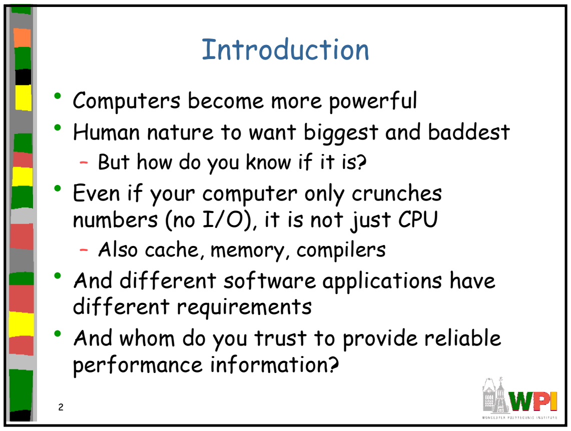
SPEC CPU2000
Measuring CPU Performance in the New Millennium

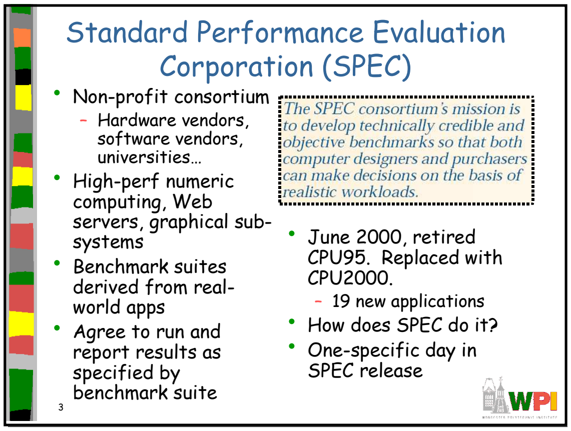
John. L. Henning
(Compaq)

IEEE Computer Magazine
2000



Introduction

- Computers become more powerful
- Human nature to want biggest and baddest
  - But how do you know if it is?
- Even if your computer only crunches numbers (no I/O), it is not just CPU
  - Also cache, memory, compilers
- And different software applications have different requirements
- And whom do you trust to provide reliable performance information?

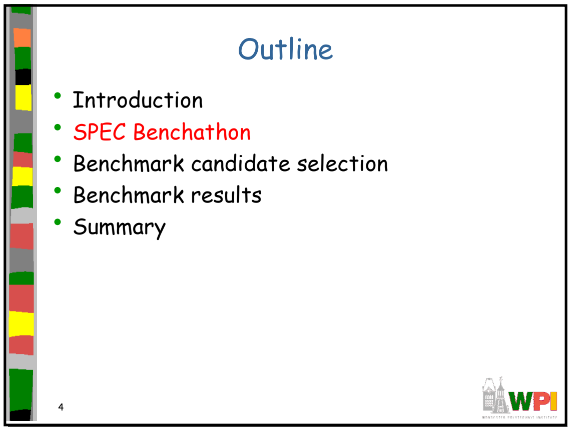


Standard Performance Evaluation Corporation (SPEC)

- Non-profit consortium
  - Hardware vendors, software vendors, universities…
- High-perf numeric computing, Web servers, graphical sub-systems
- Benchmark suites derived from real-world apps
- Agree to run and report results as specified by benchmark suite

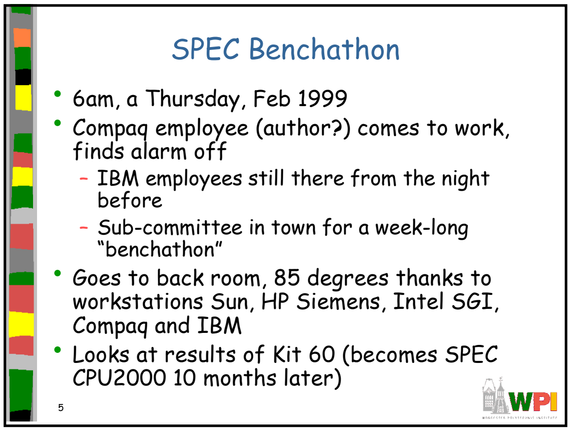
> *The SPEC consortium's mission is to develop technically credible and objective benchmarks so that both computer designers and purchasers can make decisions on the basis of realistic workloads.*

