Computer Architecture

Performance

Performance Example

	Seoul to Pusan	Speed	Passengers
Sonata	10 hours	100 km/h	5
Boeing 727	1 hour	1000km/h	100
Concorde	0.5 hour	2000km/h	20

Time – the first type of performance

- *Wall-clock time*, *response time*, or *elapsed time*
  - actual time from start to completion
  - includes everything: CPU time for other programs as well as for itself, I/O, operating system overheads, etc
- *CPU (execution) time*
  - CPU time spent for a given program
  - user CPU time + system CPU time
  - e.g., results of UNIX `time` command

```
90.7u  12.9s  2:39  65%
```

Decomposition of CPU (Execution) Time

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$

$$= \frac{\text{Cycles}}{\text{Program}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

More on CPI (Clocks or Cycles Per Instruction)

- CPI = $\dfrac{\sum_{i=1}^{n} (CPI_i \times I_i)}{\text{Instruction Count}}$

- CPI Example

Instruction Class	Frequency	CPI_i
ALU operations	43%	1
Loads	21%	2
Stores	12%	2
Branches	24%	2

$$CPI = 0.43 \times 1 + 0.21 \times 2 + 0.12 \times 2 + 0.24 \times 2$$

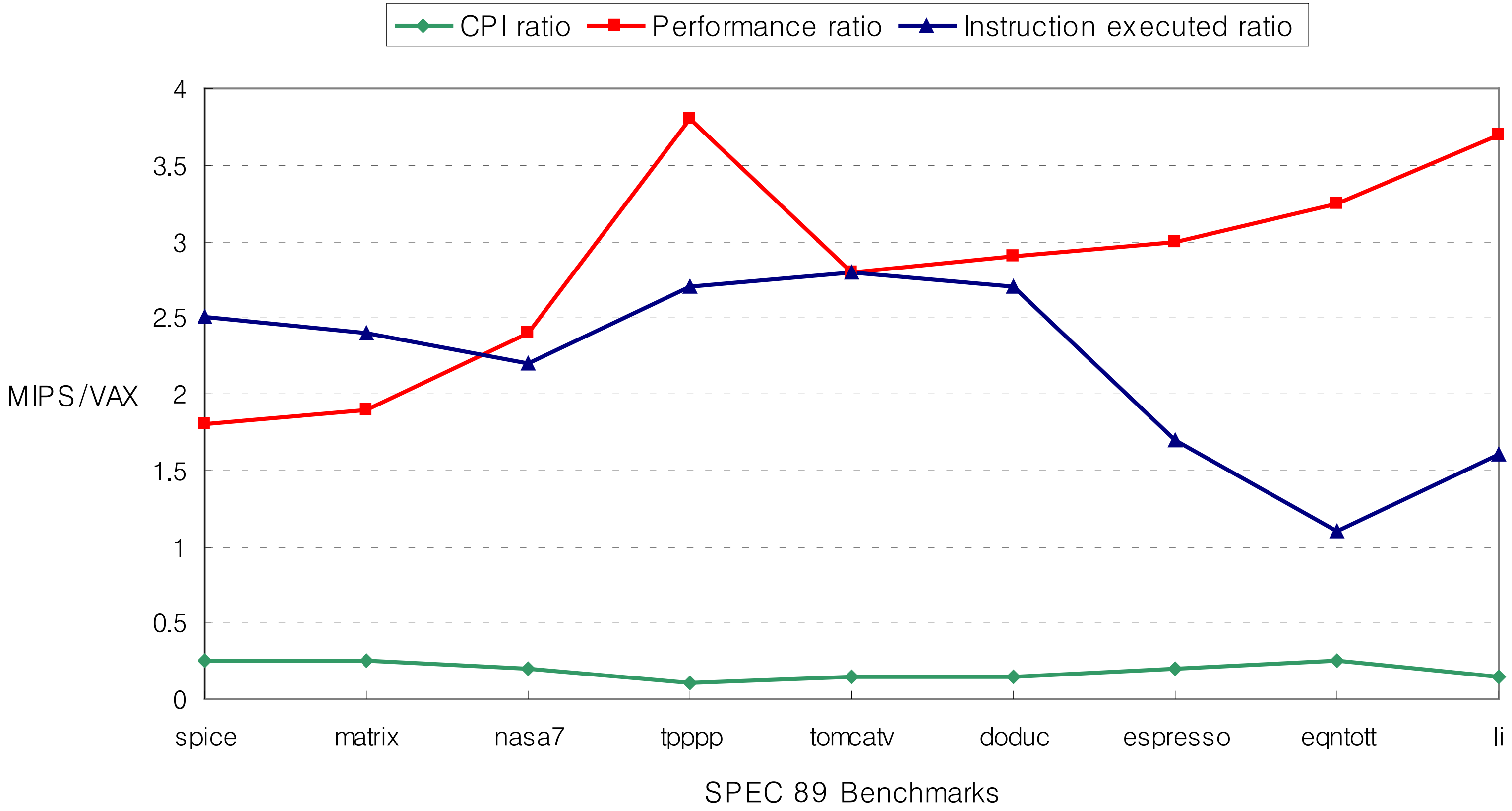
Factors involved in the CPU Time

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Instructions / Program	Cycles / Instruction	Seconds / Cycle
Program	V		
Compiler	V		
ISA	V	V	
Organization		V	V
Technology			V

RISC vs. CISC Arguments

- **MIPS (typical RISC) vs. VAX8700 (typical CISC)**



Source : Hennessy & Patterson *Computer Architecture: A Quantitative Approach, 3rd Ed.(p.108)*, Morgan Kaufmann, 2002

Rate – the second type of performance

- MIPS (million instructions per second)
  - MIPS = $\dfrac{\text{Instruction count}}{\text{Execution time} \times 10^6}$
  - Specifies performance (roughly) inversely to execution time
  - Easy to understand; faster machines means bigger MIPS
  - Problems
    - It does not take into account the capabilities of the instructions.
    - It varies between programs on the same computer.
