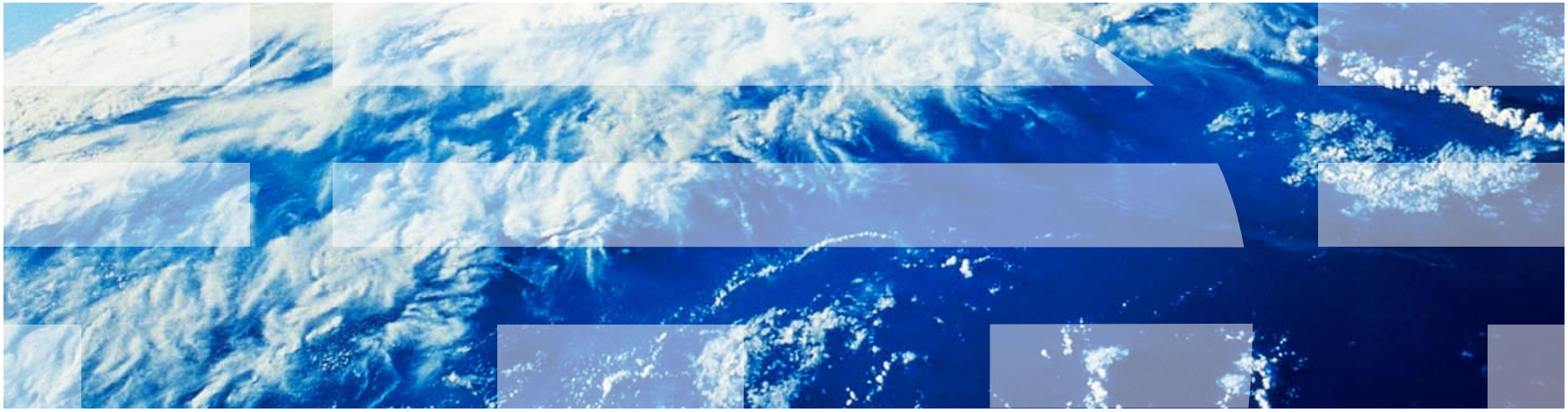
Martin Kammerer martin.kammerer@de.ibm.com

9/27/10

Linux Performance on IBM zEnterprise 196



visit us at http://www.ibm.com/developerworks/linux/linux390/perf/index.html

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.

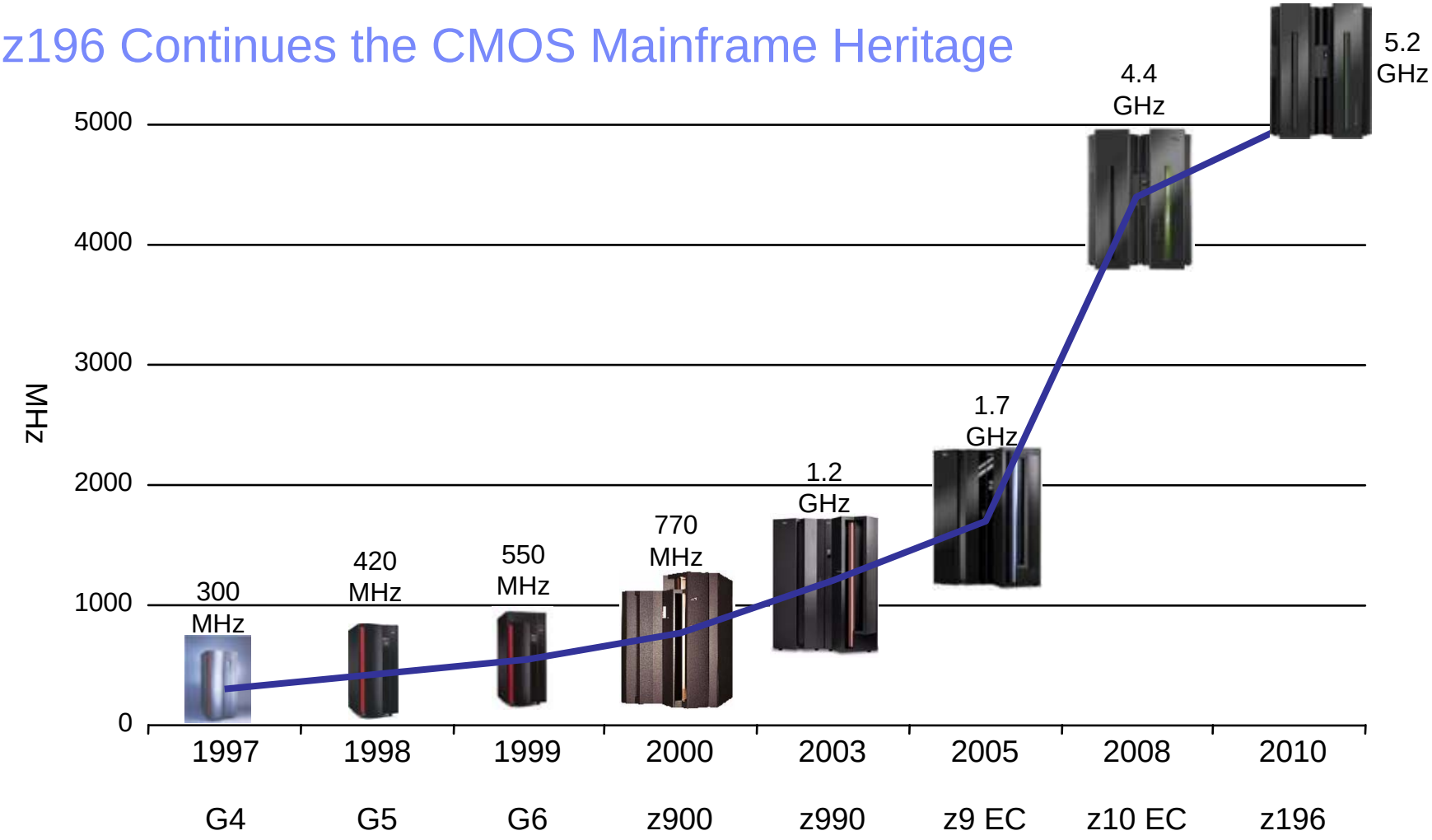
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

