step Function in System Capacity

- **z13**: 1/2015
  - Leadership System Capacity and Performance
  - Modularity & Scalability
  - Dynamic SMT
  - Supports two instruction threads
  - SIMD
  - PCIe attached accelerators
  - Business Analytics Optimized

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

August 13, 2015
IBM z13 8-core processor unit

- Up to eight active cores (PUs) per chip
  - 5.0 GHz (v5.5 GHz zEC12)
  - L1 cache/ core
    - 96 KB I-cache
    - 128 KB D-cache
  - L2 cache/ core
    - 2M+2M Byte eDRAM split private L2 cache
- Improved instruction execution bandwidth:
  - Improved branch prediction and instruction fetch to support SMT
  - Instruction decode, dispatch, complete increased up to 6 instructions per cycle
  - Issue up to 10 instructions per cycle
  - Integer and floating point execution units
- On chip 64 MB eDRAM L3 Cache
  - Shared by all cores
- I/O buses
  - One InfiniBand I/O bus
  - Two PCIe I/O buses
- Memory Controller (MCU)
  - Interface to controller on memory DIMMs

- 14S0 22nm SOI Technology
  - 17 layers of metal
  - 3.99 Billion Transistors
  - 13.7 miles of copper wire
IBM z13 8-core processor unit (cont.)

- 2 x instruction pipe width
  - Improves IPC for all modes
  - Symmetry simplifies dispatch/issue rules
  - Required for effective SMT
- Added FXU and BFU execution units
  - 4 FXUs (fixed point)
  - 2 BFUs, DFUs (binary and hexadecimal)
  - 2 new SIMD units
- SIMD unit plus vector registers
- Pipe depth re-optimized for power/performance
  - Product frequency reduced
  - Processor performance increased
- SMT support
  - Wide, symmetric pipeline
  - Full architected state per thread
  - SMT-adjusted CPU usage metering
IBM z13 – advanced system design

* No server can fully exploit its maximum I/O bandwidth

PCI – Processor Capacity Index (IBM MIPS)

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

August 13, 2015
Large Memory – potential performance gains

- 2.5 TB per drawer for a total of 10 TB available
- Enables more caching for classical databases
  - e.g. larger DB2 bufferpools, Oracle SGAs
- Helps with storage pressure under z/VM
- Enables In-Memory Databases
  - Dramatic reduction in response time by avoiding I/O wait
  - DB2 BLU / Oracle 12c
- Enables in memory analytics
- Java heaps can be increased
  - For older Java versions be sure to use –Xcompressedrefs
Agenda

IBM z13 characteristics
From a performance point of view

SMT-2
Improving the overall efficiency

Compiler
Compiler and libraries including SIMD

Experiences
What to expect when running on IBM z13

What's next
Recommendations and outlook
Simultaneous multithreading value example

Two tasks, one core, OS does dispatching, wait time unused

Two tasks, two threads continuously running

Additional capacity

(assumes SMT2 efficiency of 1.4)
Terminology in a SMT environment
SMT – hypervisor picture

- PR/SM supports SMT for SMT aware hypervisor like z/VM via core dispatching
- Hypervisor controls and manages whole core (all threads)
- SMT transparent to virtual operating systems and applications

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Simultaneous multithreading – the technology

- **Simultaneous Multithreading (SMT) technology**
  - Multiple programs (software threads) run on the same processor core
  - More efficient use of the core hardware

- **Active threads share core resources**
  - In space: data and instruction caches, TLBs, branch history tables, etc.
  - In time: pipeline slots, execution units, address translator, etc.

- **Typically increases overall throughput per core when SMT is active**
  - Amount that increase, varies widely with workload
  - Each thread runs more slowly than on a single-thread core

---

### Use of Pipeline Stages in SMT

- **Thread-A**
- **Thread-B**
- **Stage idle**

---

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
ETR / ITR – be careful with calculations

- ETR (External Transaction Rate) / ITR (Internal Transaction Rate)

  \[ ETR = \frac{\text{#Transaction}}{\text{Elapsed time}} = ITR \times \text{processor utilization} \]

- Example with SMT-2 enabled
  - 50% of the logical CPUs utilized
  - Scheduler puts them on different cores
  - Throughput is roughly equivalent to what you get with SMT-1, so 5/6 of total capacity
  - Normal calculation \( ITR = 2 \times ETR \)
    - Assumes SMT gain factor of 100%
  → Wrong result
SMT implies that each logical CPU is slower