    - It can even vary inversely with performance!!
- MFLOPS (million floating-point operations per second)

Ratio - the third type of performance

- "X is n times faster than Y" means :

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

- "X is n% faster than Y" means :

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 1 + \frac{n}{100}$$

- "X is n order of magnitude faster than Y" means :

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 10^n$$

How to Summarize Performance

- Arithmetic mean $\frac{1}{n}\sum_{i=1}^{n} T_i$

  (Time)

- Harmonic mean $\frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}}$

  (Rate)

- Geometric mean $\sqrt[n]{\prod_{i=1}^{n} Ratio_i}$

  (Ratio)

Arithmetic Mean

- Used to summarize performance given in times
  - Average Execution Time=($\sum_{i=1}^{n}$ Execution Times) / n
  - Assumes each benchmark is run an equal no. of times

- Weighted Arithmetic Mean
  - Weighted Average Execution Time =

$$\sum_{i=1}^{n} (W_i \times \text{Execution Times}) \;/\; \sum_{i=1}^{n} W_i$$

  - One possible weight assignment: equal execution time on some machine

Harmonic Mean

- Used to summarize performance given in rates (e.g., MIPS, MFLOPS):
  - Harmonic Mean = n / $\sum_{i=1}^{n} (1 / R_i)$
  - Example
    - Four programs execute at 10, 100, 50 and 20 MFLOPS, respectively
    - Harmonic mean is 4 / (1/10 + 1/100 + 1/50 + 1/20) = 22.2 MFLOPS
- Weighted Harmonic Mean
  - Weighted Harmonic Mean = $\sum_{i=1}^{n} W_i \; / \; \sum_{i=1}^{n} (W_i \; / \; R_i)$

Benchmarks

- A benchmark is distillation of the attributes of a workload
- Domain specific
- Desirable attributes
  - Relevant: meaningful within the target domain
  - Coverage: does not oversimplify important factors in the target domain
  - Understandable
  - Good metric(s)
  - Scaleable
  - Acceptance: vendors and users embrace it
- Two de facto industry standard benchmarks
  - SPEC: CPU performance
  - TPC: OLTP (On-Line Transaction Processing) performance