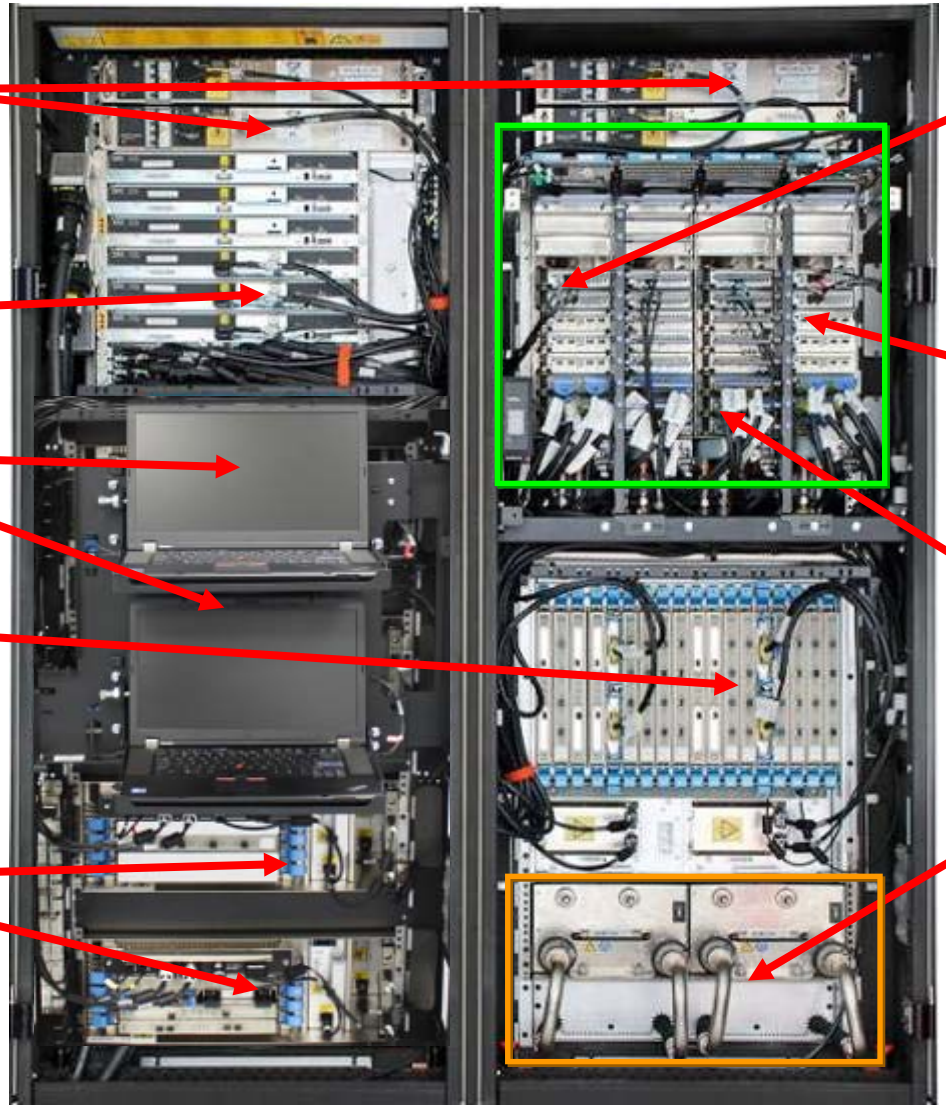
Agenda

- zEnterprise 196 design
- Linux performance comparison z196 and z10
- SPECCPU case study

z196 Continues the CMOS Mainframe Heritage



z196 Water cooled – Under the covers (Model M66 or M80) front view



Internal Batteries (optional)

Power Supplies

Support Elements

I/O cage

I/O drawers

Ethernet cables for internal System LAN connecting Flexible Service Processor (FSP) cage controller cards

Processor Books, Memory, MBA and HCA cards

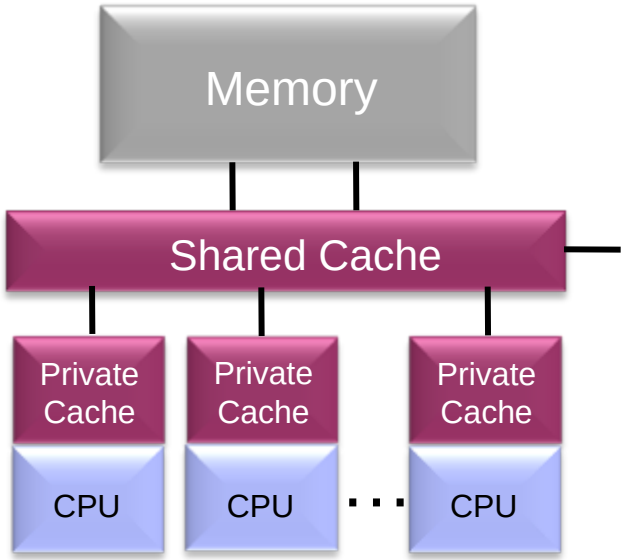
InfiniBand I/O Interconnects

2 x Water Cooling Units

z196 Processor Design Basics

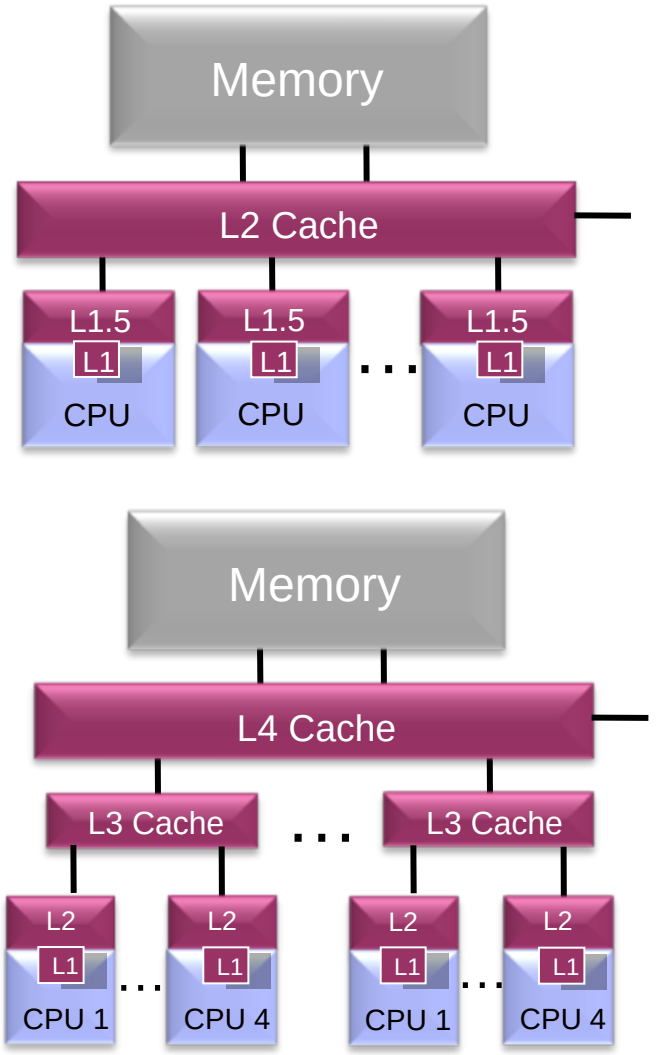
- **Processor Design**
  - CPU (core)
    - cycle time
    - pipeline, execution order
    - branch prediction
    - hardware vs. millicode
  - Memory subsystem
    - high speed buffers (caches)
      - on chip, on book
      - private, shared
    - buses
      - number, bandwidth
    - latency
      - distance
      - speed of light

