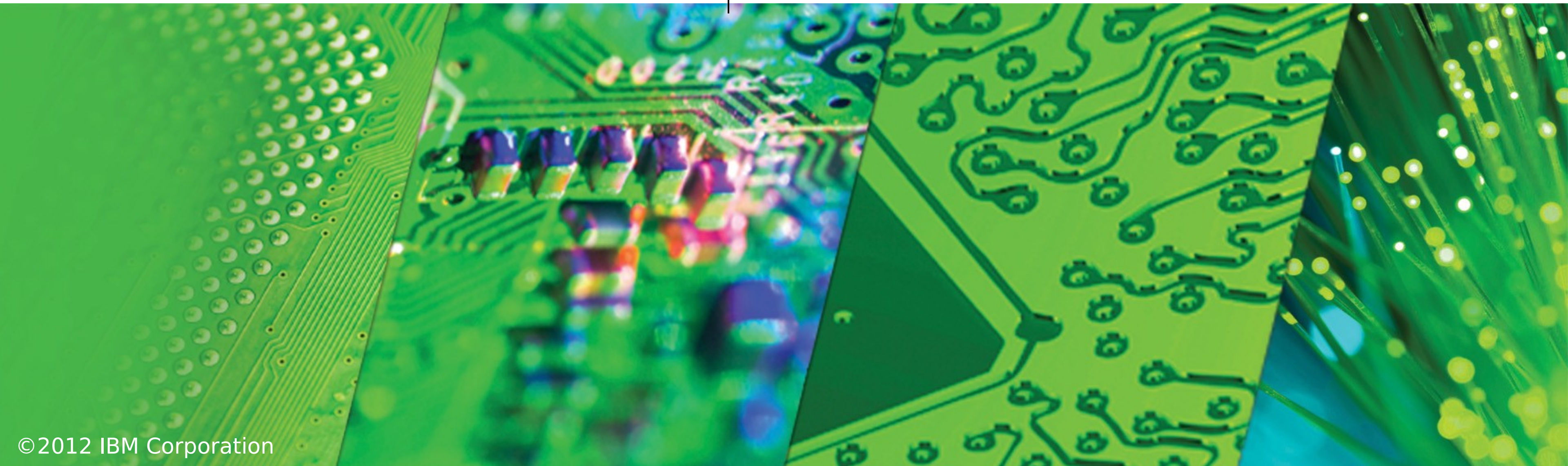
2012
IBM System z Technical University
Enabling the infrastructure for smarter computing

Linux on System z – performance update

zLG12

Mario Held



Trademarks

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.

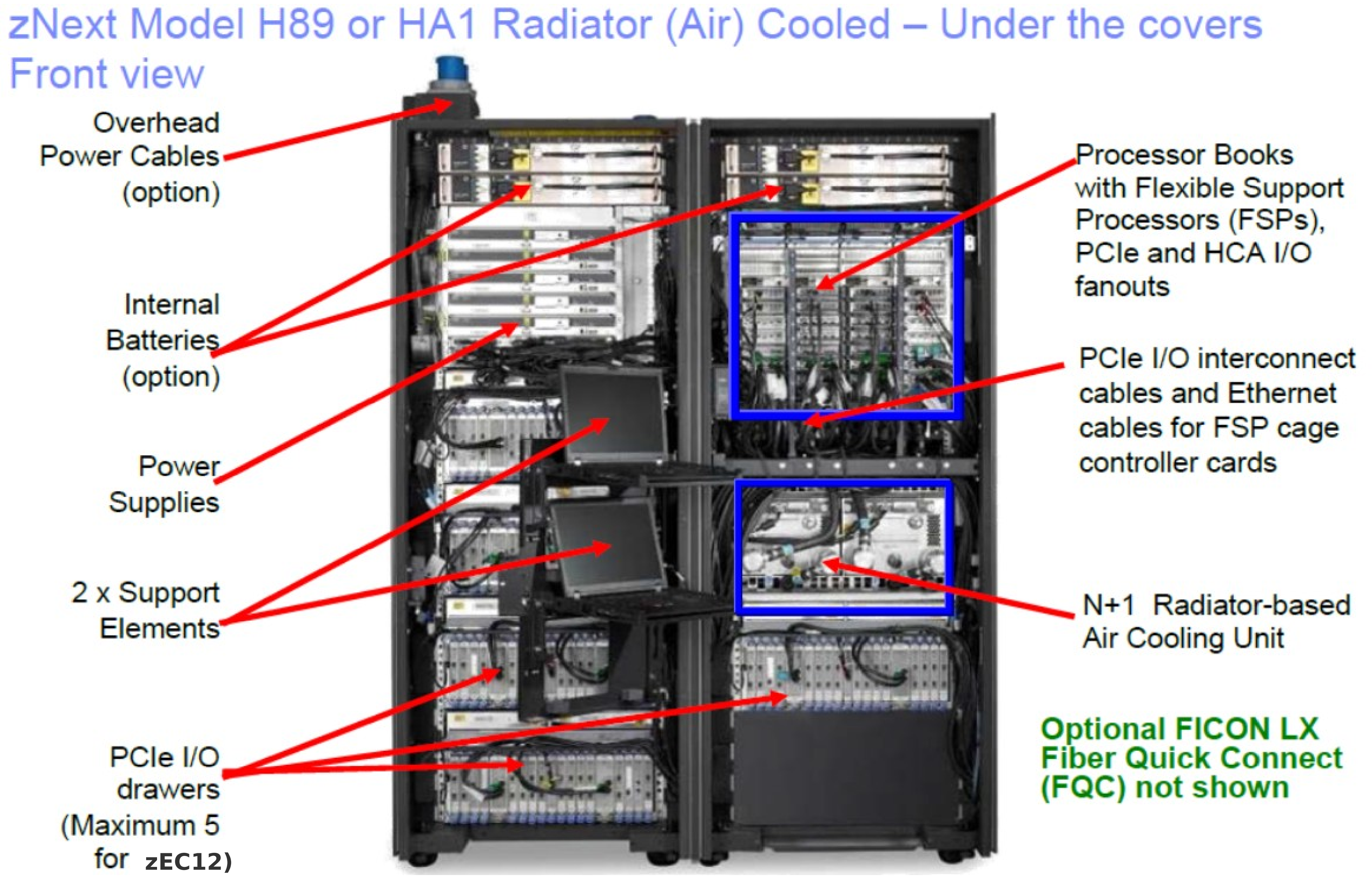
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

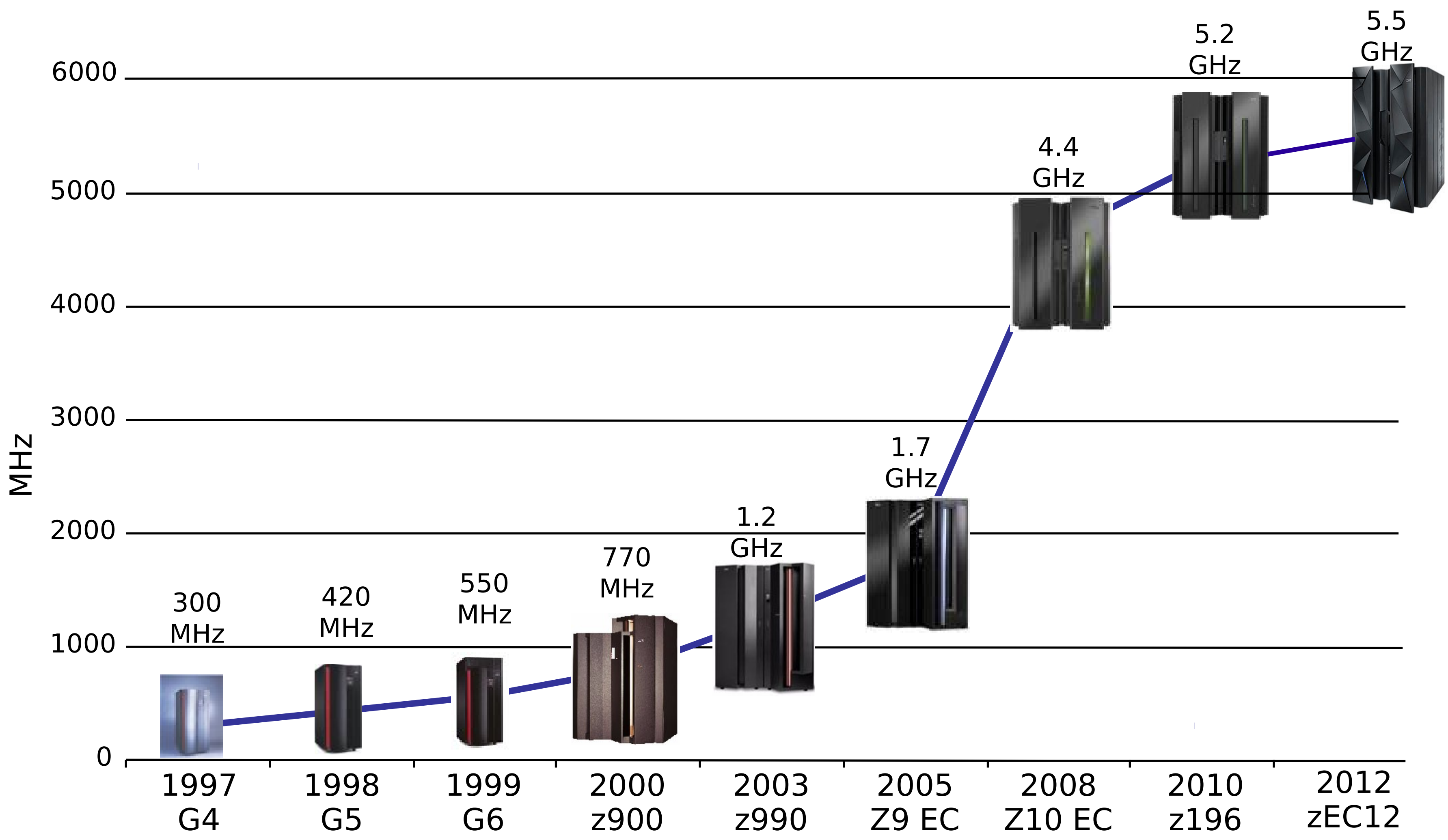
Agenda

- zEnterprise EC12 design
- Linux performance comparison zEC12 and z196
- Performance improvements in other areas
  - ✓ Java JRE 1.7.0