- June 2000, retired CPU95. Replaced with CPU2000.
  - 19 new applications
- How does SPEC do it?
- One-specific day in SPEC release



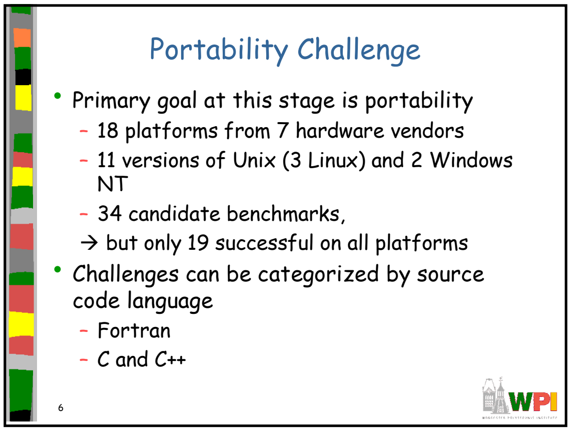
Outline

- Introduction
- SPEC Benchathon
- Benchmark candidate selection
- Benchmark results
- Summary



SPEC Benchathon

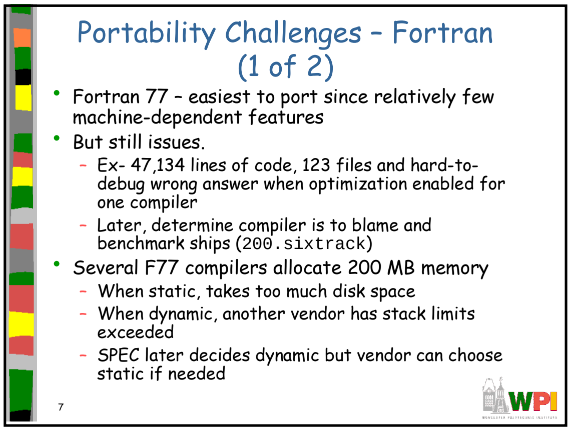
- 6am, a Thursday, Feb 1999
- Compaq employee (author?) comes to work, finds alarm off
  - IBM employees still there from the night before
  - Sub-committee in town for a week-long "benchathon"
- Goes to back room, 85 degrees thanks to workstations Sun, HP Siemens, Intel SGI, Compaq and IBM
- Looks at results of Kit 60 (becomes SPEC CPU2000 10 months later)



Portability Challenge

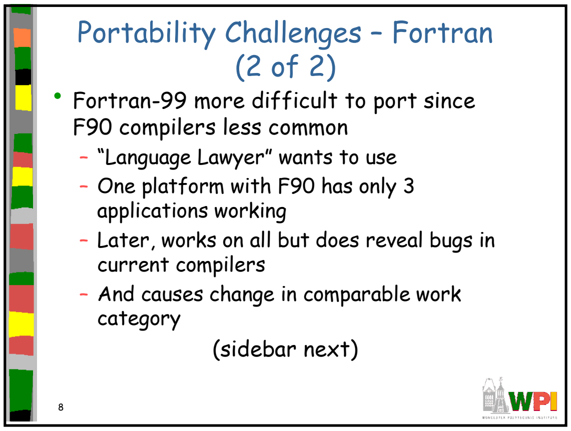
- Primary goal at this stage is portability
  - 18 platforms from 7 hardware vendors
  - 11 versions of Unix (3 Linux) and 2 Windows NT
  - 34 candidate benchmarks,

  → but only 19 successful on all platforms
- Challenges can be categorized by source code language
  - Fortran
  - C and C++



Portability Challenges – Fortran (1 of 2)