Logical View of Single Book



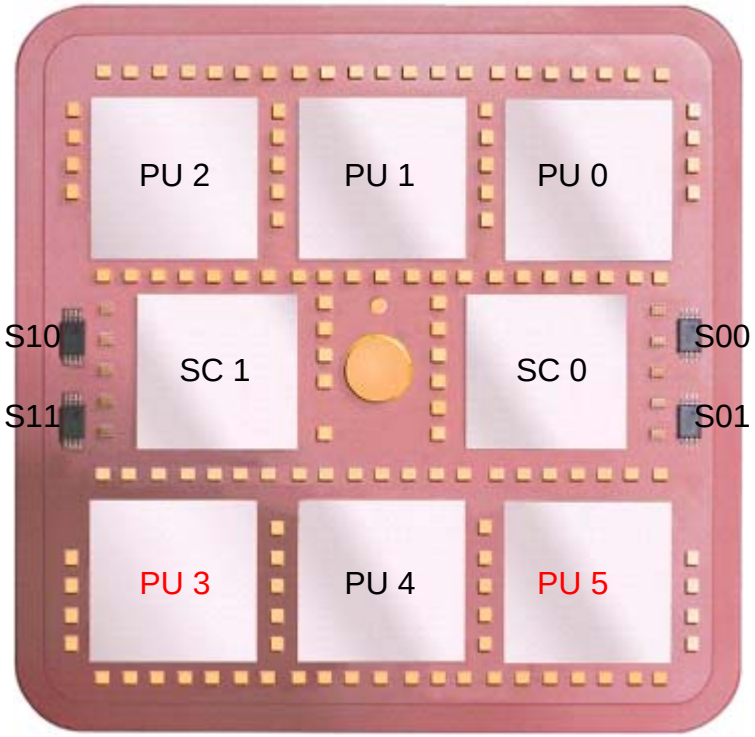
z196 vs. z10 hardware comparison

- **z10 EC**
  - CPU
    - 4.4 Ghz
  - Caches
    - L1 private 64k instr, 128k data
    - L1.5 private 3 MBs
    - L2 shared 48 MBs / book
    - book interconnect: star
- **z196**
  - CPU
    - 5.2 Ghz
    - Out-of-Order execution
  - Caches
    - L1 private 64k instr, 128k data
    - L2 private 1.5 MBs
    - L3 shared 24 MBs / chip
    - L4 shared 192 MBs / book
    - book interconnect: star



z196 Multi-Chip Module (MCM) Packaging

- **96mm x 96mm MCM**
  - 103 Glass Ceramic layers
  - 8 chip sites
  - 7356 LGA connections
  - 20 and 24 way MCMs
  - Maximum power used by MCM is 1800W



- **CMOS 12s chip Technology**
  - PU, SC, S chips, 45 nm
  - 6 PU chips/MCM – Each up to 4 cores
    - One memory control (MC) per PU chip
    - 1.4 billion transistors/PU chip
    - L1 cache/PU core
    - L2 cache/PU core
    - L3 cache shared by 4 PUs per chip
  - 2 Storage Control (SC) chip
    - 24.427 mm x 19.604 mm
    - 1.5 billion transistors/SC chip
    - L4 Cache 96 MB per SC chip (192 MB/Book)
    - L4 access to/from other MCMs
  - 4 SEEPROM (S) chips
    - 2 x active and 2 x redundant
    - Product data for MCM, chips and other engineering information
  - Clock Functions – distributed across PU and SC chips
    - Master Time-of-Day (TOD) function is on the SC

z196 PU core

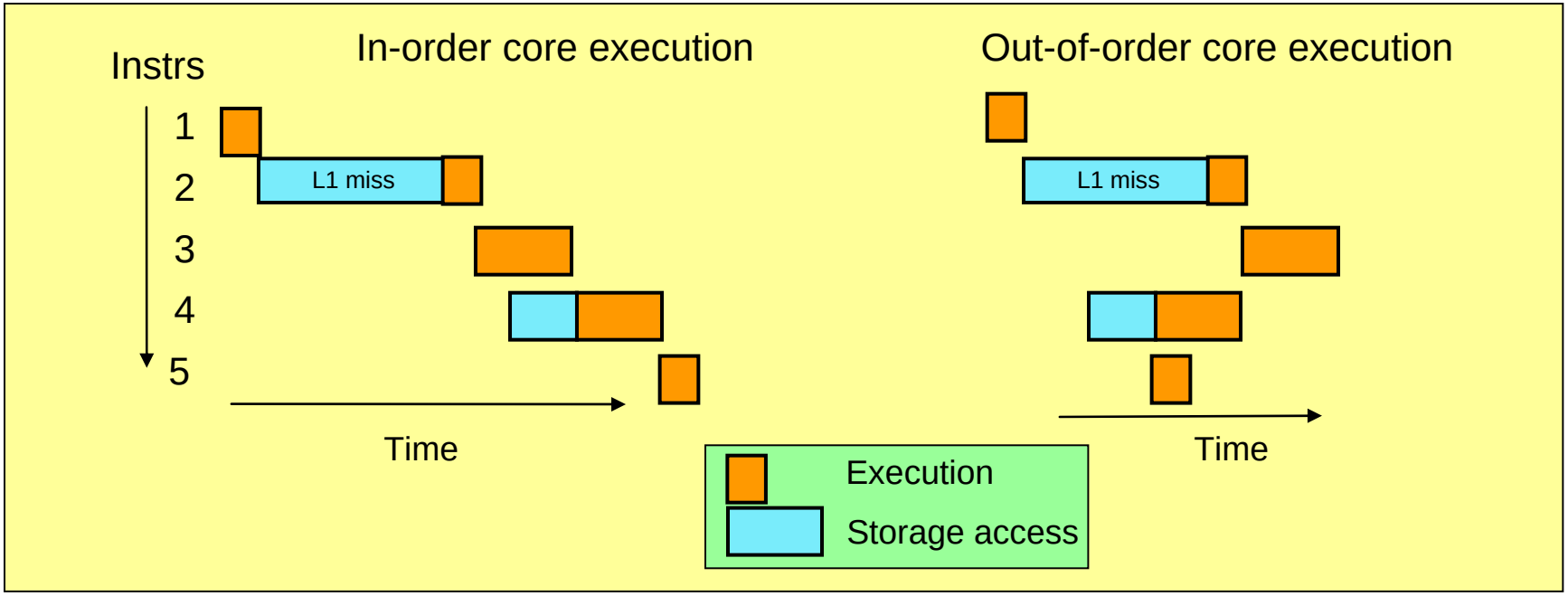
- **Each core is a superscalar, out of order processor with these characteristics:**
  - Six execution units
    - 2 fixed point (integer), 2 load/store, 1 binary floating point, 1 decimal floating point
  - Up to three instructions decoded per cycle (vs. 2 in z10)
  - 211 complex instructions cracked into multiple internal operations
    - 246 of the most complex z/Architecture instructions are implemented via millicode
  - Up to five instructions/operations executed per cycle (vs. 2 in z10)
  - Execution can occur out of (program) order
    - Memory address generation and memory accesses can occur out of (program) order
    - Special circuitry to make execution and memory accesses appear in order to software
  - Each core has 3 private caches
    - 64KB 1st level cache for instructions, 128KB 1st level cache of data
    - 1.5MB L2 cache containing both instructions and data