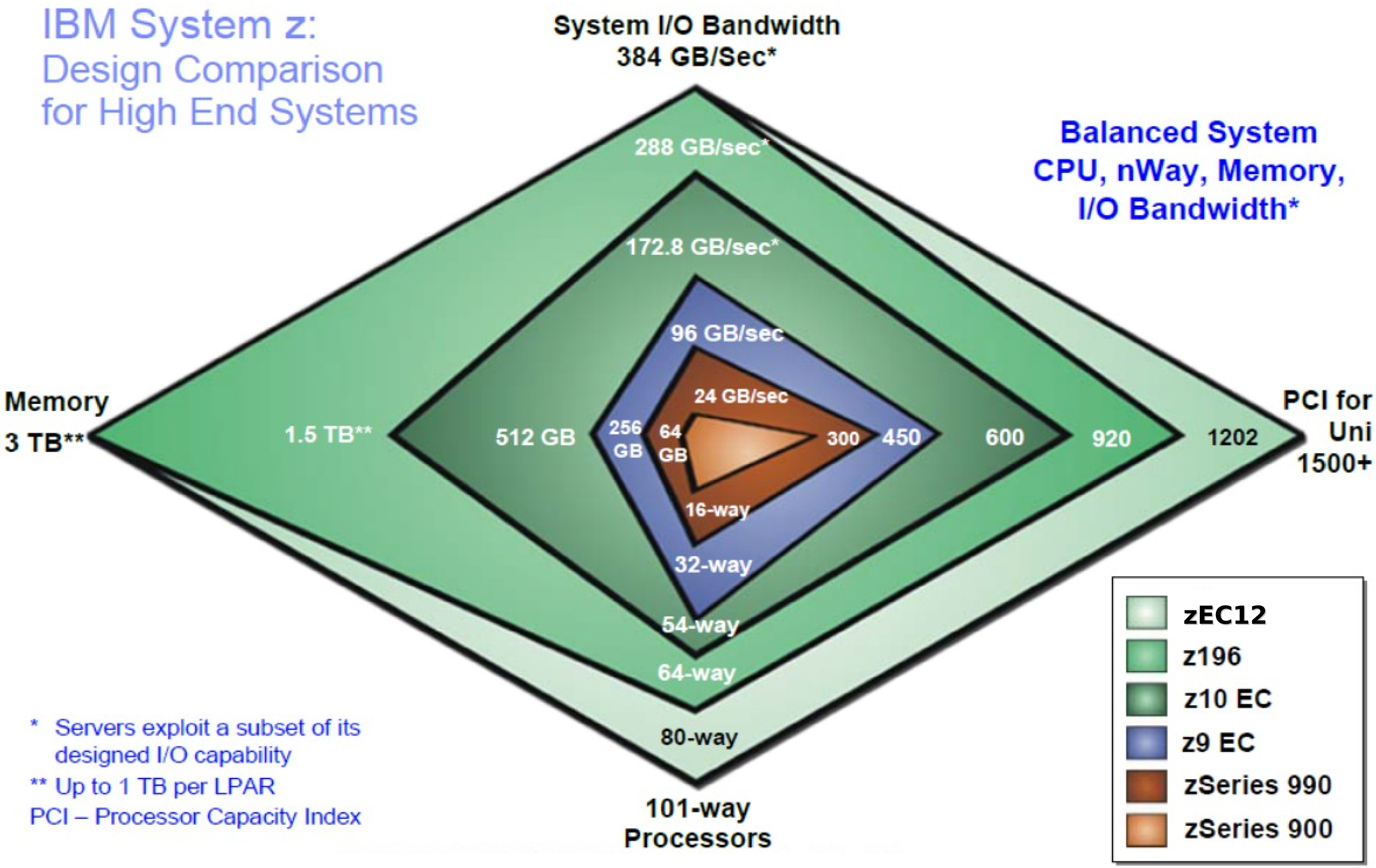
zEC12 – Under the covers



zEC12 Continues the Mainframe Heritage



The evolution of mainframe generations



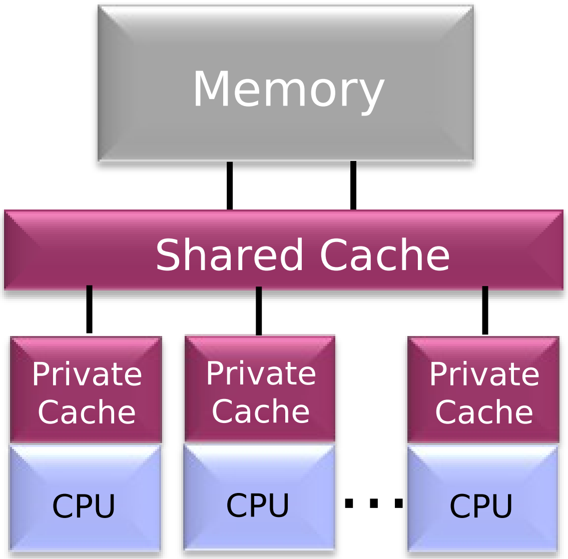
Processor Design Basics

CPU (core)

- ✓ Cycle time
- ✓ Pipeline, execution order
- ✓ Branch prediction
- ✓ Hardware versus millicode

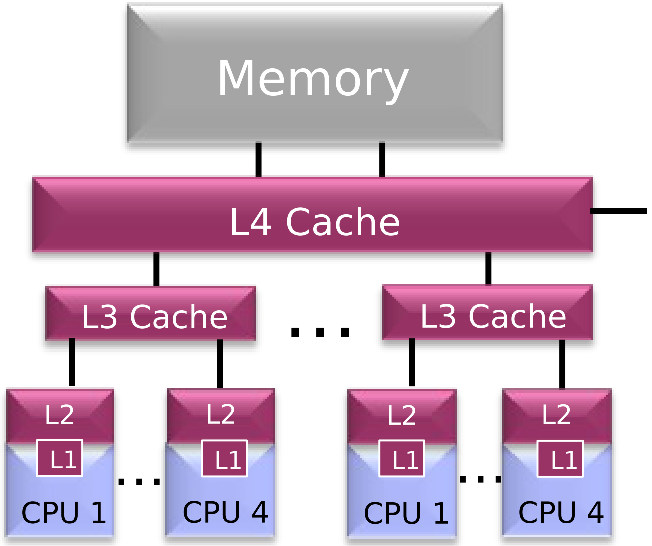
- Memory subsystem
  - ✓ High speed buffers (caches)
    - On chip, on book
    - Private, shared
    - Coherency required
  - ✓ Buses
    - Number
    - Bandwidth
  - ✓ Limits
    - Distance + speed of light
    - Space

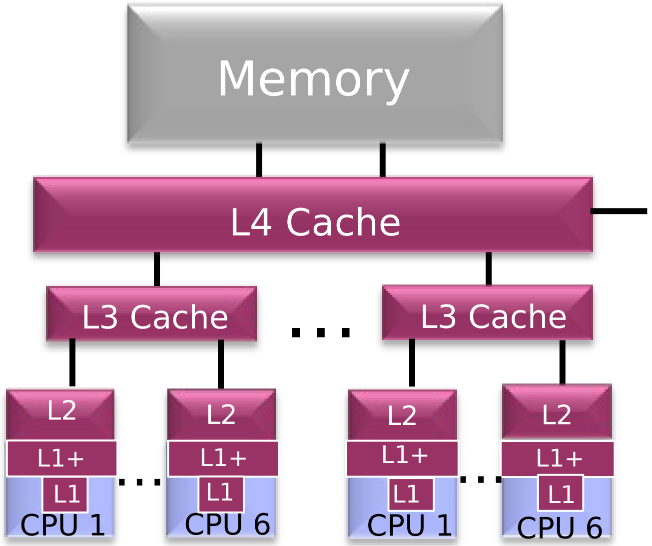
Generic Hierarchy example



zEC12 vs. z196 Hardware Comparison

- z196
  - ✓ CPU
    - 5.2 Ghz
    - Out-of-Order execution
  - ✓ Caches
    - L1 private 64k instr, 128k data
    - L2 private 1.5 MiB
    - L3 shared 24 MiB per chip
    - L4 shared 192 MiB per book
- zEC12
  - ✓ CPU
    - 5.5 GHz
    - Improved Out-of-Order execution
  - ✓ Caches
    - L1 private 64k instr, 96k data
    - L1+ 1 MiB (acts as second level data cache)
    - L2 private 1 MiB (acts as second instruction cache)
    - L3 shared 48 MiB per chip
    - L4 shared 2 x 192 MiB => 384 MiB per book