Benchmarks

- Benefits of good benchmarks
  - Good benchmarks
    - Define the playing field
    - Accelerate progress
      - Engineers do a great job once objective is measurable and repeatable
    - Set the performance agenda
      - Measure release-to-release progress
      - Set goals
  - Lifetime of benchmarks
    - Good benchmarks drive industry and technology forward
    - At some point, all reasonable advances have been made
    - Benchmarks can become counter productive by engineering artificial optimizations
    - So, even good benchmarks become obsolete over time

Grand Example 1 – SPEC Benchmark

- SPEC (Standard Performance Evaluation Corporation)
  - Formed in 1988 to establish, maintain, and endorse a standardized set of relevant benchmarks and metrics for performance evaluation of modern computer systems
  - Founded by EE times, Sun, MIPS, HP, Apollo, DEC
  - For more info, see http://www.spec.org
- Create standard list of programs, inputs, reporting rules:
  - Based on real programs and includes real system effects
  - Specifies rules for running and reporting
  - Measures observed execution time

Grand Example 1 – SPEC Benchmark

CINT2000 Result

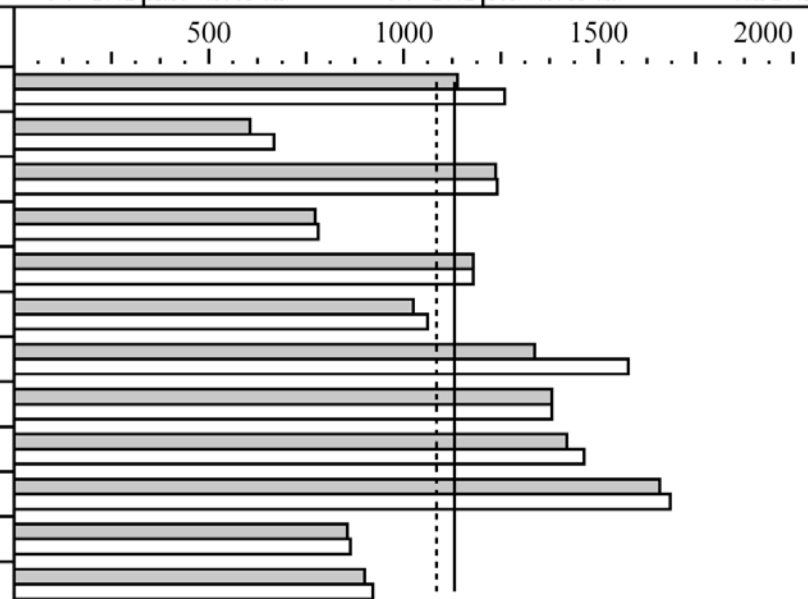
Copyright ©1999-2002, Standard Performance Evaluation Corporation

Dell
Precision WorkStation 350 (3.06 GHz P4)

SPECint2000 =	1130
SPECint_base2000 =	1085

SPEC license #: 55	Tested by: Dell, Round Rock, TX	Test date: Nov-2002	Hardware Avail: Nov-2002	Software Avail: Jun-2002

Benchmark	Reference Time	Base Runtime	Base Ratio	Runtime	Ratio
164.gzip	1400	123	1138	111	1260
175.vpr	1400	231	606	210	668
176.gcc	1100	89.0	1236	88.7	1240
181.mcf	1800	233	773	231	781
186.crafty	1000	84.8	1179	84.8	1179
197.parser	1800	176	1025	170	1062
252.eon	1300	97.2	1337	82.5	1577
253.perlbmk	1800	130	1381	130	1380
254.gap	1100	77.5	1419	75.2	1463
255.vortex	1900	115	1658	113	1685
256.bzip2	1500	175	856	174	863
300.twolf	3000	333	900	326	922