z196 New Instruction Set Architecture

- **Re-compiled code/apps get further performance gains through 110+ new instructions, e.g.:**
- **High-Word Facility (30 new instructions)**
  - Independent addressing to high word of 64-bit GPRs
  - Effectively provides compiler/ software with 16 additional 32-bit registers
- **Interlocked-Access Facility (12 new instructions)**
  - Interlocked (atomic) load, value update and store operation in a single instruction
  - Immediate exploitation by Java
- **Load/Store-on-Condition Facility (6 new instructions)**
  - Load or store conditionally executed based on condition code
  - Dramatic improvement in certain codes with highly unpredictable branches
- **Distinct-Operands Facility (22 new instructions)**
  - Independent specification of result register (different than either source register)
  - Reduces register value copying
- **Population-Count Facility (1 new instruction)**
  - Hardware implementation of bit counting ~5x faster than prior software implementations
- **Integer to/from Floating point converts (21 new instructions)**

z196 Out of Order (OOO) Value

- **OOO yields significant performance benefit for compute intensive apps through**
  - Re-ordering instruction execution
    - Later (younger) instructions can execute ahead of an older stalled instruction
  - Re-ordering storage accesses and parallel storage accesses
- **OOO maintains good performance growth for traditional apps**



z196 Capacity/Performance compared to z10 EC

	z196 to z10 EC Ratios	z196 PCI Values
LSPR Mixed workload average, multi-image for z/OS 1.11 with HiperDispatch active on z196		
Uni-processor	1.33	1202
16-way z196 to 16-way z10 EC	1.38	14371
32-way z196 to 32-way z10 EC	1.38	25241
64-way z196 to 64-way z10 EC	1.41	44953
80-way z196 to 64-way z10 EC	1.64	52286

PCI - Processor Capacity Index

Agenda

- zEnterprise 196 design
- Linux performance comparison z196 and z10
- SPECCPU case study

Environment

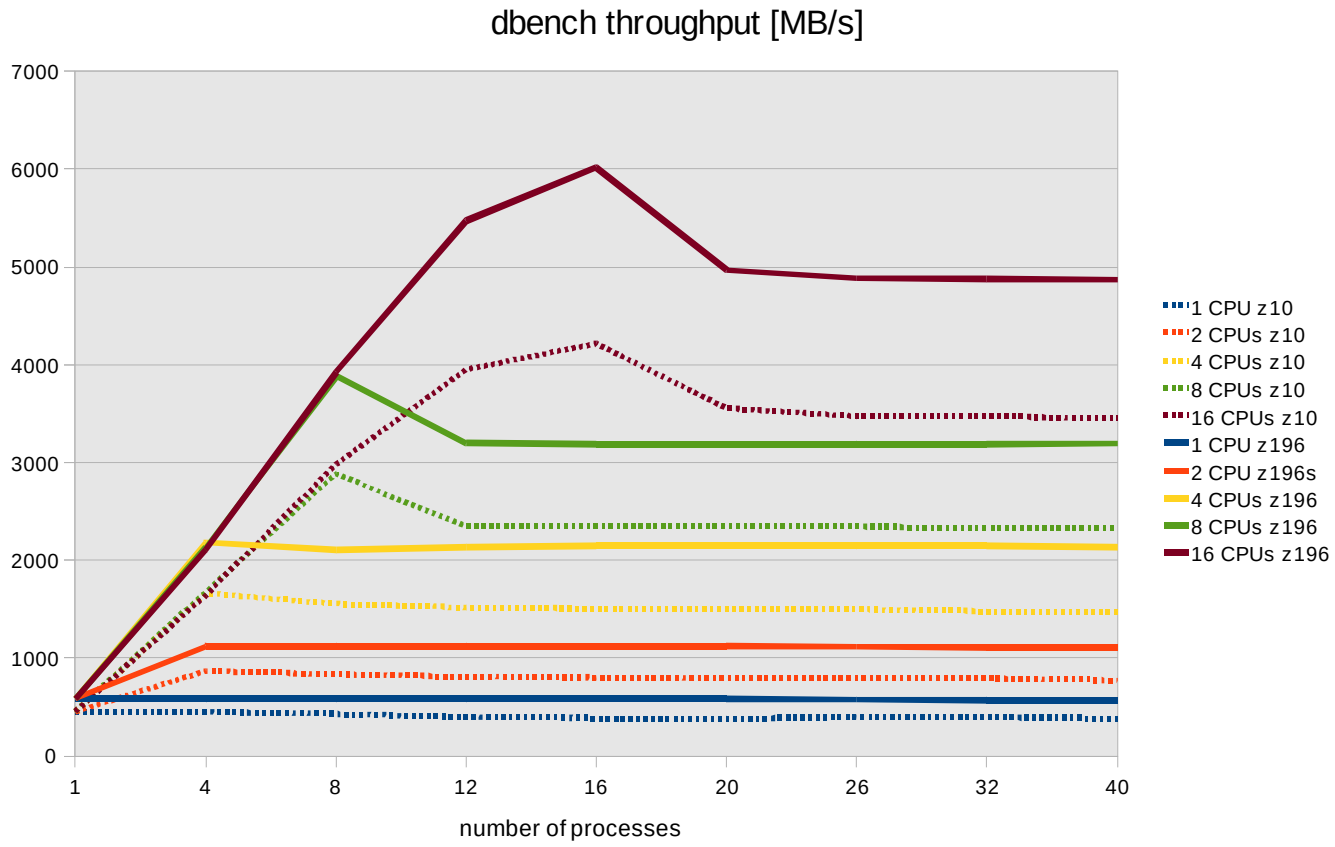
- Hardware
  - z196: 2817-718 M49
  - z10 : 2097-726 E26
  - z9 : 2094-718 S18
- Linux
  - Linux distribution with recent kernel
    - SLES11 SP1: 2.6.32.12
    - Linux in LPAR
    - Shared processors
    - Other LPARs deactivated