Agenda

- zEnterprise zEC12 design
- Linux performance comparison zEC12 and z196
- Performance improvements in other areas
  - ✓ Java JRE 1.7.0

zEC12 vs z196 comparison Environment

- Hardware
  - ✓ zEC12 — 2827-708 H66 with pre-GA microcode, pre-GA hardware
  - ✓ z196 — 2817-766 M66
  - ✓ (z10 — 2097-726 E26)
- Linux distribution with recent kernel
  - ✓ SLES11 SP2: 3.0.13-0.27-default
  - ✓ Linux in LPAR
  - ✓ Shared processors
  - ✓ Other LPARs deactivated

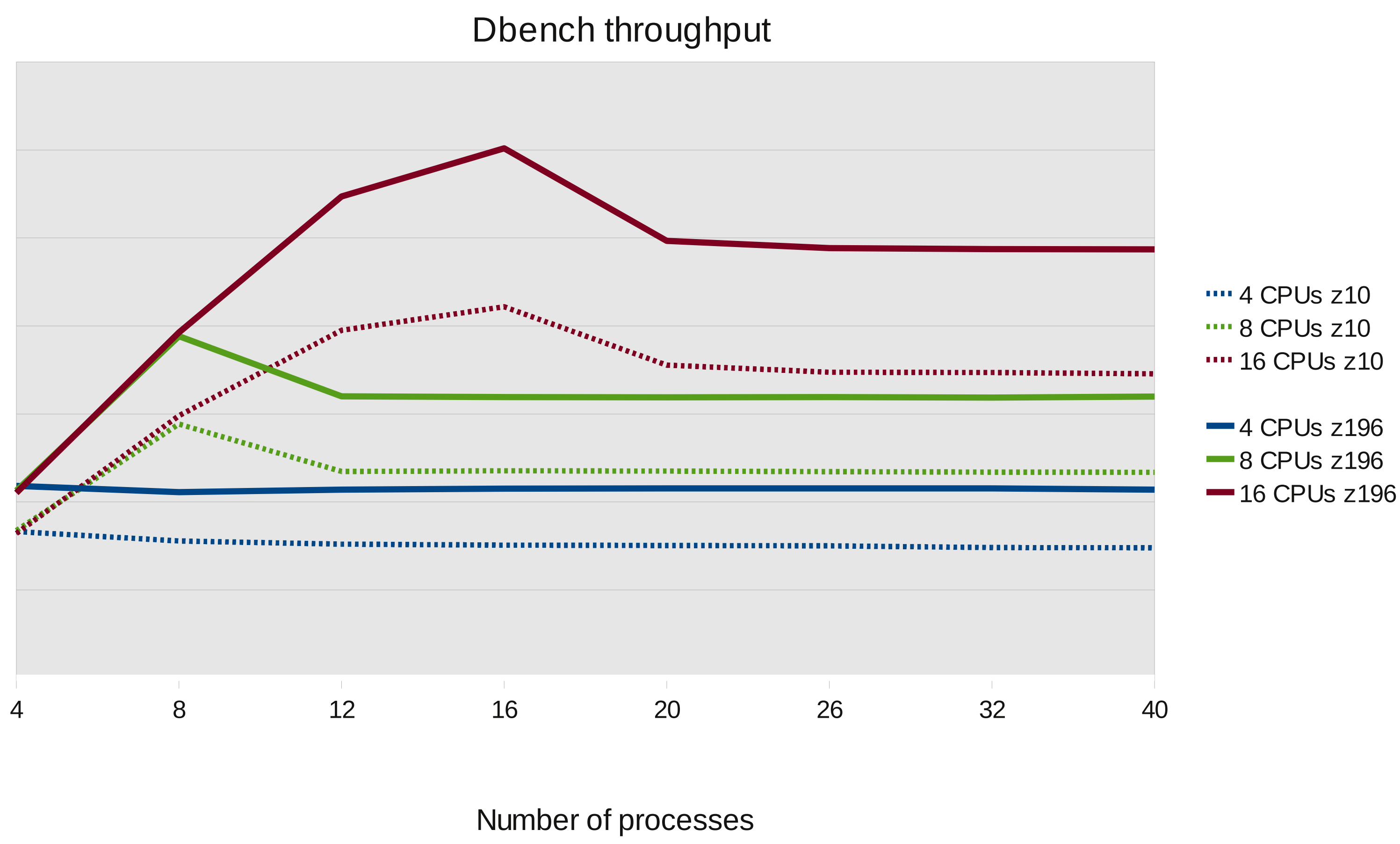
File server benchmark description

- Dbench 3
  - ✓ Emulation of Netbench benchmark
  - ✓ Generates file system load on the Linux VFS
  - ✓ Does the same I/O calls like the smbd server in Samba (without networking calls)
  - ✓ Mixed file operations workload for each process: create, write, read, append, delete
  - ✓ Measures throughput of transferred data
- Configuration
  - ✓ 2 GiB memory, mainly memory operations
  - ✓ Scaling processors 1, 2, 4, 8, 16
  - ✓ For each processor configuration scaling processes 1, 4, 8, 12, 16, 20, 26, 32, 40

Dbench3 (IBM internal driver)

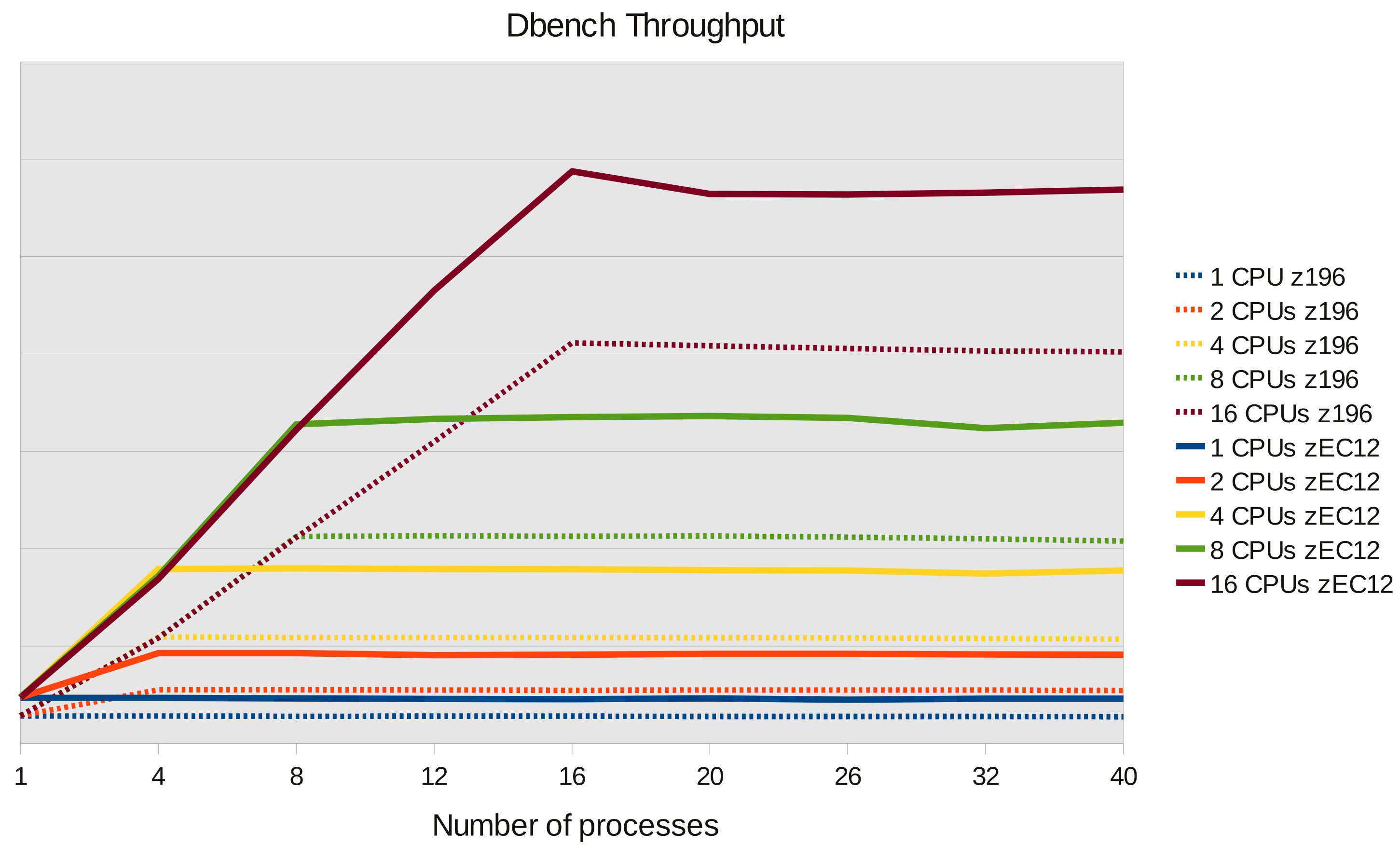
2010

- Throughput improves by 40 percent in this scaling experiment comparing z196 to z10



Dbench3

- Throughput improves by 38 to 68 percent in this scaling experiment comparing zEC12 to z196



Kernel benchmark description

- Lmbench 3
  - ✓ Suite of operating system micro-benchmarks
  - ✓ Focuses on interactions between the operating system and the hardware architecture
  - ✓ Latency measurements for process handling and communication
  - ✓ Latency measurements for basic system calls
  - ✓ Bandwidth measurements for memory and file access, operations and movement
  - ✓ Configuration
    - 2 GB memory
    - 4 processors