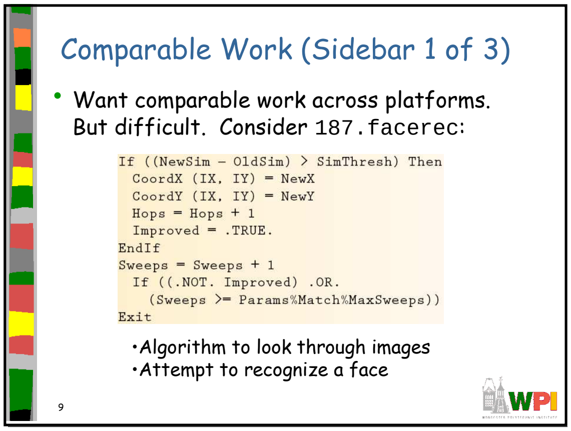
- Fortran 77 – easiest to port since relatively few machine-dependent features
- But still issues.
  - Ex- 47,134 lines of code, 123 files and hard-to-debug wrong answer when optimization enabled for one compiler
  - Later, determine compiler is to blame and benchmark ships (`200.sixtrack`)
- Several F77 compilers allocate 200 MB memory
  - When static, takes too much disk space
  - When dynamic, another vendor has stack limits exceeded
  - SPEC later decides dynamic but vendor can choose static if needed



Portability Challenges – Fortran (2 of 2)

- Fortran-99 more difficult to port since F90 compilers less common
  - "Language Lawyer" wants to use
  - One platform with F90 has only 3 applications working
  - Later, works on all but does reveal bugs in current compilers
  - And causes change in comparable work category

(sidebar next)

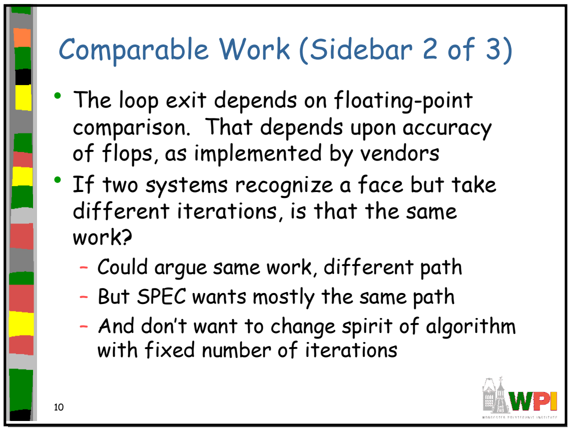


Comparable Work (Sidebar 1 of 3)

- Want comparable work across platforms. But difficult. Consider `187.facerec`:

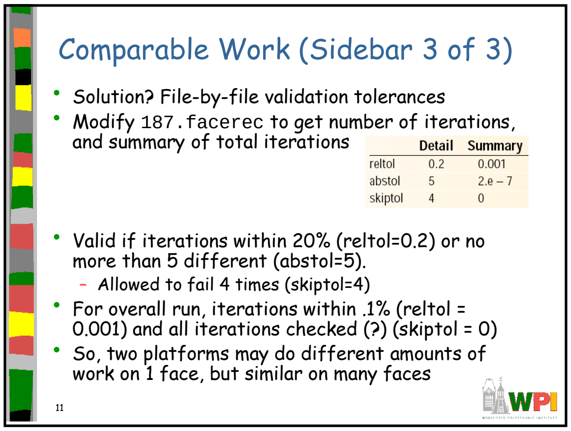
```
If ((NewSim - OldSim) > SimThresh) Then
  CoordX (IX, IY) = NewX
  CoordY (IX, IY) = NewY
  Hops = Hops + 1
  Improved = .TRUE.
EndIf
Sweeps = Sweeps + 1
  If ((.NOT. Improved) .OR.
    (Sweeps >= Params%Match%MaxSweeps))
Exit
```

•Algorithm to look through images
•Attempt to recognize a face



Comparable Work (Sidebar 2 of 3)

- The loop exit depends on floating-point comparison. That depends upon accuracy of flops, as implemented by vendors
- If two systems recognize a face but take different iterations, is that the same work?
  - Could argue same work, different path
  - But SPEC wants mostly the same path
  - And don't want to change spirit of algorithm with fixed number of iterations