File server benchmark description

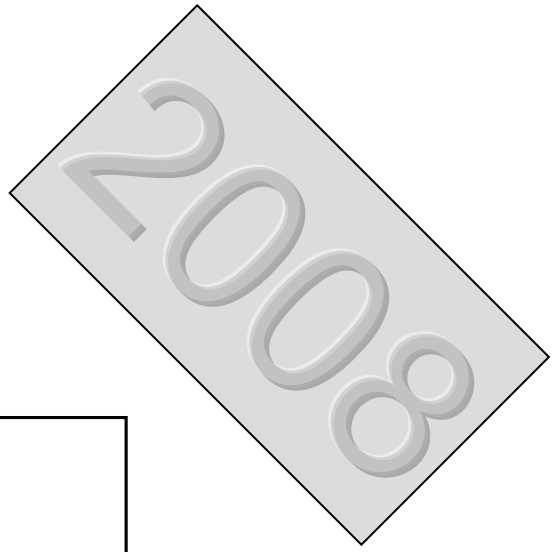
- dbench 3
  - Emulation of Netbench benchmark
  - Generates file system load on the Linux VFS
  - Does the same I/O calls like the smbd server in Samba (without networking calls)
  - Mixed file operations workload for each process: create, write, read, append, delete
  - Measures throughput of transferred data
  - Configuration
    - 2 GB 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

dbench

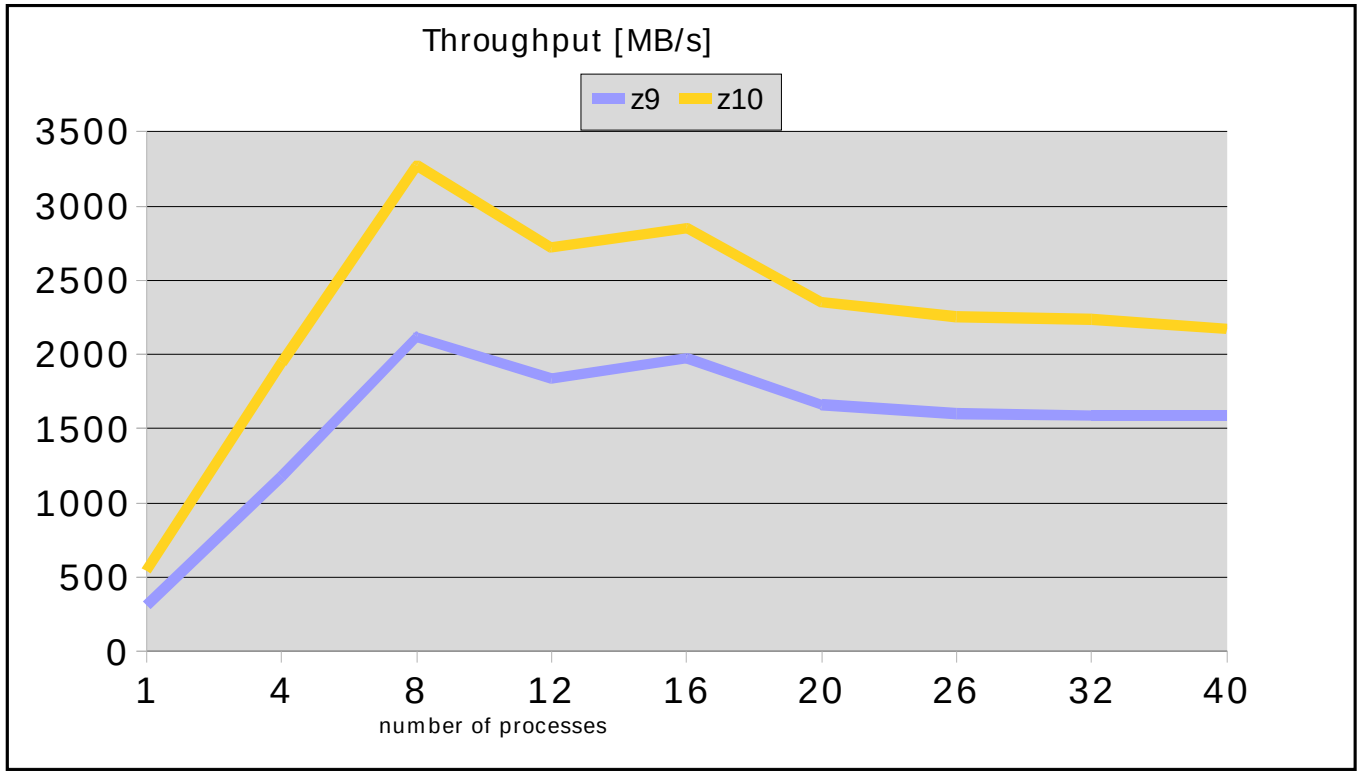
- SLES11 improves on average by 40%



z10 versus z9



- Improvement z10 versus z9
  - Measured with 8 CPUs: average improvement is 50%



Kernel benchmark description

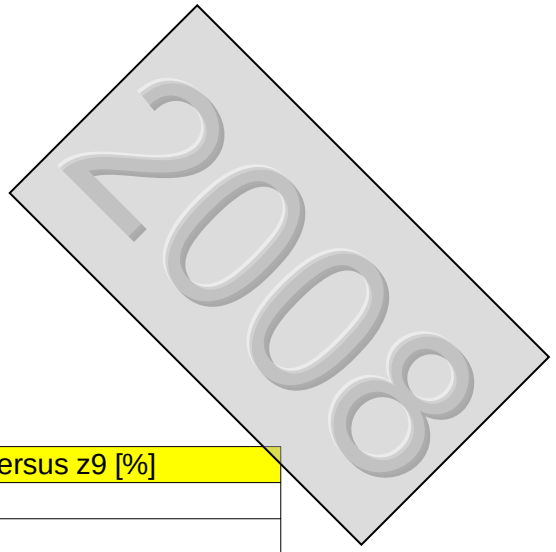
- lmbench 3
  - Suite of operating system microbenchmarks
  - Focuses on interactions between the operating system and the hardware architecture
  - Bandwidth measurements for memory and file access, operations and movement
  - Latency measurements for process handling and communication
  - Latency measurements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

Imbench

- Most benefits in L3 and L4 cache

all numbers are deviation z196 versus z10 [%]	**SLES11**
simple syscall	-40
simple read/write	0
select of file descriptors	30
signal handler	-25
process fork	30
libc bcopy aligned L1 / L2 / L3 / L4 cache	10 / 10 / 100 / 300
libc bcopy unaligned L1 / L2 / L3 / L4 cache	15 / 0 / 0 / 40
memory bzero L1 / L2 / L3 / L4 cache	35 / 90 / 300 / 800
memory partial read L1 / L2 / L3 / L4 cache	45 / 40 / 130 / 500
memory partial read/write L1 / L2 / L3 / L4 cache	20 / 20 / 20 / 150
memory partial write L1 / L2 / L3 / L4 cache	80 / 30 / 60 / 300
memory read L1 / L2 / L3 / L4 cache	10 / 30 / 50 / 300
memory write L1 / L2 / L3 / L4 cache	50 / 30 / 30 / 180
Mmap read L1 / L2 / L3 / L4 cache	50 / 35 / 85 / 300
Mmap read open2close L1 / L2 / L3 / L4 cache	30 / 0 / 15 / 150
Read L1 / L2 / L3 / L4 cache	10 / 25 / 80 / 300
Read open2close L1 / L2 / L3 / L4 cache	15 / 30 / 85 / 300
Unrolled bcopy unaligned L1 / L2 / L3 / L4 cache	100 / 75 / 75 / 200

z10 versus z9



- in many cases improvements of 25%, peaks at 100%
- some results for context switch tests decreased