Grand Example 1 – SPEC Benchmark

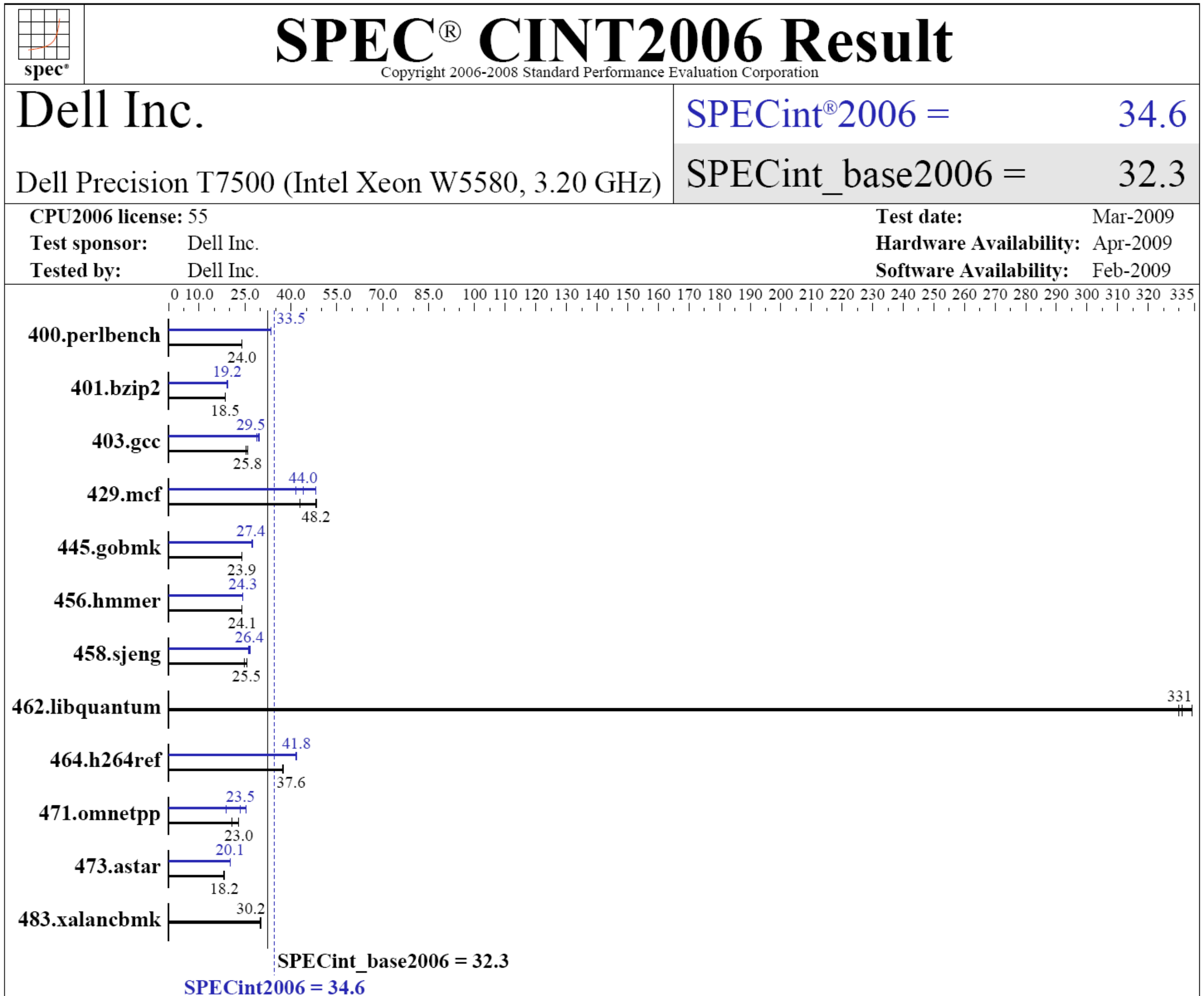
Hardware		Software	
CPU:	Intel Pentium 4 (533 MHz system bus)	Operating System:	Windows XP Professional
CPU MHz:	3066	Compiler:	Intel C++ Compiler 6.0 (020613Z) Microsoft Visual Studio .NET (7.0.9466) MicroQuill SmartHeap Library 6.01
FPU:	Integrated	File System:	NTFS
CPU(s) enabled:	1	System State:	Default
CPU(s) orderable:	1		
Parallel:	No		
Primary Cache:	12K(I) micro-ops + 8KB(D) on chip		
Secondary Cache:	512KB(I+D) on chip		
L3 Cache:	N/A		
Other Cache:	N/A		
Memory:	2 x 256MB PC1066-32 ECC RDRAM		
Disk Subsystem:	1 x 20GB Maxtor 7.2K ATA/100		
Other Hardware:			