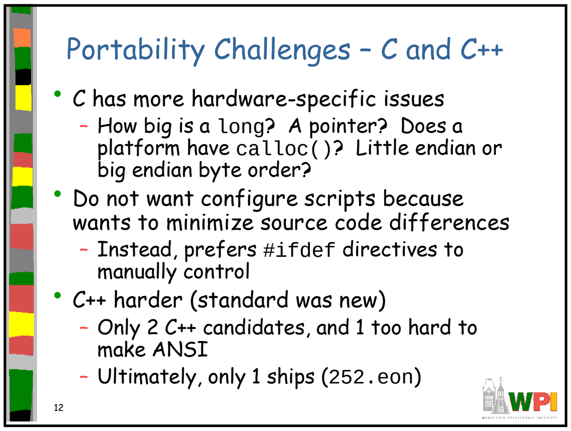


Comparable Work (Sidebar 3 of 3)

- Solution? File-by-file validation tolerances
- Modify `187.facerec` to get number of iterations, and summary of total iterations

	Detail	Summary
reltol	0.2	0.001
abstol	5	2.e – 7
skiptol	4	0

- Valid if iterations within 20% (reltol=0.2) or no more than 5 different (abstol=5).
  - Allowed to fail 4 times (skiptol=4)
- For overall run, iterations within .1% (reltol = 0.001) and all iterations checked (?) (skiptol = 0)
- So, two platforms may do different amounts of work on 1 face, but similar on many faces



Portability Challenges – C and C++

- C has more hardware-specific issues
  - How big is a `long`? A pointer? Does a platform have `calloc()`? Little endian or big endian byte order?
- Do not want configure scripts because wants to minimize source code differences
  - Instead, prefers `#ifdef` directives to manually control
- C++ harder (standard was new)
  - Only 2 C++ candidates, and 1 too hard to make ANSI
  - Ultimately, only 1 ships (`252.eon`)

February 1999 Benchathon Results

Table 1. February 1999 benchathon results.

	19 Feb	26 Feb
Compile errors	22	2
Runtime errors	18	6
Validation errors	60	41
Total	100	49

- Goal of benchathon is to have project leaders in place to resolve technical issues from multiple stakeholders
  - Employees from different companies, helping each other debug

Project Leader Structure

- Project leader shepherds candidate benchmarks
  - "Owns" resolution of portability problems
  - One has 10, but later lightens load
  - One has only 3, but difficult challenges
- Example 1: simulator gets different answers on different platforms
  - Later dropped
- Example 2: another requires 64 bit integers. Compilers for 32-bit platform can specify
- Example 3: app constructs color pixmap. Subtle differences in shades. Since not detectable by eye, deemed ok.

Outline

- Introduction
- SPEC Benchathon
- Benchmark candidate selection
- Benchmark results
- Summary

Benchmark Selection (1 of 3)

- Porting is clearly technical. Answer question "does benchmark work?"
- Selecting benchmarks harder
- Solicit candidates through search process on Web
- Members of SPEC vote. "Yes" if:
  - Many users
  - Exercises significant hardware resources
  - Solves interesting technical problem
  - Published results in journal
  - Or adds variety to suite

Benchmark Selection (2 of 3)

- "No" if:
  - Too hard to port
  - Does too much I/O so not CPU bound
  - Was previously in SPEC CPU suite
  - Code fragment rather than complete application
  - Is redundant
  - Appears to do different work on different platforms

Benchmark Selection (3 of 3)