- Evaluate your workload
  - Single thread speed dependency?
  - A logwriter process involved?
  - Heavy I/O processing?
SMT gain dependent on #cores used

- Cores on a chip share resources
  - “Imbalanced” demand can deplete a resource when using more cores → can be a scaling bottleneck
  - Can happen earlier with SMT-2

- This is just an example. There is a lot of variability with regard to workload benefit
SMT requires more logical CPUs

- Reduced capacity
  - More CPUs required

- More CPUs
  - SMP n-way effect

- Workload scalability is really important

- Revisit your #virtual CPU sizing

- Measure before you deploy

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
SMT real world example – the good

- SAP Workload
  - 2 CPs, 2 zIIPs, DB server
  - 16 IFL App Server
  - SMT2 with z/VM
  - Overall +41% ITR
SMT real world example – the good (cont.)

- Java Benchmark
  - 12 Cores on two chips in one node
  - Staying inside one Node
  - SMT-2 with Linux

![Java Benchmark Chart]

*Figure 1: Java Benchmark Results*
SMT real world example – the ugly

- The truth is, you have to evaluate YOUR workload
Recommendations for enabling SMT

**z/VM**
- Create a new LPAR with a z/VM that has SMT enabled
- Move one workload (type) / guest at a time
  - Remember to increase the # of virtual CPUs
  - Check the memory!
  - Measure throughput, CPU utilization and response time before and after the movement, keep your monitor record!
- Workloads not showing enough benefit should be run on the z/VM with SMT disabled

**LPAR**
- Test on separate LPAR with SMT2 turned on
  - You can do this directly on your test LPAR
  - Virtual CPUs will automatically double
- Check memory
- Measure throughput, CPU utilization and response time before and after the movement
- Depending on the outcome turn on SMT in the production LPAR
Agenda

IBM z13 characteristics
SMT-2
Compiler
Experiences
What's next

From a performance point of view
Improving the overall efficiency
Compiler and libraries including SIMD
What to expect when running on IBM z13
Recommendations and outlook
SIMD – Single Instruction Multiple Data

Scalar operations
- 64-bit operations one by one

Vector SIMD operations
- many operations up to 128 bit such as sixteen 8-bit operations in the same instant
z13 SIMD – overlaid vector / floating point register file

- Overlaid register file
  - Bits 0:63 of SIMD registers 0-15 will correspond to FPRs 0-15
  - When writing to an FPR, bits 64:127 of the corresponding vector register will become unpredictable

- SIMD width 128 bits
  - 1x128b, 2x64b, 4x32b, 8x16b, 16x8b integer
  - 2x64b, 1x64b floating-point
z13 SIMD – Business analytics vector processing

- Single Instruction Multiple Data instruction set
  - Support
    - Vector load / store, pack / unpack, merge, permute, select
    - Vector gather / scatter element
    - Vector load / store with length; load to block boundary
  - Integer
    - 8b...128b add / subtract (with / without carry / borrow)
    - 8b...64b min, max, average, complement / neg / pos
    - 8b...64b vector compare; single element compare
    - 8b...32b multiply, multiply / add [low / high / even / odd]
    - Full-vector bitops & shifts, 8b..64b element shifts / rotates
    - Sum-across, population count, checksum
    - Galois field multiply sum / and accumulate
z13 SIMD – Business analytics vector processing

- Single Instruction Multiple Data instruction set
  - Floating-point
    - DP add, sub, mul, div, sqrt, multiply-and-add/sub
    - Conversions (integer vs. DP, SP vs. DP)
    - Compare & test data class
    - Scalar forms of all instructions (single-element DP)
    - Full IEEE support (rounding modes, exceptions)
  - String
    - Supported character types: 8b, 16b, 32b
    - Vector Find Any Element [Not] Equal [Or Zero]
    - Vector Find Element [Not] Equal [Or Zero]
    - Vector Isolate String
    - Vector String Range Compare
z13 SIMD – Software exploitation

- Kernel & Hypervisor support
  - Enable vector facility for user space applications
  - Handle vector registers across context switch etc.