Grand Example 1 – SPEC Benchmark

Notes/Tuning Information

```
PORTABILITY FLAGS
  176.gcc:      -Dalloca=_alloca -F10000000
  186.crafty:   -DNT_i386
  253.perlbmk: -DSPEC_CPU2000_NTOS -DPERLDLL -MT
  254.gap:      -DSYS_HAS_CALLOC_PROTO -DSYS_HAS_MALLOC_PROTO
FEEDBACK-DIRECTED OPTIMIZATION
  FDO:          PASS1= -Qprof_gen  PASS2= -Qprof_use
BASE TUNING
  C:            -Qipo -QxW +FDO shlW32M.lib
  C++:          -QxW -GX -GR +FDO
PEAK TUNING
  164.gzip:     -Qipo -QxW -O3 +FDO
  175.vpr:      -Qipo -QxW -O3 +FDO
  176.gcc:      -Qipo -QxW -O3 +FDO
  181.mcf:      -Qipo -QxW -O3 +FDO shlW32M.lib
  186.crafty:   -Qipo -QxW -O3 +FDO
  197.parser:   -Qipo -QxW -O3 +FDO
  252.eon:      -Qipo -QxW -O3 +FDO
  253.perlbmk: -Qipo -QxW +FDO shlW32M.lib
  254.gap:      -Qipo -QaxW -O3 +FDO
```

SPEC Benchmark Update (2006)



SPEC Benchmark Update (2006)

SPEC CINT2006 Result

Dell Inc.

Dell Precision T7500 (Intel Xeon W5580, 3.20 GHz)