Area	z10 improvement versus z9 [%]
simple syscall	30
simple read/write	30/40
select of file descriptors	60
signal handler overhead	50
process fork	30
libc bcopy aligned L1/L1.5/L2cache/main memory	130 / 300 / 10 / **-10**
libc bcopy aligned L1/L1.5/L2cache/main memory	100 / **-20 / -30 / -20**
memory bzero L1/L1.5/L2cache/main memory	80 / 50 / **-40 / -60**
memory partial read L1/L1.5/L2cache/main memory	150 / 250 / 30 / 15
memory partial read/write L1/L1.5/L2cache/main memory	100 / 270 / 130 / 15
memory partial write L1/L1.5/L2cache/main memory	100 / 270 / 130 / 15
memory read L1/L1.5/L2cache/main memory	120 / 200 / 100 / 10
memory write L1/L1.5/L2cache/main memory	40 / 80 / 60 / 15
Mmap read L1/L1.5/L2cache/main memory	60 / 200 / 50 / 5
Mmap read open2close L1/L1.5/L2cache/main memory	50 / 70 / 40 / 0
Read L1/L1.5/L2cache/main memory	80 / 130/ 60 / 0
Read open2close L1/L1.5/L2cache/main memory	80 / 130/ 60 / 5
Unrolled bcopy unaligned L1/L1.5/L2cache/main memory	70 / 120/ 70 / 60

Java benchmark description

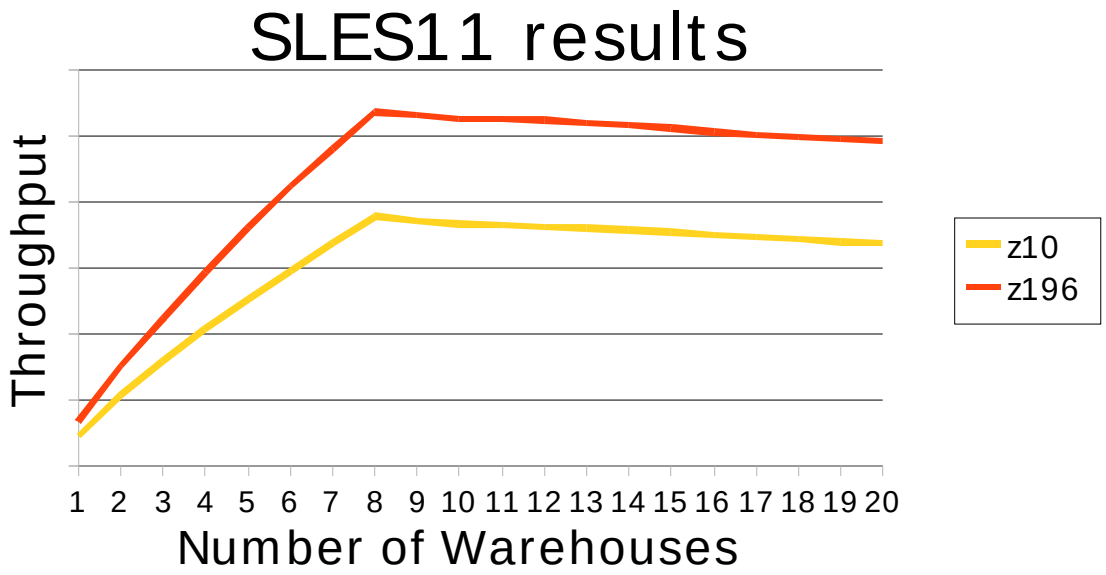
- Java server benchmark
  - evaluates the performance of server side Java
  - Exercises
    - Java Virtual Machine (JVM)
    - Just-In-Time compiler (JIT)
    - Garbage collection
    - Multiple threads
    - Simulates real-world applications including XML processing or floating point operations
  - Can be used to measure performance of CPUs, memory hierarchy and scalability
  - Configurations
    - 8 processors, 2 GB memory, 1 JVM
    - 16 processors, 8 GB memory, 4 JVMs
    - Java Version 6

SpecJBB

- Overall throughput improvement: close to 45%

throughput improvement [%]

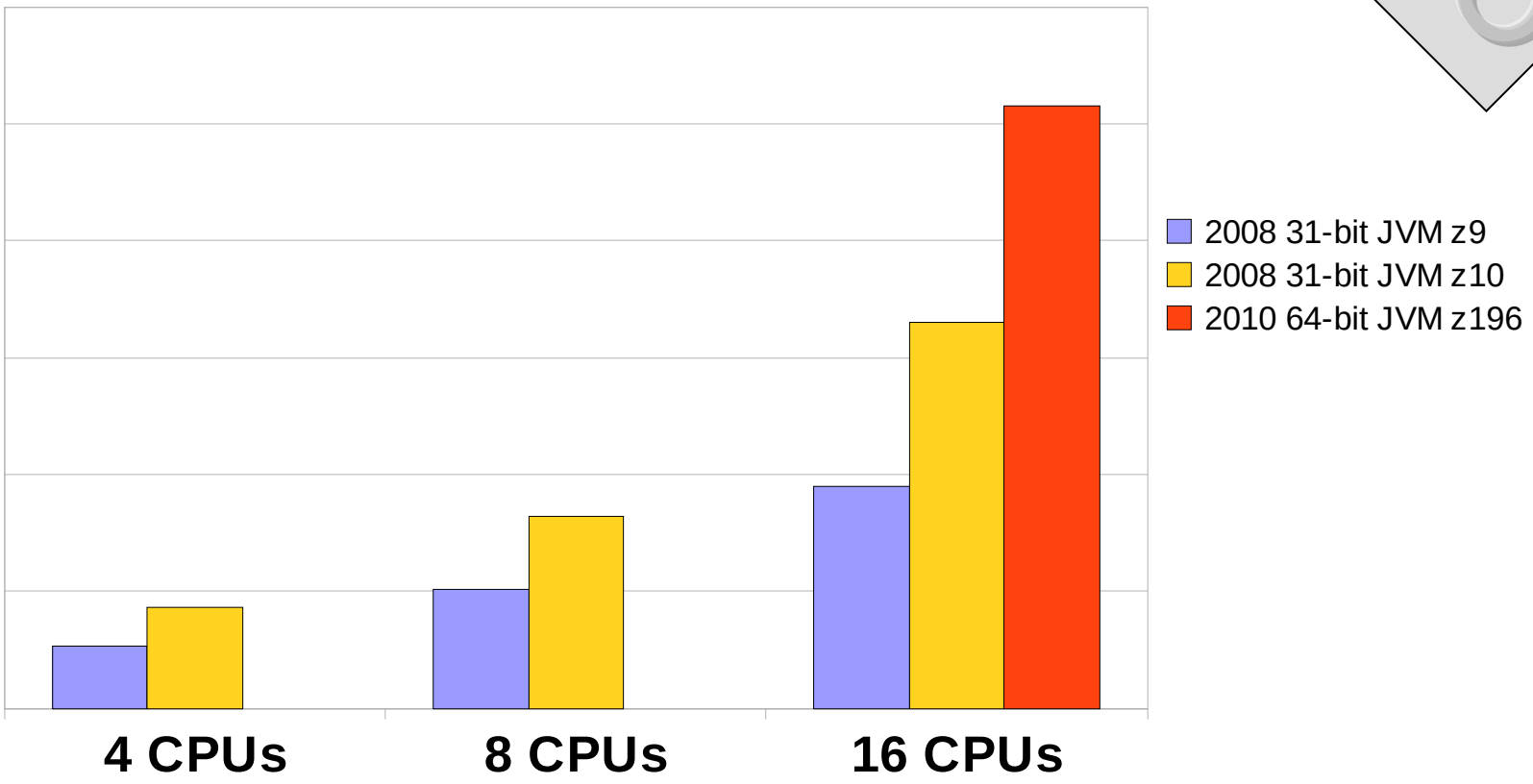
SLES11 SP1	comment
44	2 GB, 8 CPUs
45	8 GB, 16 CPUs, 4 JVMs