Lmbench3

2010

- Most benefits in L3 and L4 cache, overall +40%

Measured operation	Deviation z196 to z10 in %
simple syscall	-30
simple read/write	0
select of file descriptors	35
signal handler	-22
process fork	25
libc bcopy aligned L1 / L2 / L3 / L4 cache / main memory	0 / 20 / 100 / 300 / n/a
libc bcopy unaligned L1 / L2 / L3 / L4 cache / main memory	15 / 0 / 0 / 40 / n/a
memory bzero L1 / L2 / L3 / L4 cache / main memory	35 / 90 / 300 / 800 / n/a
memory partial read L1 / L2 / L3 / L4 cache / main memory	45 / 25 / 130 / 500 / n/a
memory partial read/write L1 / L2 / L3 / L4 cache / main memory	15 / 15 / 10 / 120 / n/a
memory partial write L1 / L2 / L3 / L4 cache / main memory	80 / 30 / 60 / 300 / n/a
memory read L1 / L2 / L3 / L4 cache / main memory	10 / 30 / 40 / 300 / n/a
memory write L1 / L2 / L3 / L4 cache / main memory	50 / 30 / 30 / 180 / n/a
Mmap read L1 / L2 / L3 / L4 cache / main memory	50 / 35 / 85 / 300 / n/a
Mmap read open2close L1 / L2 / L3 / L4 cache / main memory	40 / 35 / 50 / 200 / n/a
Read L1 / L2 / L3 / L4 cache / main memory	20 / 40 / 90 / 300 / n/a
Read open2close L1 / L2 / L3 / L4 cache / main memory	25 / 35 / 90 / 300 / n/a
Unrolled bcopy unaligned L1 / L2 / L3 / L4 cache / main memory	100 / 75 / 75 / 200 / n/a
memory	70 / 0 / 80 / 300 / n/a
mappings	40

Lmbench3

- Benefits seen in the very most operations

Measured operation	Deviation zEC12 to z196 in %
simple syscall	52
simple read/write	46 /43
select of file descriptors	32
signal handler	55
process fork	25
libc bcopy aligned L1 / L2 / L3 / L4 cache / main memory	0 / 12 / 25 / 10 / n/a
libc bcopy unaligned L1 / L2 / L3 / L4 cache / main memory	0 / 26 / 25 / 35 / n/a
memory bzero L1 / L2 / L3 / L4 cache / main memory	40 / 13 / 20 / 45 / n/a
memory partial read L1 / L2 / L3 / L4 cache / main memory	-10 / 25 / 45 / 105 / n/a
memory partial read/write L1 / L2 / L3 / L4 cache / main memory	75 / 75 / 90 / 180 / n/a
memory partial write L1 / L2 / L3 / L4 cache / main memory	45 / 50 / 62 / 165 / n/a
memory read L1 / L2 / L3 / L4 cache / main memory	5 / 10 / 45 / 120 / n/a
memory write L1 / L2 / L3 / L4 cache / main memory	80 / 92 / 120 / 250 / n/a
Mmap read L1 / L2 / L3 / L4 cache / main memory	0 / 13 / 35 / 110 / n/a
Mmap read open2close L1 / L2 / L3 / L4 cache / main memory	23 / 18 / 19 / 55 / n/a
Read L1 / L2 / L3 / L4 cache / main memory	60 / 30 / 35 / 50 / n/a
Read open2close L1 / L2 / L3 / L4 cache / main memory	27 / 30 / 35 / 60 / n/a
Unrolled bcopy unaligned L1 / L2 / L3 / L4 cache / main memory	35 / 28 / 60 / 35 / n/a
Unrolled partial bcopy unaligned L1 / L2 / L3 / L4 cache / main memory	35 / 13 / 45 / 20 / n/a
mappings	34-41

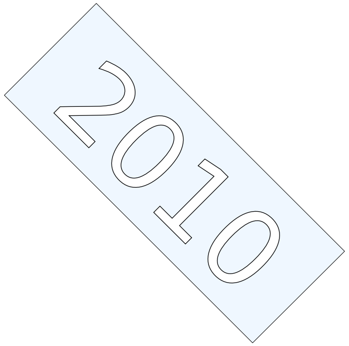
Java benchmark description

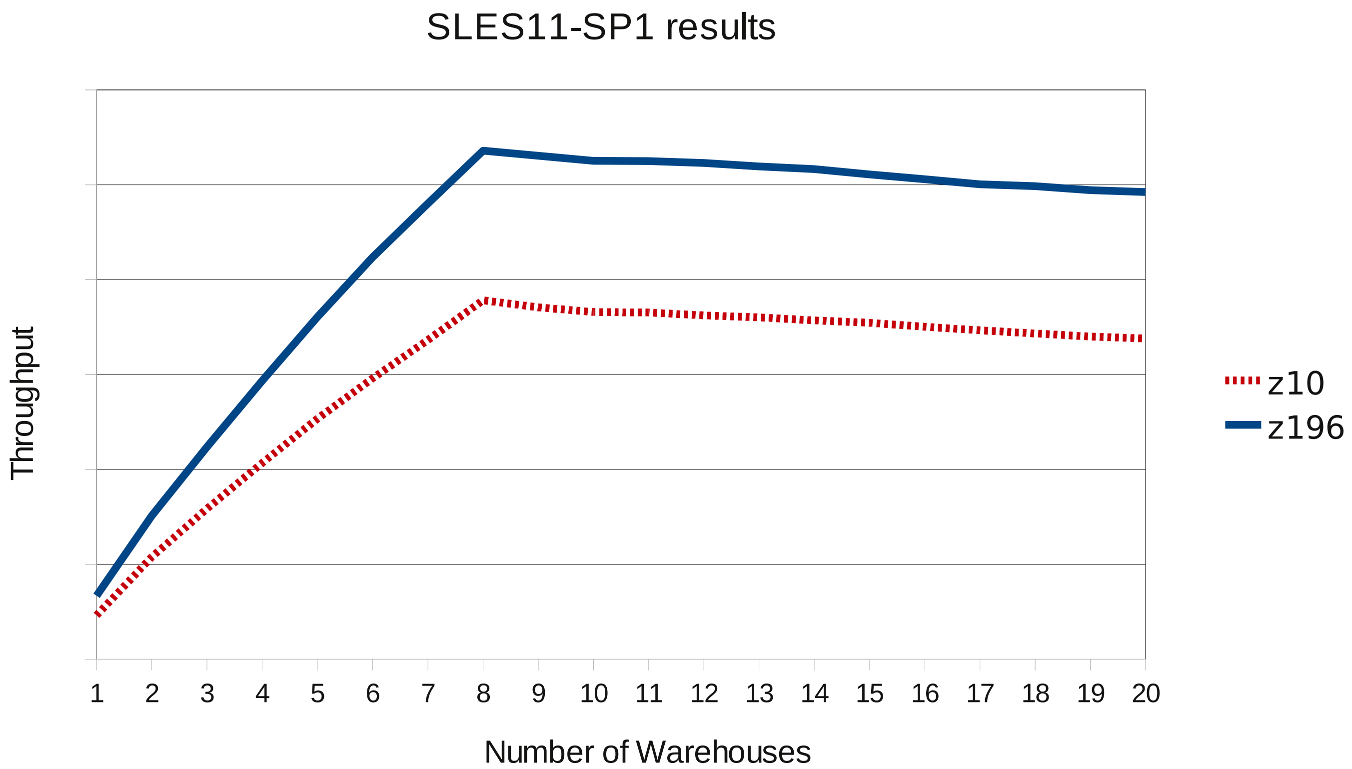
- Java server benchmark
  - ✓ Evaluates the performance of server side Java
  - ✓ Exercises
    - Java Virtual Machine (JVM)
    - Just-In-Time compiler (JIT)
    - Garbage collection
    - Multiple threads
    - Simulates real-world applications including XML processing or floating point operations
  - ✓ Can be used to measure performance of processors, memory hierarchy and scalability