SPECint2006 = 34.6

SPECint_base2006 = 32.3

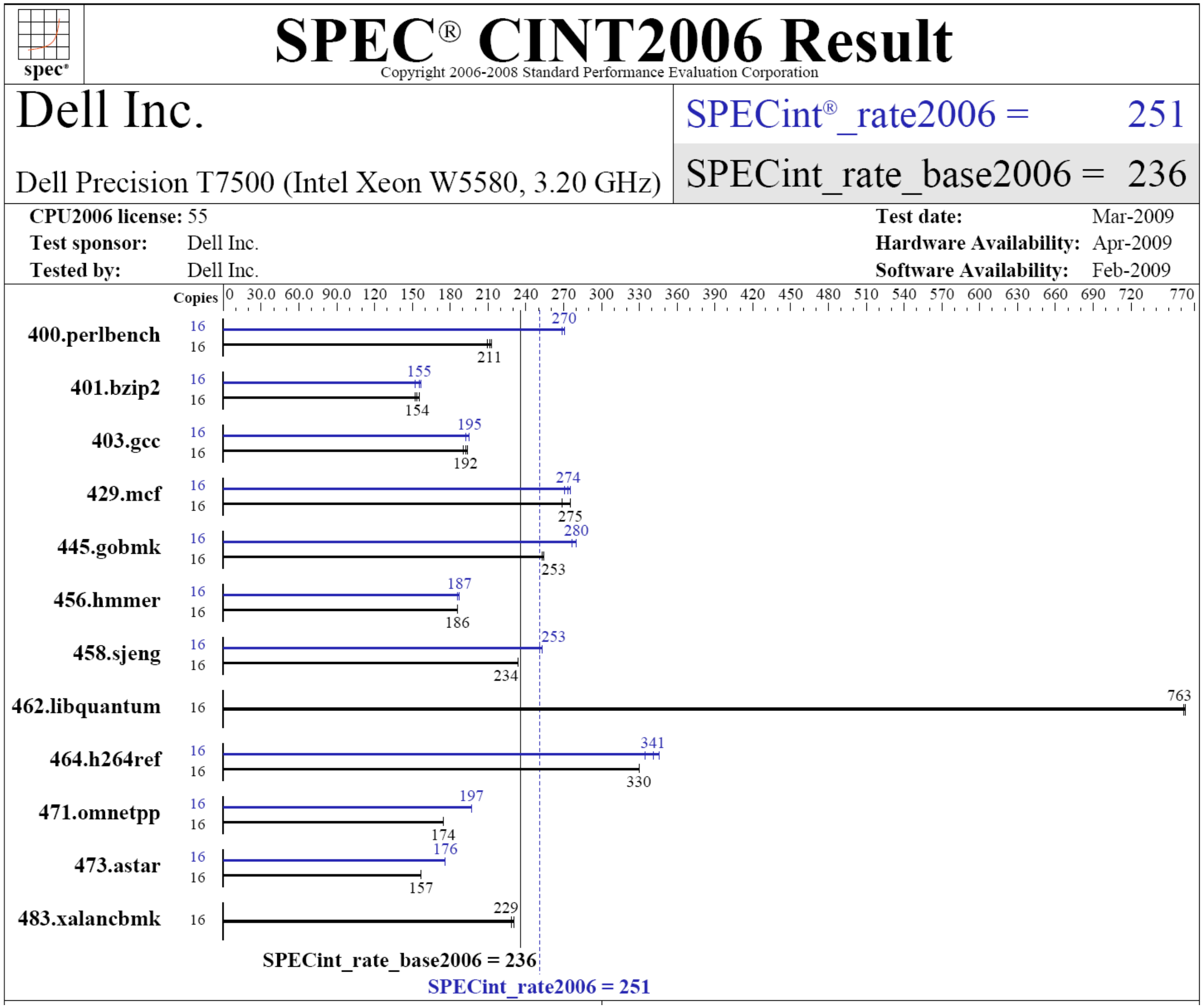
CPU2006 license: 55
Test sponsor: Dell Inc.
Tested by: Dell Inc.

Test date: Mar-2009
Hardware Availability: Apr-2009
Software Availability: Feb-2009

Results Table

	Base						Peak					
Benchmark	**Seconds**	**Ratio**	**Seconds**	**Ratio**	**Seconds**	**Ratio**	**Seconds**	**Ratio**	**Seconds**	**Ratio**	**Seconds**	**Ratio**
400.perlbench	406	24.1	**407**	**24.0**	407	24.0	291	33.5	**291**	**33.5**	291	33.5
401.bzip2	**522**	**18.5**	522	18.5	522	18.5	**502**	**19.2**	501	19.3	502	19.2
403.gcc	319	25.3	**312**	**25.8**	310	26.0	278	28.9	272	29.6	**273**	**29.5**
429.mcf	213	42.9	**189**	**48.2**	188	48.4	**207**	**44.0**	190	48.1	220	41.5
445.gobmk	438	23.9	**438**	**23.9**	438	24.0	385	27.3	382	27.4	**382**	**27.4**
456.hmmer	**388**	**24.1**	387	24.1	388	24.1	**384**	**24.3**	383	24.3	384	24.3
458.sjeng	487	24.9	475	25.5	**475**	**25.5**	**458**	**26.4**	456	26.6	462	26.2
462.libquantum	62.8	330	62.0	334	**62.6**	**331**	62.8	330	62.0	334	**62.6**	**331**
464.h264ref	592	37.4	**589**	**37.6**	589	37.6	**530**	**41.8**	529	41.8	531	41.7
471.omnetpp	272	23.0	304	20.6	**272**	**23.0**	**266**	**23.5**	247	25.4	333	18.8
473.astar	**387**	**18.2**	391	18.0	386	18.2	349	20.1	**349**	**20.1**	349	20.1
483.xalancbmk	**229**	**30.2**	230	30.0	228	30.2	**229**	**30.2**	230	30.0	228	30.2

SPEC Benchmark Update (2006)



SPEC Benchmark Update (2006)

SPEC CINT2006 Result

Dell Inc.