SpecJBB throughput compared to last generations

2008

- In 2008 the message was
  - 60% more throughput with z10 than z9

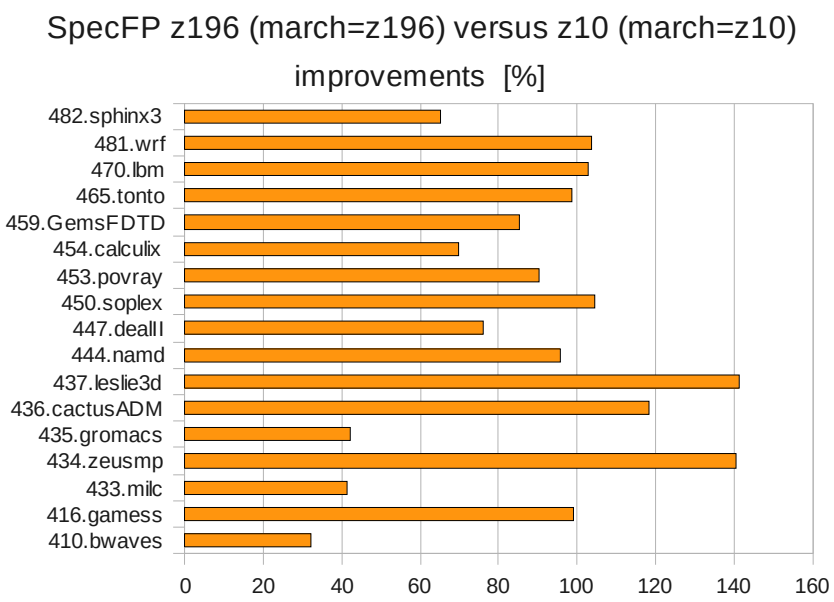


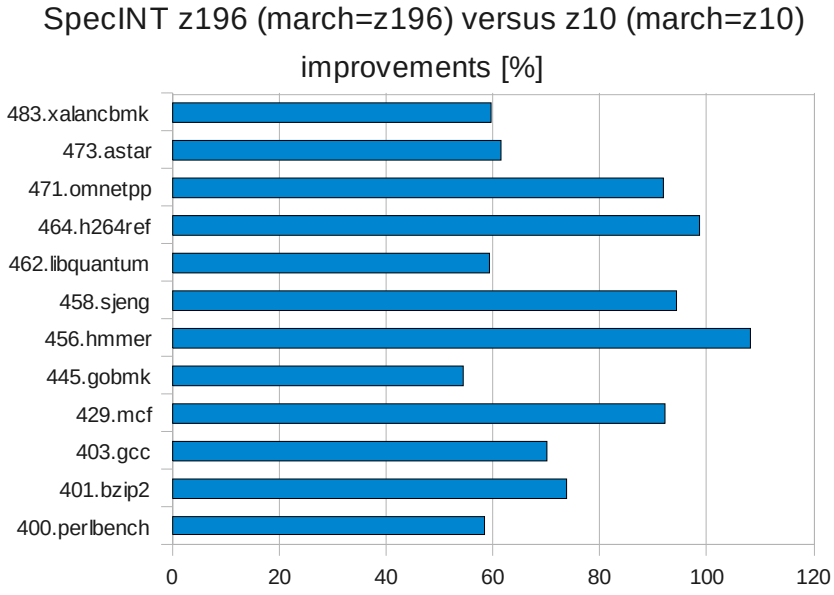
CPU-intensive benchmark suite

- 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 GB memory
  - Executing one test case at a time

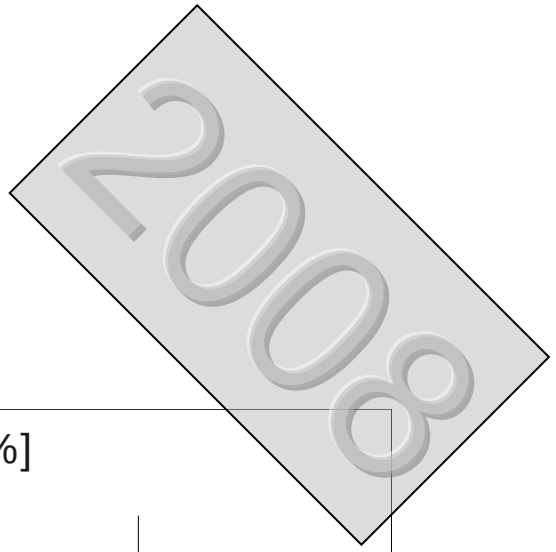
SpecCPU

- Linux: Internal driver, kernel 2.6.29, gcc 4.5, glibc 2.9.3
- SpecFP suite improves by 86%
- SpecINT suite improves by 76%