- Configurations
  - ✓ 8 processors, 2 GiB memory, 1 JVM

Java benchmark

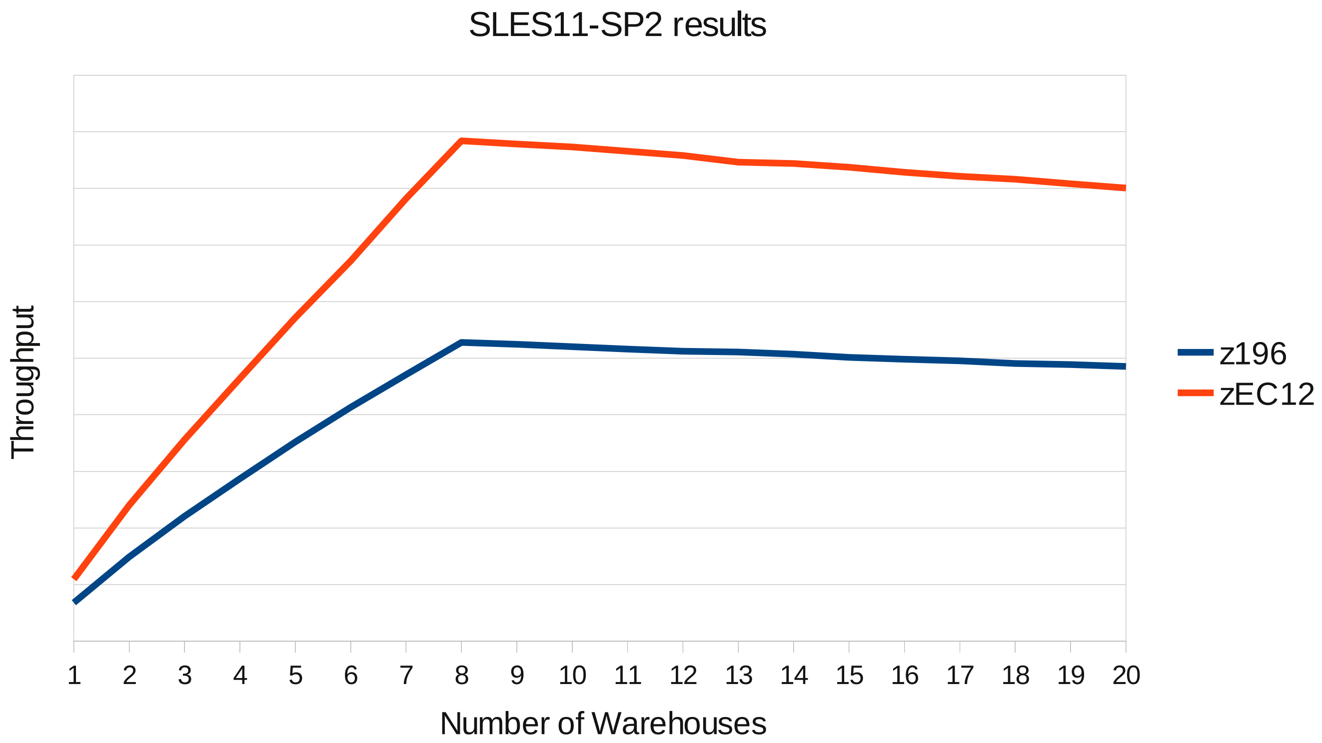
- Business operation throughput improved by approximately 44%
  - ✓ IBM J9 JRE 1.6.0 SR9 64-bit
  - ✓ 8 processors, 2 GiB memory, 1 JVM





Java benchmark

- Business operation throughput improved by approximately 65%
  - ✓ IBM J9 JRE 1.6.0 SR9 64-bit
  - ✓ 8 processors, 2 GiB memory, 1 JVM
- Results seen with a single LPAR active on the machine
- On a fully utilized machine we expect approximately 30%



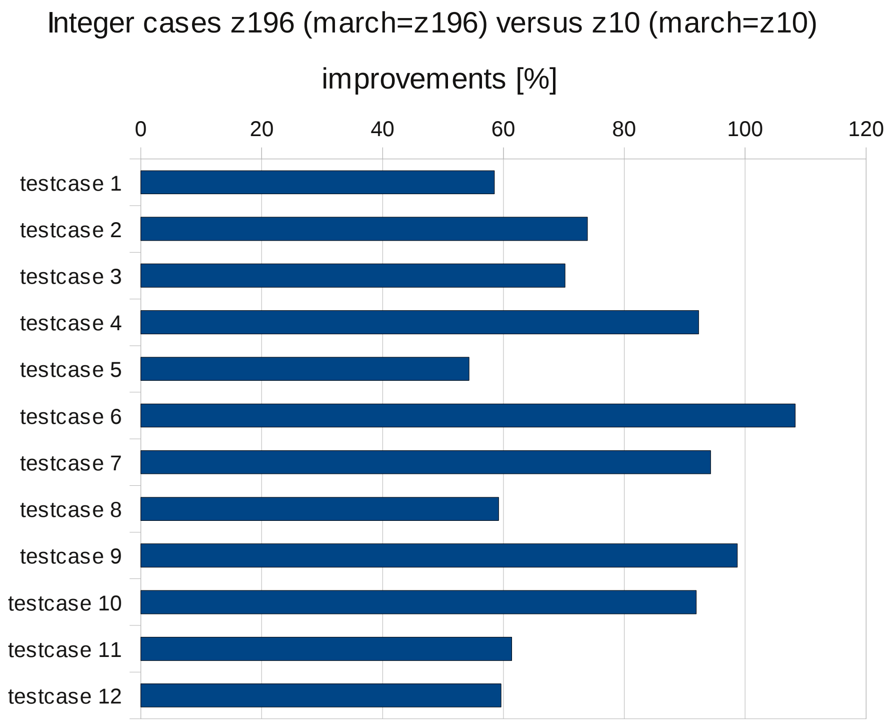
CPU-intense benchmark suite

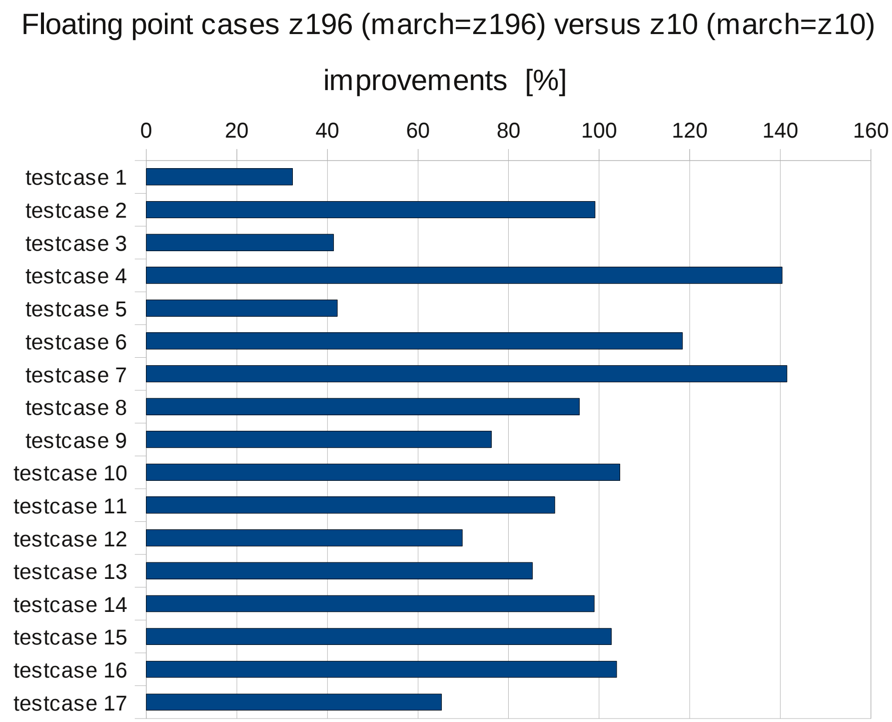
- Stressing a system's processor, memory subsystem and compiler
- Workloads developed from real user applications
- Exercising integer and floating point in C, C++, and Fortran programs
- Can be used to evaluate compile options
- Can be used to optimize the compiler's code generation for a given target system
- Configuration
  - ✓ 1 processor, 2 GiB memory, executing one test case at a time

Single-threaded, compute-intense workload

2010

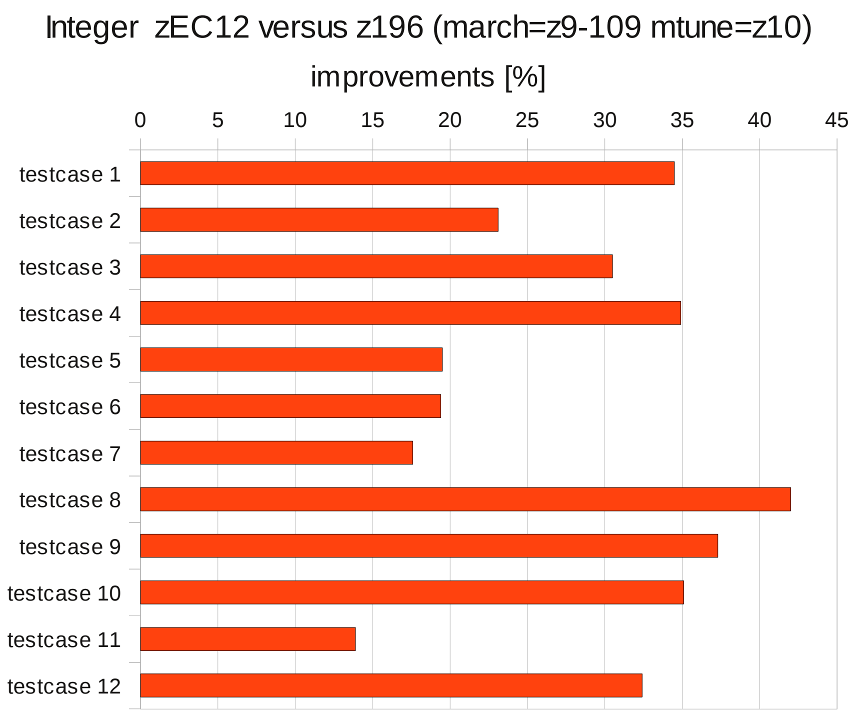
- Linux: Internal driver (kernel 2.6.29) gcc 4.5, glibc 2.9.3
  - ✓ Integer suite improves by 76% (geometric mean)
  - ✓ Floating Point suite improves by 86% (geometric mean)