Dell Precision T7500 (Intel Xeon W5580, 3.20 GHz)

SPECint_rate2006 = 251

SPECint_rate_base2006 = 236

CPU2006 license: 55
Test sponsor: Dell Inc.
Tested by: Dell Inc.

Test date: Mar-2009
Hardware Availability: Apr-2009
Software Availability: Feb-2009

Results Table

Benchmark	Base							Peak						
	Copies	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Copies	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
400.perlbench	16	745	210	732	213	**743**	**211**	16	580	269	**579**	**270**	577	270
401.bzip2	16	**1006**	**154**	994	155	1012	152	16	**997**	**155**	986	157	1018	152
403.gcc	16	**669**	**192**	667	194	679	190	16	**662**	**195**	659	195	672	192
429.mcf	16	543	269	531	275	**532**	**275**	16	540	270	**533**	**274**	532	275
445.gobmk	16	661	254	665	253	**663**	**253**	16	598	280	**600**	**280**	606	277
456.hmmer	16	**804**	**186**	805	186	804	186	16	799	187	803	186	**799**	**187**
458.sjeng	16	828	234	**828**	**234**	828	234	16	768	251	765	253	**768**	**253**
462.libquantum	16	**435**	**763**	435	762	435	763	16	**435**	**763**	435	762	435	763
464.h264ref	16	**1074**	**330**	1073	330	1075	330	16	1024	346	**1039**	**341**	1061	334
471.omnetpp	16	573	174	572	174	**572**	**174**	16	509	197	510	197	**509**	**197**
473.astar	16	713	157	713	157	**713**	**157**	16	636	176	636	176	**636**	**176**
483.xalancbmk	16	482	229	**481**	**229**	481	230	16	482	229	**481**	**229**	481	230

Grand Example 2 – TPC Benchmark

- TPC (Transaction Processing Performance Council)
  - OLTP benchmark
  - Founded in August 1988 by Omri Serlin and 8 vendors
  - Currently four different benchmarks available
    - TPC-C: Order-entry benchmark
    - TPC-H, TPC-R: Decision support benchmark
    - TPC-W: Transactional web benchmark
  - For more info, see http://www.tpc.org

Grand Example 2 – TPC Benchmark

- TPC-C Benchmark
  - Databases consisting of a wide variety of tables
  - Concurrent transactions of five different types over the database
    - New-order: enter a new order from a customer
    - Payment: update customer balance to reflect a payment
    - Delivery: delivery orders (done as a batch transaction)
    - Order-status: retrieve status of customer's most recent order
    - Stock-level: monitor warehouse inventory
  - Transaction integrity (ACID properties)
  - Users and database scale linearly with throughput
  - Performance metric
    - Transaction rate: tpmC
    - Price per transaction: $/tpmC

Grand Example 2 – TPC Benchmark

TPC Transaction Processing Performance Council

Top Ten TPC-C by Performance
Version 5 Results As of 6-Apr-2009 4:49 AM [GMT]