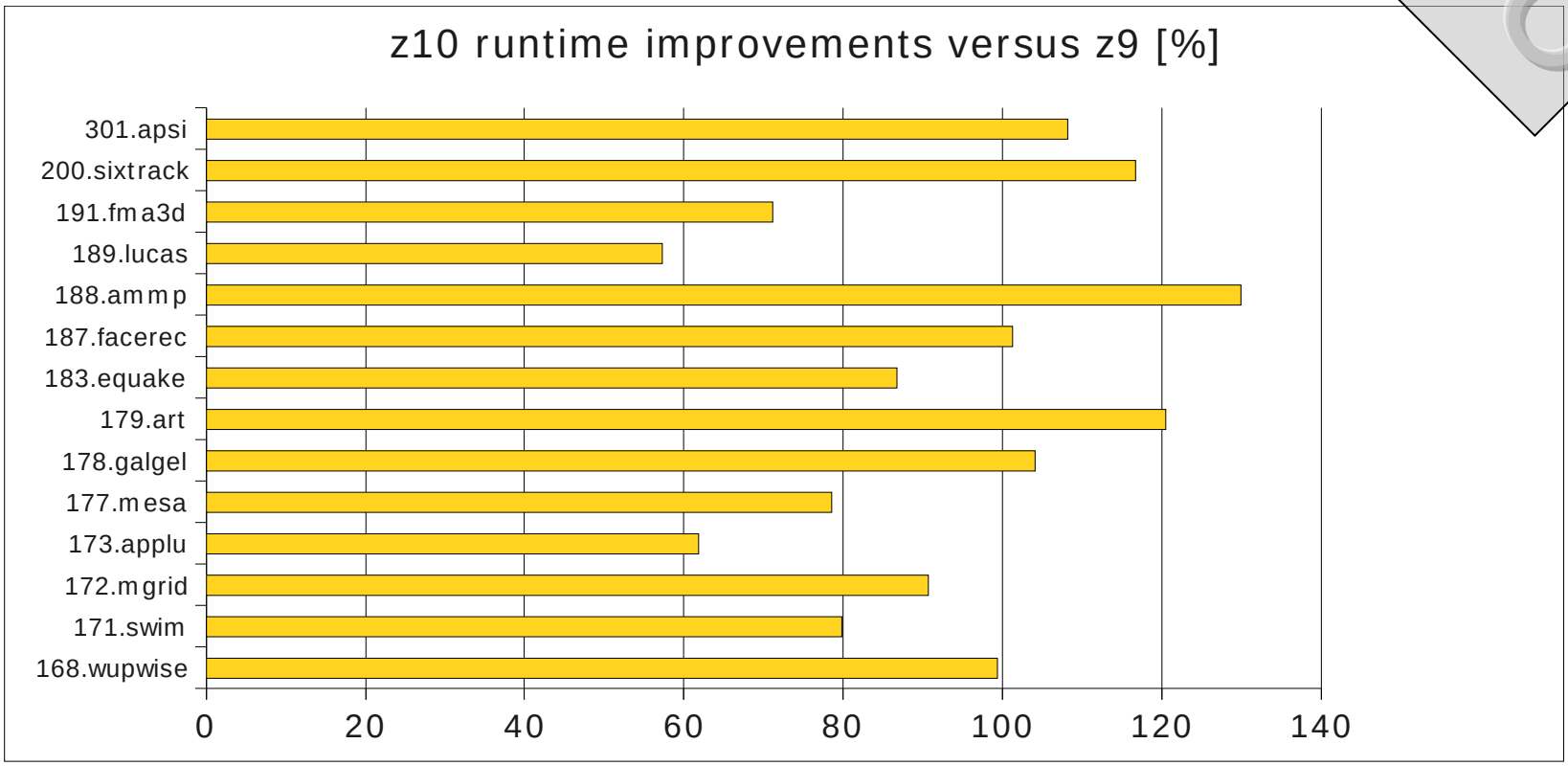




z10 versus z9



- Overall improvement with z10 versus z9: 90%



Agenda

- zEnterprise 196 design
- Linux performance comparison z196 and z10
- SPECCPU case study

SPECINT2006, 456.hmmer

- Search for patterns in a gene sequence database
- Use of Profile Hidden Markov Models
- Programming Language: C (57 source files)

Function Profile

%Total	Cumulative	Function
95.6	95.6	P7Viterbi
1.9	97.5	sre_random
1.8	99.3	FChoose

456.hmmer, hotloop

```
for (k = 1; k <= M; k++) {
      mc[k] = mpp[k-1]   + tpmm[k-1];
      if ((sc = ip[k-1]  + tpim[k-1]) > mc[k])  mc[k] = sc;
      if ((sc = dpp[k-1] + tpdm[k-1]) > mc[k])  mc[k] = sc;
      if ((sc = xmb  + bp[k])         > mc[k])  mc[k] = sc;
      mc[k] += ms[k];
      if (mc[k] < -INFTY) mc[k] = -INFTY;
      dc[k] = dc[k-1] + tpdd[k-1];
      if ((sc = mc[k-1] + tpmd[k-1]) > dc[k]) dc[k] = sc;
      if (dc[k] < -INFTY) dc[k] = -INFTY;
      if (k < M) {
ic[k] = mpp[k] + tpmi[k];
if ((sc = ip[k] + tpii[k]) > ic[k]) ic[k] = sc;
ic[k] += is[k];
if (ic[k] < -INFTY) ic[k] = -INFTY;
      }
```

456.hmmer, hotloop, code generated for z10

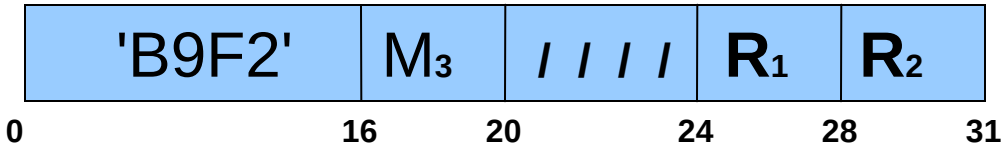
```
if ((sc = xmb  + bp[k])          > mc[k])  mc[k] = sc;

GCC generates:

1834                                                  LR GPR3,GPR4
1846          LR GPR4,GPR6
E380F0B00004  LG GPR8,176(,GPR15)   Problem:
5A481004      A  GPR4,4(GPR8,GPR1)  Address Generation Interlock
1943          CR GPR4,GPR3
A7240088      BHRC *+272            Problem:
                                    Branch Mispredict
...
50412004      ST GPR4,4(GPR1,GPR2)
a7F4FF79      BRC *-270
```

LOCR, Load on Condition

LOCR R_1,R2,M3 **RRF-c**



The second operand is placed unchanged at the first operand location if the condition code has one of the values specified by M_3; otherwise, the first operand remains unchanged.

456.hmmer, hotloop, code generated for z196

```
if ((sc = xmb  + bp[k])          > mc[k])  mc[k] = sc;
```

GCC generates:

```
5AA06000     A        GPR10,0(,GPR6)
191A         CR       GPR1,GPR10
B9F2501A     LOCRNHE  GPR1,GPR10
50102004     ST       GPR1,4(,GPR2)
```

~70% more throughput for this code compared to the z10-code both running on z196

Summary

- z196 performance advantages
  - Higher clock speed
  - More cache
  - OOO processing
  - New instructions
  - More processors
  - Processor scalability
- Performance gains with Linux workloads
  - Up to 45% for Java
  - Up to 86% for single threaded CPU intense
  - 40% when scaling processors and/or processes
  - SLES11 SP1 improvements

Thank You. Questions??