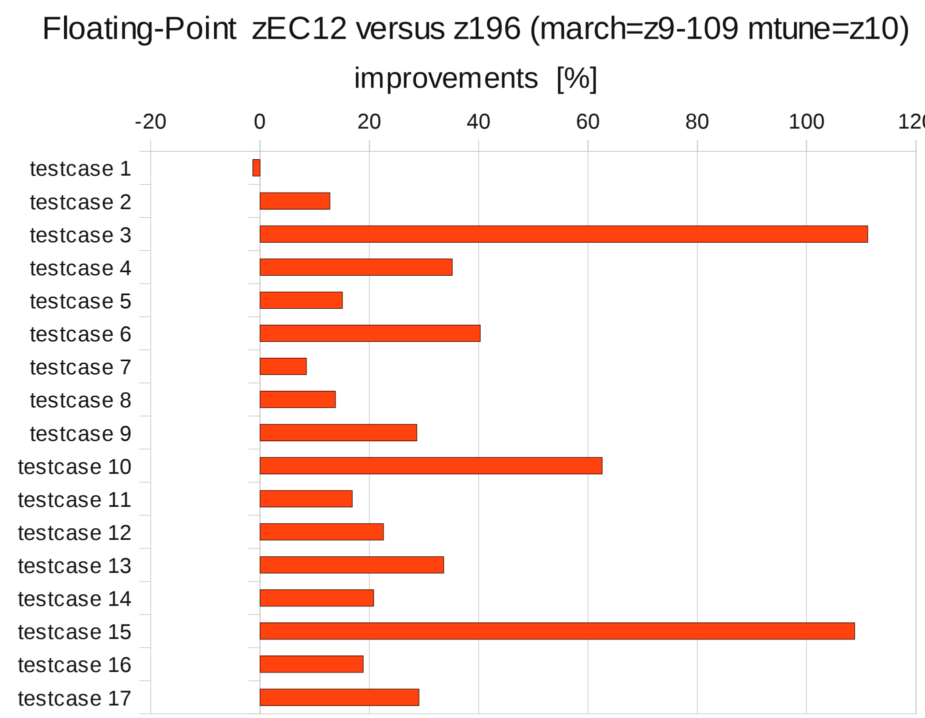




Single-threaded, compute-intense workload

- SLES11 SP2 GA, gcc-4.3-62.198, glibc-2.11.3-17.31.1 using default machine optimization options as in gcc-4.3 s390x
  - ✓ Integer suite improves by 28% (geometric mean)
  - ✓ Floating Point suite improves by 31% (geometric mean)



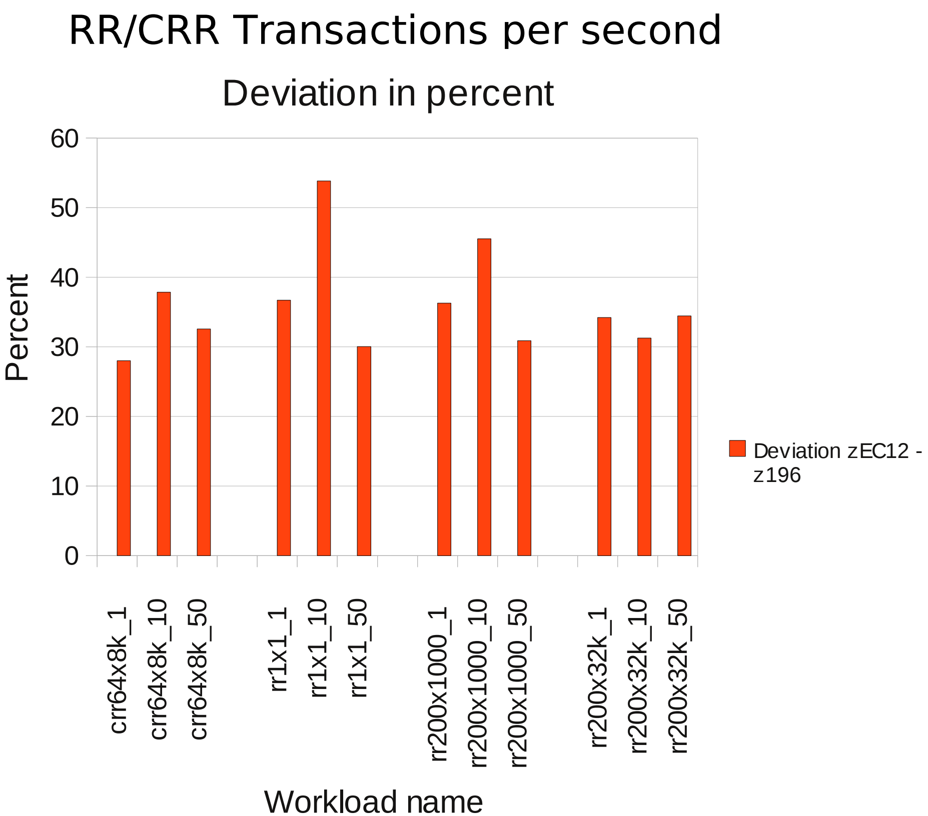


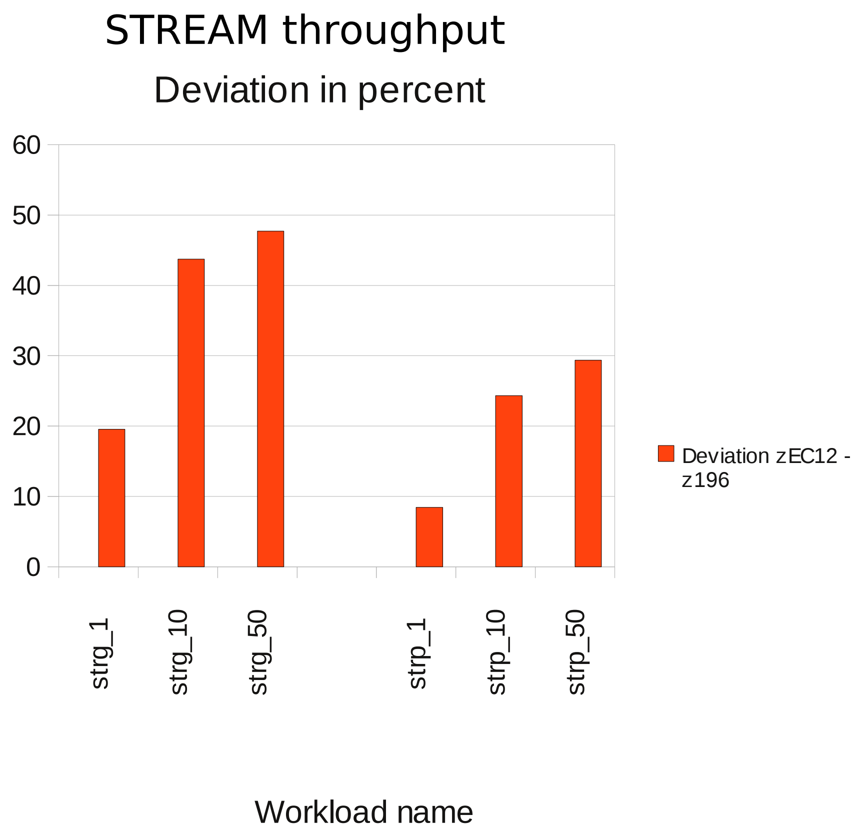
Benchmark description – Network

- 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)

AWM Hipersockets MTU-32k IPv4 LPAR-LPAR

- More transactions / throughput with 1, 10 and 50 connections
- More data transferred at 20 to 30 percent lower processor consumption