Note 1: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

◉ All Results ◯ Clustered Results ◯ Non-Clustered Results Currency All

Rank	Company	System	tpmC	Price/tpmC	System Availability	Database	Operating System	TP Monitor
1	IBM	IBM Power 595 Server Model 9119-FHA	6,085,166	2.81 USD	12/10/08	IBM DB2 9.5	IBM AIX 5L V5.3	Microsoft COM+
***	Bull	Bull Escala PL6460R	6,085,166	2.81 USD	12/15/08	IBM DB2 9.5	IBM AIX 5L V5.3	Microsoft COM+
2	hp invent	HP Integrity Superdome-Itanium2/1.6GHz/24MB iL3	4,092,799	2.93 USD	08/06/07	Oracle Database 10g R2 Enterprise Edt w/Partitioning	HP-UX 11i v3	BEA Tuxedo 8.0
3	IBM	IBM System p5 595	4,033,378	2.97 USD	01/22/07	IBM DB2 9	IBM AIX 5L V5.3	Microsoft COM+
4	IBM	IBM eServer p5 595	3,210,540	5.07 USD	05/14/05	IBM DB2 UDB 8.2	IBM AIX 5L V5.3	Microsoft COM+
5	FUJITSU	PRIMEQUEST 580A 32p/64c	2,382,032	3.76 USD	12/04/08	Oracle Database 10g R2 Enterprise Edt w/Partitioning	Red Hat Enterprise Linux 4 AS	BEA Tuxedo 8.1
6	FUJITSU	PRIMEQUEST 580 32p/64c	2,196,268	4.70 USD	04/30/08	Oracle 10g Enterprise Ed R2 w/ Partitioning	Red Hat Enterprise Linux 4 AS	BEA Tuxedo 8.1
7	IBM	IBM System p 570	1,616,162	3.54 USD	11/21/07	IBM DB2 Enterprise 9	IBM AIX 5L V5.3	Microsoft COM+
***	Bull	Bull Escala PL1660R	1,616,162	3.54 USD	12/16/07	IBM DB2 9.1	IBM AIX 5L V5.3	Microsoft COM+
8	IBM	IBM eServer p5 595	1,601,784	5.05 USD	04/20/05	Oracle Database 10g Enterprise Edition	IBM AIX 5L V5.3	Microsoft COM+
9	FUJITSU	PRIMEQUEST 540A 16p/32c	1,354,086	3.25 USD	11/22/08	Oracle Database 10g release2 Enterprise Edt	Red Hat Enterprise Linux 4 AS	BEA Tuxedo 8.1
10	NEC	NEC Express5800/1320Xf (16p/32c)	1,245,516	4.57 USD	04/30/08	Oracle Database 10g R2 Enterprise Edt w/Partitioning	Red Hat Enterprise Linux 4 AS	BEA Tuxedo 8.1

Amdahl's Law

- Concept : The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.

- Execution time after improvement

$$= \left[\frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected} \right]$$

Amdahl's Law Example

- Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time. How much do I have to improve the speed of multiplication if I want my program to run five times faster

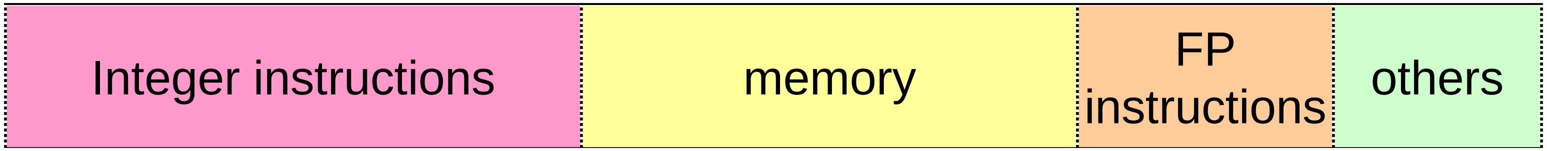
$$\left[\frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected} \right]$$

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80\text{seconds})$$

$$20 \text{ seconds} = \frac{80 \text{ seconds}}{n} + 20 \text{ seconds}$$

$$0 = \frac{80 \text{ seconds}}{n}$$

Amdahl's Law Example



After adding a pipelined integer instruction execution unit and cache memory (with FP emulation)

Summary

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- "Execution time is the only and unimpeachable measure of performance"
  - CPU time equation can predict performance by estimating the effects of changing features.
- Measuring performance requires good care
  - Good workloads (benchmarks)
  - Good ways to summarize performance
- Amdahl's Law
  - Speedup is limited by unimproved part of program