Benchmark	Language	KLOC	Resident size (Mbytes)	Virtual size (Mbytes)	Description
SPECint2000					
164.gzip	C	7.6	181	200	Compression
175.vpr	C	13.6	50	55.2	FPGA circuit placement and routing
176.gcc	C	193.0	155	158	C programming language compiler
181.mcf	C	1.9	190	192	Combinatorial optimization
186.crafty	C	20.7	2.1	4.2	Game playing: Chess
197.parser	C	10.3	37	62.5	Word processing
252.eon	C++	34.2	0.7	3.3	Computer visualization
253.perlbmk	C	79.2	146	159	Perl programming language
254.gap	C	62.5	193	196	Group theory, interpreter
255.vortex	C	54.3	72	81	Object-oriented database
256.bzip2	C	3.9	185	200	Compression
300.twolf	C	19.2	1.9	4.1	Place and route simulator
SPECfp2000					
168.wupwise	F77	1.8	176	177	Physics: Quantum chromodynamics
171.swim	F77	0.4	191	192	Shallow water modeling
172.mgrid	F77	0.5	56	56.7	Multigrid solver: 3D potential field
173.applu	F77	7.9	181	191	Partial differential equations
177.mesa	C	81.8	9.5	24.7	3D graphics library
178.galgel	F90	14.1	63	155	Computational fluid dynamics
179.art	C	1.2	3.7	5.9	Image recognition/neural networks
183.equake	C	1.2	49	51.1	Seismic wave propagation simulation
187.facerec	F90	2.4	16	18.5	Image processing: Face recognition
188.ammp	C	12.9	26	30	Computational chemistry
189.lucas	F90	2.8	142	143	Number theory/primality testing
191.fma3d	F90	59.8	103	105	Finite-element crash simulation
200.sixtrack	F77	47.1	26	59.8	Nuclear physics accelerator design
301.apsi	F77	6.4	191	192	Meteorology: Pollutant distribution

Objective Criteria

- Want objective technical reasons for choosing/not choosing benchmarks
- But often at odds since technical reasons may be confidential
- Solution was all members provided some objective data and kept confidential
- Info: I/O, cache and main memory behavior, floating-point op mixes, branches, etc.

Subjective Criteria (1 of 3)

- Confidence in benchmark maintainability
  - Some have errors that are difficult to diagnose
  - Some have error fixed then re-appears
  - Some have easy to fix errors, but take sub-committee time
- All contribute to confidence level
- Needs to be manageable

Subjective Criteria (2 of 3)

- If stable quickly enough then can be analyzed
- Can be complex, but should not be misleading
- Workload should be describable in ordinary English and technical language

Subjective Criteria (3 of 3)

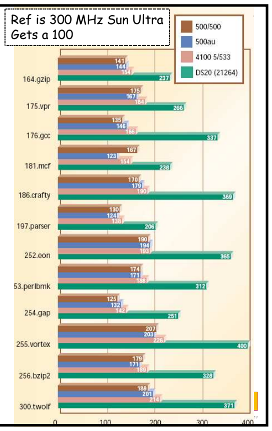
- Vendor interest matters. Temptation to vote accordingly. Two factors reduce influence
  - Generally, do not know numbers on competitors hardware. Hardware may not even be released. So, hard to vote for a benchmark because it is bad. Better to just vote on *merit*
  - Hard to argue the converse. I.e.- "you should vote for `999.favorite` because it helps my company".
- Of course, vendor interest represented. Want to keep level playing field

Outline

- Introduction
- SPEC Benchathon
- Benchmark candidate selection
- Benchmark results
- Summary

Results

- Want something we cannot learn by looking at clock speed
- 21164 differ by 92.3 to 331 (non-green)
- 500 MHz differ by more than 5% on 18 of 26 (clock diff is 7%)
- 533 MHz wins by 10% 3 times, < 3 % 3 times and loses 3 times
- Difference in memory hierarchy?

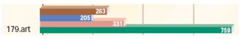

Ref is 300 MHz Sun Ultra
Gets a 100

Memory Hierarchy Differences

- Memory differences not clear
  - 500/500 has largest L3 cache
  - 500au has best memory latency
  - 4100 533 has highest main memory bwidth

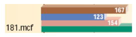
CPU		Alpha 21164	
System	500/500	500au	4100 5/533
CPU MHz	500	500	**533**
L1 cache on chip		8 Kbytes (instruction) + 8 Kbytes (data)	
L2 cache on chip		96 Kbytes	
Off-chip cache			
Size (Mbytes)	**8**	2	4
Latency (ns)	82	**58**	62
Latency (processor cycles)	41	**29**	33
Main memory			
Latency (ns)	341	**247**	248
Latency (processor cycles)	171	**124**	132
Bandwidth (Mbytes/s)	200	238	**272**

- 533 wins largest on `179.art`.