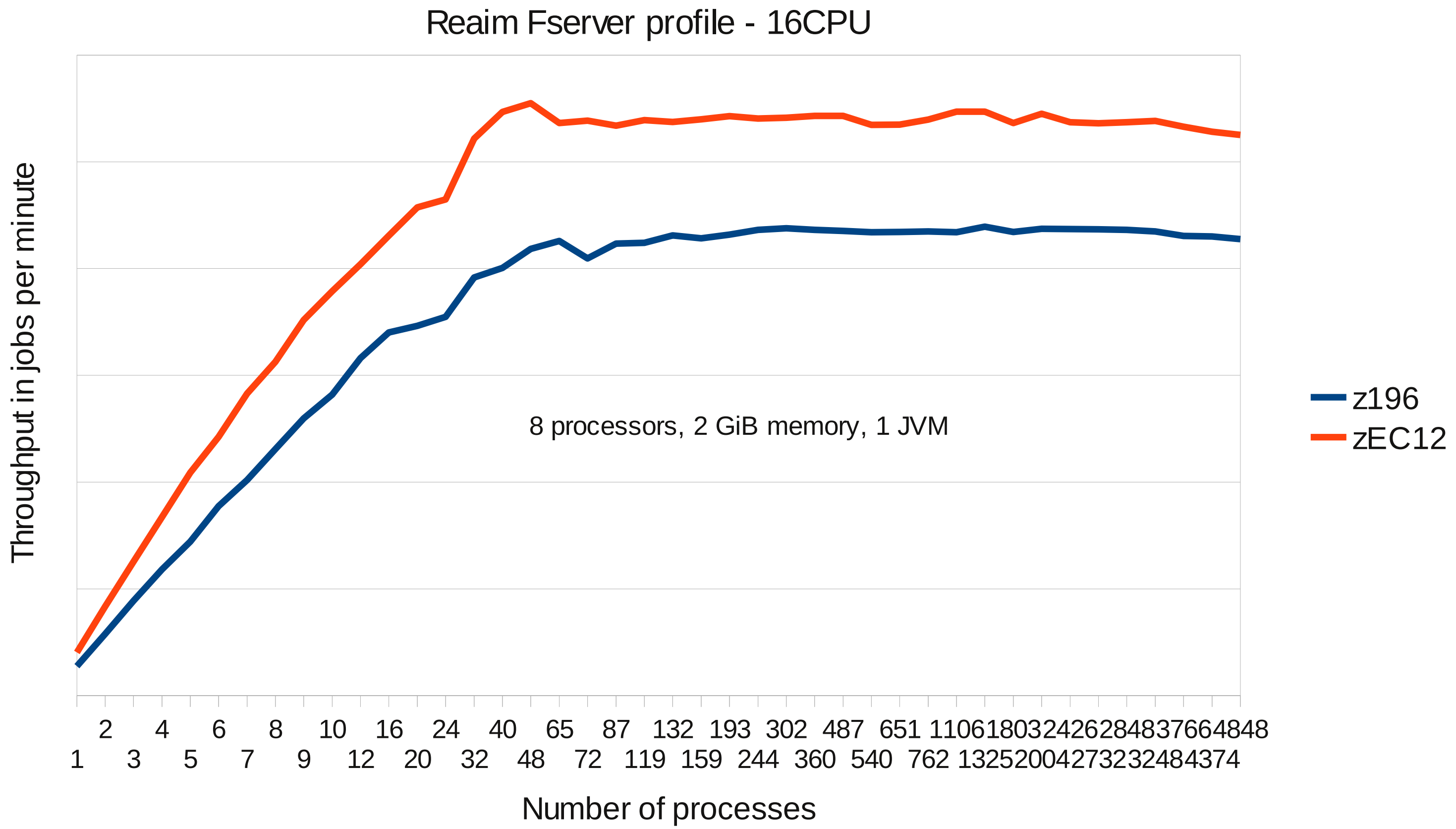


Benchmark description – Re-Aim 7

- Scalability benchmark Re-Aim-7
  - ✓ Open Source equivalent to the AIM Multiuser benchmark
  - ✓ Workload patterns describe system call ratios (patterns can be more ipc, disk or calculation intensive)
  - ✓ The benchmark then scales concurrent jobs until the overall throughput drops
    - Starts with one job, continuously increases that number
    - Overall throughput usually increases until #threads ≈ #CPUs
    - Then threads are further increased until a drop in throughput occurs
    - Scales up to thousands of concurrent threads stressing the same components
  - ✓ Often a good check for non-scaling interfaces
    - Some interfaces don't scale at all (1 Job throughput ≈ multiple jobs throughput, despite >1 CPUs)
    - Some interfaces only scale in certain ranges (throughput suddenly drops earlier as expected)
  - ✓ Measures the amount of jobs per minute a single thread and all the threads can achieve
- Our Setup
  - ✓ 2, 8, 16 CPUs, 4 GiB memory, scaling until overall performance drops
  - ✓ Using a journaled file system on an xpram device (stress FS code, but not be I/O bound)
  - ✓ Using fserver, new-db and compute workload patterns

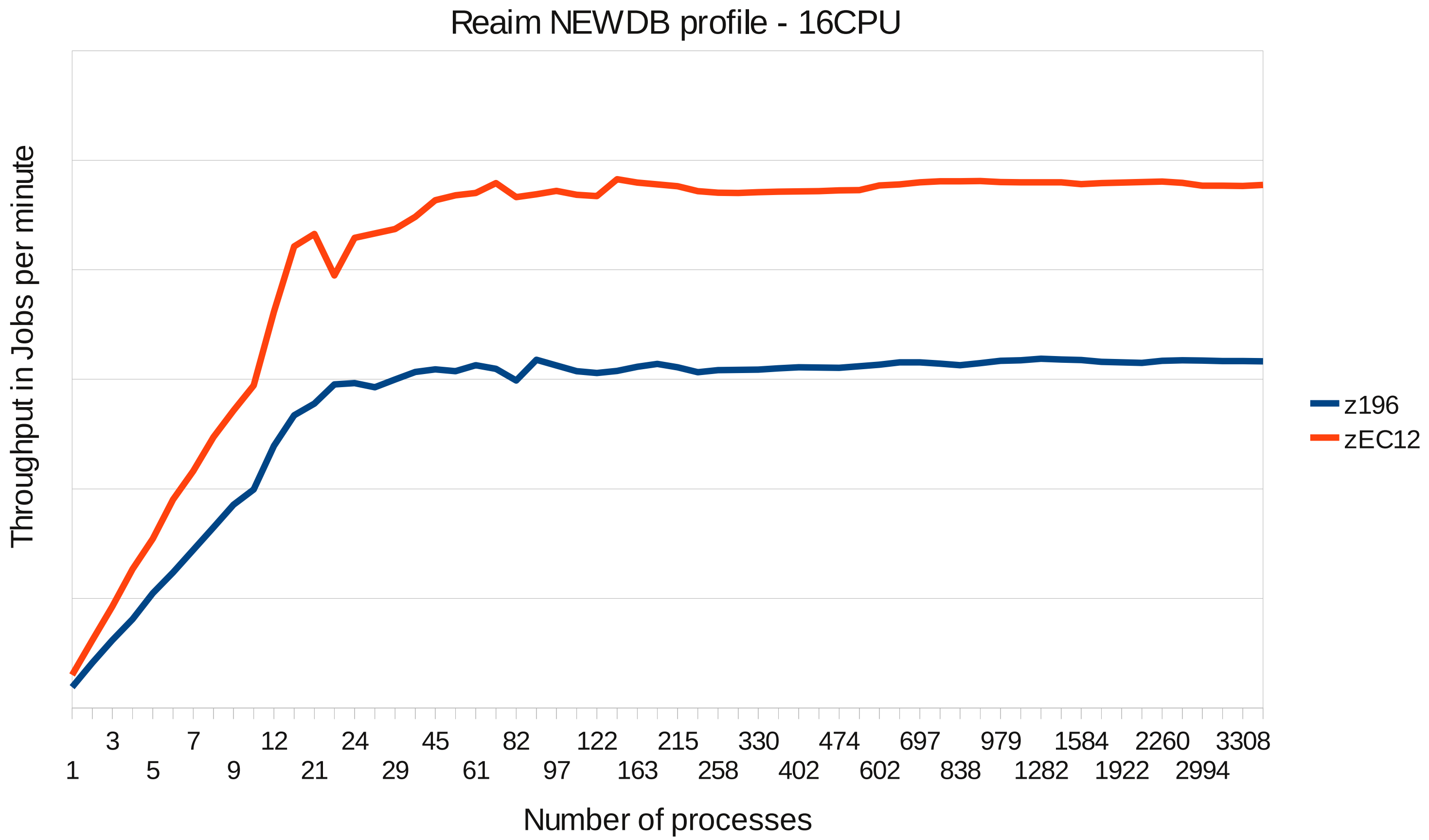
Re-Aim Fserver

- Higher throughput with 4, 8, and 16 PUs (25 to 50 percent) at 30 percent lower processor consumption



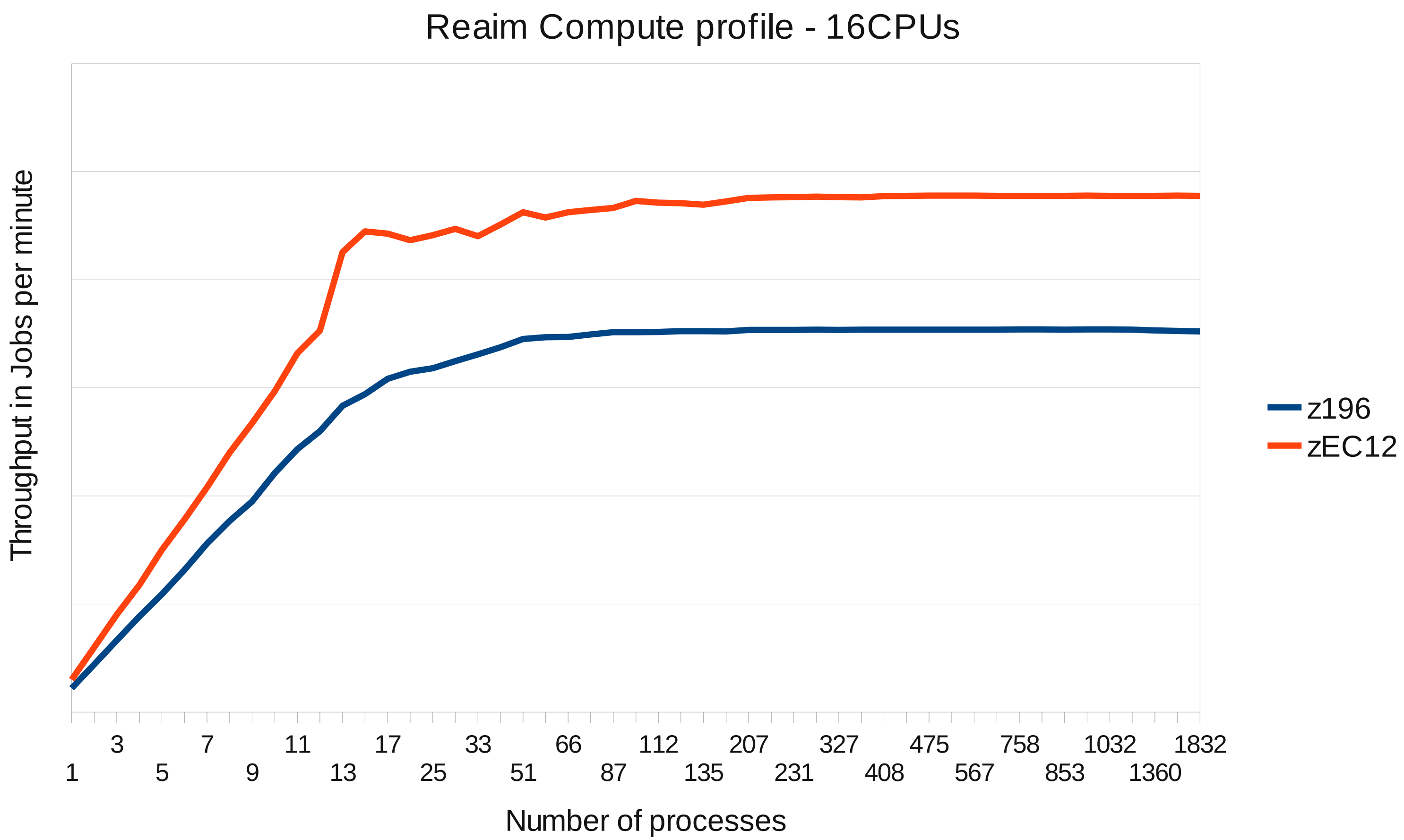
Re-Aim Newdb

- Higher throughput with 4, 8, and 16 CPUs (42 to 66 percent) at 35 percent lower processor consumption