- Compiler & Toolchain support
  - Enable application exploitation of vector facility
    - Hand-written assembler code
    - C / C++ language extension (vector types, operators, intrinsics)
      - Automatic vectorization by the compiler
    - Optimized libraries (string, math routines)
  - Open-source toolchain
    - GNU tools: binutils, gcc, gdb, glibc
    - LLVM & clang
    - IBM proprietary tools: IBM Java 8

- Application & middleware exploitation
  - Analytics use cases, e.g. Cognos BI
z13 SIMD – Linux kernel support

- Vector registers
  - Save/restore VRs on context switch
  - Save/restore VRs across signal handler invocation
  - Debugger access (ptrace / core file) to VR register set
  - Kernel indicates support via “vx” feature bit
    - Reported via cat /proc/cpuinfo “features” string
    - Also indicates hardware support
    - **Note: Only checking machine type not sufficient!**

```
[root@s42lp06 ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 32
bogomips per cpu: 20325.00
features        : esan3 zarch stfle msa ldisp eimm dfp edat etf3eh highgprs te vx
cache0          : level=1 type=Data scope=Private size=128K line_size=256 associativity=8
```
IBM z13: GCC support

- `-march=z13` enables z13 instruction set and builtins.

- `-mvx | -mno-vx` enables / disables vector support with ABI (default enabled with `-march=z13`)

- z13 specific instruction scheduling not finished yet

- Scalar vector instruction support

- Auto-vectorization support

- String operations: so far only strlen

- 128 bit integer support

- Low-level builtins: `__builtin_s390_<instruction mnemonic>`

- High-level builtins: Altivec-style mostly compatible to XLC and LLVM
z13 SIMD – C/C++ vector language extension

- Enabled in GCC / clang via `-mzvector` command line option

- Vector types (closely modeled after AltiVec/VSX)
  - Integer: vector [un]signed (char|short|int|long long)
  - Boolean: vector bool (char|short|int|long long)
  - Floating-point: vector double
  - Context-sensitive “vector” keyword

- Vector operators
  - Arithmetic / logical operators extended to vector types
  - Relational / comparison operators return vector of boolean results

- Vector intrinsics
  - Provided by header file `<vecintrin.h>`
  - Adapted to cover all z Systems vector instructions

```
vector signed int absdiff (vector signed int x, vector signed int y) {
    vector bool int cond = x > y;
    vector signed int vt = x - y;
    vector signed int vf = y - x;
    return vec_sel(vt, vf, cond);
}
```
z13 SIMD – Implementation status

- Open-source upstream status
  - Linux kernel: upstream since 3.19 (some fixes in 4.0)
  - Binutils support: upstream (will be in 2.26)
  - GCC support: upstream (will be in 6.1, backported to 5.2)
  - glibc support: optimized memory/string routines posted
  - GDB support: upstream (will be in 7.10)
  - LLVM/clang support: upstream

- Future work:
  - Further performance tuning
  - Improved debug tools (e.g. valgrind), performance tuning (e.g. scheduling, auto-vectorization)
IBM z13: GNU C library – Glibc

- Runtime optimizations with hand-optimized assembler code for many functions available already:
  - `{str | wcs}{len | nlen | cpy | ncpy | cat | ncat | cmp | ncmp | chr | chrnul | rchr | spn | pbrk | cspn}, stpcpy/wcpcpy
  - memchr, rawmemchr, wmemchr, memccpy, wmemset, wmemcmp, memrchr

- Currently GCC / Glibc needs to be forced to actually use them: `-fno-builtins -D_NO_STRING_INLINES`

- Suitability checked via GNU_IFUNC feature

- Overhead for small strings due to library call
  - GCC in the future will handle up to 16 bytes before calling the library

Upstream planned for Glibc 2.23
Many string operations exploit SIMD already
SIMD would help a lot also in other areas
Agenda

IBM z13 characteristics  From a performance point of view
SMT-2  Improving the Overall Efficiency
Compiler  Compiler and libraries including SIMD
Experiences  What to expect when running on IBM z13
What's next  Recommendations and outlook

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

August 13, 2015
Single Core improvement for existing C and C++ workloads (integer)

gcc-5.2-150707-rev225504 with highest possible machine optimization (-march=zEC12 on zEC12 versus -march=z13 on z13)