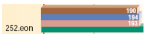


- Why? Perhaps because the benchmark fits in 4 Mbyte cache, but not 2 Mbyte cache

Effects of Cache



- 500/500 outperforms 533 most on `181.mcf`
- Looking at profile, can see benefits from larger cache
- `252.eon` is only place 500au wins. Very small, maybe in range of validation test. But could be from lower cache latency



Effects of Main Memory

- Cache matters, but many apps depend upon main memory
  - Where 533 is best

Benchmark	2 Mbytes	4 Mbytes	8 Mbytes
179.art	29.1	0.5	0.4
171.swim	23.9	23.7	23.6
183.equake	23.6	22.2	21.0
189.lucas	19.6	19.3	18.9
173.applu	14.2	14.0	13.0

- 2 of top 5 generators do not get better when same system has bigger cache
- 533 versus 500au has 14% better mem bandwidth. Provides 1%, 12%, 7% improvement for `171.swim`, `189.lucas`, `173.applu`

189.Lucas Analysis

- 21164 can have only two outstanding mem requests. Stalls after third
  - So code that spreads out memory requests will work better than if bunched
- 189.Lucas was hand-unrolled before submitting to SPEC and spread memory references
- So, overall, if you want good performance, benchmarks show not just CPU speed

Processor Performance (1 of 3)

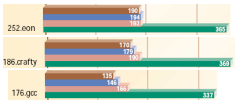
- DS20 has new chip 21264, but still 500 MHz
- Memory system different

CPU		Alpha 21164		Alpha 21264
System	500/500	500au	4100 5/533	DS20
CPU MHz	500	500	**533**	500
L1 cache on chip		8 Kbytes (instruction) + 8 Kbytes (data)		64 Kbytes (instruction) + 64
L2 cache on chip		96 Kbytes		None
Off-chip cache				
Size (Mbytes)	**8**	2	4	4
Latency (ns)	82	**58**	62	32
Latency (processor cycles)	41	**29**	33	16
Main memory				
Latency (ns)	341	**247**	248	184
Latency (processor cycles)	171	**124**	132	92
Bandwidth (Mbytes/s)	200	238	**272**	1,232

- Cache latency by factor 1.8x, memory latency by 1.3x,
- Bwdith latency by 4.5x

Processor Performance (2 of 3)

- Three integer benchmarks biggest



1.89x

1.94x

2.03x

- `176.gcc` greatest because of workload
  - Was only 1.61x better for CPU95
- CPU95 gcc ran for 79 seconds, 47 MB vm
- CPU2000 gcc runs 327 seconds and uses 156 MB vm

Processor Performance (3 of 3)

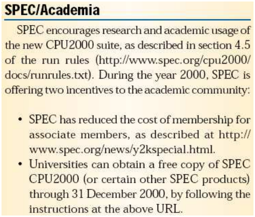
- `171.swim`, `189.lucas`, `173.applu` (earlier table) should benefit from 4.5x mem bandwidth
  - 4.9x, 2.2x, and 2.8x respectively
- `183.equake` improves by only 1.7x despite high miss rate. Analysis shows because program is bound by latency not memory bwidth.

Compiler Effects

- All results in article use single compiler and "base" tuning
  - No more than 4 switches and same switches for all benchmarks in a suite
- Different tuning would have different results
- Highlights:
  - 400,000 lines of new float code with '-fast' flag make it tougher to be robust
  - Unrolling can really help. Ex: `178.galgel` unrolled had 70% improvement versus base tuning
- Note, recommend continued compiler improvements but should improve general applications and not just SPEC benchmarks

Summary

- SPEC encourages industry and academia to study more
- Now is the time for CPU 200x (CPU 2004)
- (Have ordered CPU2000 for those that want a go)



SPEC/Academia

SPEC encourages research and academic usage of the new CPU2000 suite, as described in section 4.5 of the run rules (http://www.spec.org/cpu2000/docs/runrules.txt). During the year 2000, SPEC is offering two incentives to the academic community:

- SPEC has reduced the cost of membership for associate members, as described at http://www.spec.org/news/y2kspecial.html.
- Universities can obtain a free copy of SPEC CPU2000 (or certain other SPEC products) through 31 December 2000, by following the instructions at the above URL.