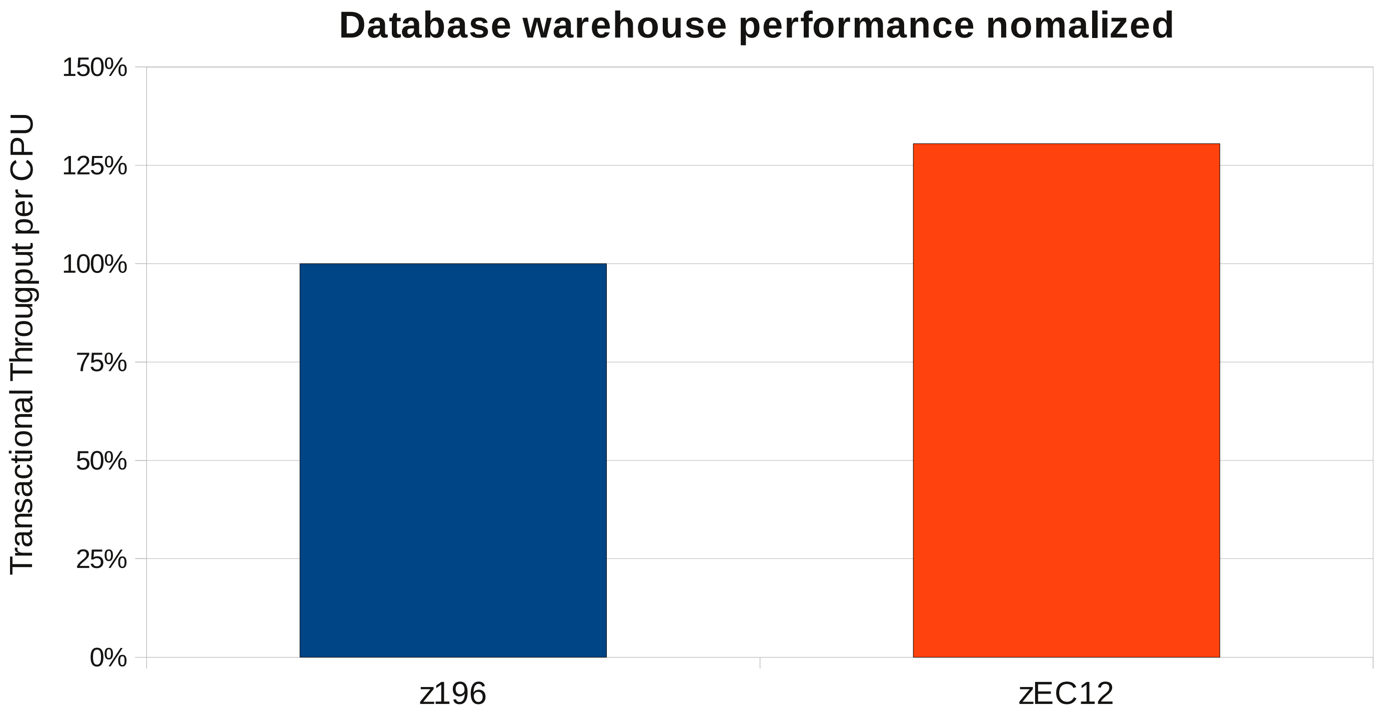
Re-Aim Compute

- Higher throughput with 4, 8, and 16 CPUs (25to 45 percent) at 20 to 30 percent lower processor consumption



DB2 database workload

- Benchmark: complex database warehouse application running on DB2 V10.1
- Upgrade to from z196 to z12EC provides
  - ✓ Improvements of throughput by 30.4 percent
  - ✓ Reduction of processor load
- Another 50.2 percent performance improvement we see when comparing z196 to z10

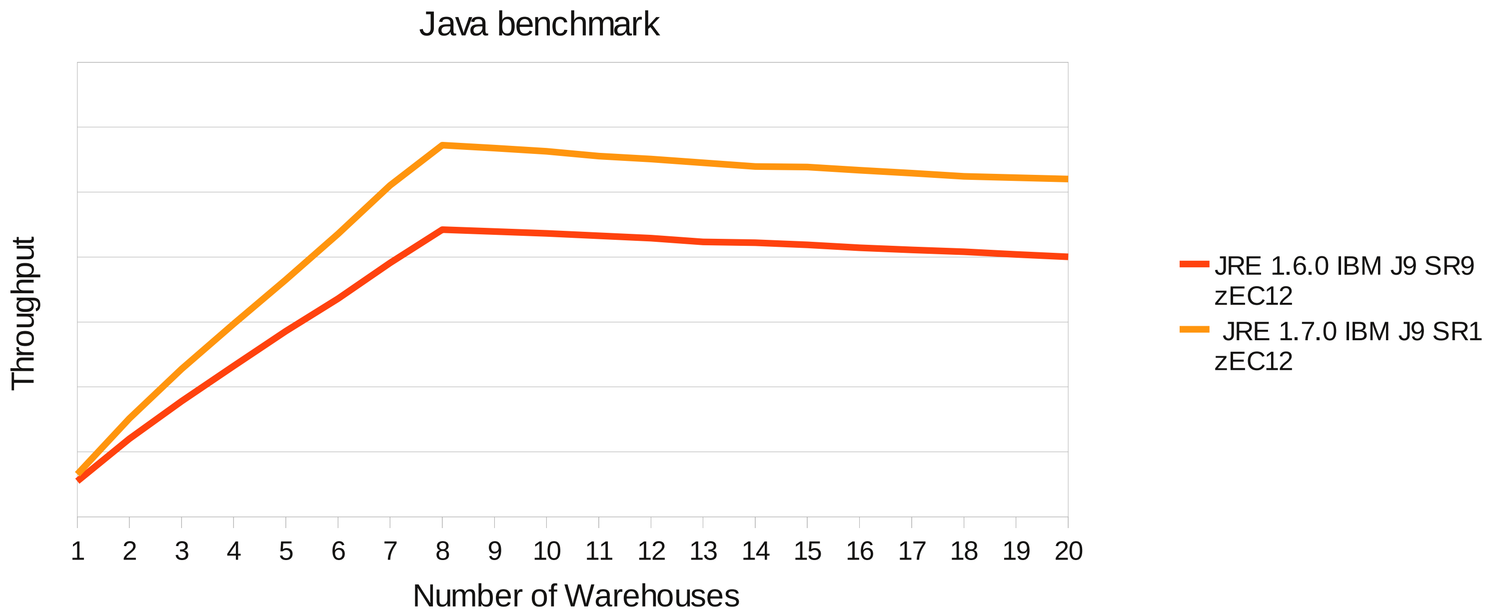


Agenda

- zEnterprise zEC12 design
- Linux performance comparison zEC12 and z196
- Performance improvements in other areas
  - ✓ Java JRE 1.7.0

Java – JRE 1.6.0 SR9 vs. JRE 1.7.0 SR1

- Business operation throughput improved by 29%
  - ✓ 2 GiB, 8CPU, 1 JVM, only Java versions substituted
    - JRE 1.6.0 IBM J9 2.4 SR9 20110624_85526 (JIT enabled, AOT enabled)
    - JRE 1.7.0 IBM J9 2.6 SR1 20120322_106209 (JIT enabled, AOT enabled)
- Similar improvements seen over the last years when upgrading to newer Java versions
  - ✓ Some software products are bundled with a particular Java version
  - ✓ In this case the software product needs an upgrade to profit of the improved performance



Summary

- Tremendous performance gains
  - ✓ Performance improvement seen in close to all areas measured yet
  - ✓ Often combined with processor consumption reduction
  - ✓ More improvement than just from higher rate to expect
    - Rate is up from 5.2 GHz to 5.5 GHz which means close to 6 percent higher
    - New cache setup with much bigger caches
    - Out-of-order execution of the second generation
    - Better branch prediction
- Some exemplary performance gains with Linux workloads
  - ✓ About 30 to 67 percent for Java
  - ✓ Up to 30 percent for complex database
  - ✓ Up to 31 percent for single threaded CPU intense
  - ✓ About 38 to 68 percent when scaling processors and/or processes
- Performance team has to measure more scenarios with intense disk access and network access when an exclusive z12EC GA measurement environment with required I/O options gets available
- No new zEC12 instructions exploited yet because no machine optimized GCC available in a supported distribution yet

Questions

- Further information is located at
  - ✓ Linux on System z – Tuning hints and tips
    http://www.ibm.com/developerworks/linux/linux390/perf/index.html
  - ✓ Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

IBM

Mario Held

Linux on System z
System Software
Performance Engineer

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

Phone +49 (0)7031–16–4257
Email mario.held@de.ibm.com