Overall throughput improvement is 15%.
Single Core improvement for existing C, C++ and Fortran workloads (floating point)

gcc-5.2-150707-rev225504 with machine optimization (-march=zEC12 on zEC12 versus -march=zEC12 and -march=z13 on z13)

Overall throughput improvement is 26% respective 36% (SIMD enabled)
Compare zEC12 to z13 Java workload
6 cores – all on one chip

<table>
<thead>
<tr>
<th></th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZEC12 6 core Java 7.1 SR2</td>
<td>+35%</td>
</tr>
<tr>
<td>Z13 6 core Java 7.2 SR2</td>
<td>+12%</td>
</tr>
<tr>
<td>Z13 6 core Java 7.2 SR2 – SMT-2</td>
<td></td>
</tr>
</tbody>
</table>

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

August 13, 2015
Compare zEC12 to z13 Java workload
12 cores - 2 chips – all on one node

Throughput

+22%
+13%

zEC12 12 core Java 7.1 SR2
z13 12 core Java 7.2 SR2
z13 12 core Java 7.2 SR2 – SMT-2
Agenda

IBM z13 characteristics
SMT-2
Compiler
Experiences
What's next

From a performance point of view
Improving the overall efficiency
Compiler and libraries including SIMD
What to expect when running on IBM z13
Recommendations and outlook
Linux performance fixes coming

- During testing several problems got identified

- Patches are already available or in test

- Expect release later this year into the service stream of RHEL6.x, RHEL7.x, SLES11.x, SLES12.x
  - Alpha and beta versions in test

Note: Future Linux distribution contents depend on distributor support and are always subject to change without notice!
# GCC versions in Linux z Systems distributions

<table>
<thead>
<tr>
<th>GCC stream</th>
<th>First release</th>
<th>Max -march</th>
<th>Included in SUSE distribution</th>
<th>Included in Red Hat distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>02/2006</td>
<td>z9-109</td>
<td>SLES10</td>
<td>RHEL5</td>
</tr>
<tr>
<td>4.2</td>
<td>05/2007</td>
<td>z9-109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>05/2008</td>
<td>z9-ec</td>
<td>SLES11 (z10 backport)</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>04/2009</td>
<td>z10</td>
<td></td>
<td>RHEL5.6**/6.1 (z196 backport)</td>
</tr>
<tr>
<td>4.5</td>
<td>04/2010</td>
<td>z10</td>
<td>SLES11 SP1</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>03/2011</td>
<td>z196</td>
<td>SLES11 SP2*</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>03/2012</td>
<td>z196</td>
<td>SLES11 SP3 (zEC12 backport)**</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>03/2013</td>
<td>zEC12</td>
<td>SLES12</td>
<td>RHEL7.2 (z13 backport) ?</td>
</tr>
<tr>
<td>4.9</td>
<td>04/2014</td>
<td>zEC12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>04/2015</td>
<td>5.2:z13</td>
<td>SLES12 SP1 (5.2 with z13)**</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2016 ?</td>
<td>z13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* included in SDK, optional, not supported  
** fully supported add-on compiler  

Note: Future Linux distribution contents depend on distributor support and are always subject to change without notice! Also schedules of gcc.gnu.org are always subject to change without notice!
Java recommendations

Linux on z Systems requires the following Java releases and above for optimal performance to exploit hardware features in z9, z10, z196, zEC12 and z13:

<table>
<thead>
<tr>
<th>Java Release</th>
<th>SR or FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java6</td>
<td>SR16 FP3</td>
</tr>
<tr>
<td>Java6.1</td>
<td>(VM 2.6) SR8 FP3</td>
</tr>
<tr>
<td>Java7</td>
<td>SR7 FP10</td>
</tr>
<tr>
<td>Java7.1</td>
<td>SR2 FP10</td>
</tr>
<tr>
<td>Java 8</td>
<td></td>
</tr>
</tbody>
</table>

For a list of Java SDK versions shipped and supported by Websphere Application Server fix packs see following link:
Verify Java SDK version shipped with IBM WebSphere Application Server
Ensure that you update all the middleware that comes with an embedded Java version
Questions?

- Further information
  - Linux on z Systems – Tuning hints and tips
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

Mario Held

IBM Deutschland Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

E-mail:
mario.held@de.ibm.com