CS 347
Benchmarking

Todd C. Mowry

March 5, 1998

Topics

- Peak measures
- Benchmark measures

Characterizing Computer Performance

Computer buyers want a single number that predicts performance of real applications.

Computer makers have resisted measures that would allow meaninful direct comparisons.

- **lack of operating system and language standards**
- **difficult to develop portable and realistic applications**

1970's and 1980's:

- **era of meaningless rates**

1980's:

- **age of meaningless benchmarks**

1990's:

- **dawn of semi-realistic benchmarks**

Meaningless rate #1: MHz

MHz = millions of clock cycles/sec

MHz doesn't predict running time:

- **T secs = I inst x (C cycles/I inst) x 1/(MHz x 10^6) cycles/sec**

CPU	MHz	System	SPECfp95 time (secs)
Pentium Pro	180	Alder	6,440
POWER2	77	RS/6000 591	3,263

Meaningless rate #2: peak MIPS

MIPS = millions of instructions / second

Peak MIPS = MIPS for some optimal instuction stream.

Peak MIPS doesn't predict running time:

- **number of instructions executed don't predict running time.**
- **optimal instruction stream can be meaningless**

Example:

If the instruction stream is a sequence of NOPS, then a 100 MHz Pentium is a 3,200 MIPS machine!

Instruction decoder looks at 32 bytes at a time. NOP is a one-byte instruction. Decoder discards NOP's.

Meaningless rate #3: peak MFLOPS

MFLOPS = millions of floating operations /sec

peak MFLOPS = MFLOPS for some optimal instruction stream.

MFLOPS doesn't predict execution time:

- **floating point operations do not predict running time**
- **even if the did, the ideal instruction stream is usually unrealistic**

Measured MFLOPS on Intel i860 (peak MFLOPS = 80):

Program	1d fft	sasum	saxpy	sdot	sgemm	sgemv	spvma
MFLOPS	8.5	3.2	6.1	10.3	6.2	15.0	8.1
%peak	11%	4%	7%	13%	8%	19%	10%

Benchmarking

Goal: Measure a set of programs (benchmarks) that represent the workload of real applications and that predict the running time of those applications.

Steps in the benchmarking process:

(1) Choose representative benchmark programs.

- difficult to find realistic AND portable programs.

(2) Choose an individual performance measure (for each benchmark)

- time, normalized time, rate?

(3) Choose an aggregate performance measure (for all benchmarks)

- sum, normalized sum, mean, normalized mean?

Why Do Benchmarking?

How we evaluate differences

- **Different systems and changes to single system**

Provide a target for system developers

- **Benchmarks should represent large class of important programs**
- **Improving benchmark performance should help many programs**

For better or worse, benchmarks shape a field

- **Good ones accelerate progress**
  - good target for development
- **Bad benchmarks hurt progress**
  - help real programs v. sell machines/papers?
  - Inventions that help real programs don't help benchmark

"Ounce of honest data is worth more than a pound of marketing hype."

Benchmark examples

(Toy) Benchmarks

- 10-100 line
- e.g.,: sieve, puzzle, quicksort

Synthetic Benchmarks

- attempt to match average frequencies of real workloads
- e.g., Whetstone, Dhrystone

Kernels

- Time critical excerpts of real programs
- e.g., Livermore loops, fast Fourier transform

Real programs

- e.g., gcc, jpeg

Successful Benchmark Suite: SPEC

www.specbench.org/osg/

1987: RISC industry mired in "bench marketing":

- **"That is 8 MIPS machine, but they claim 10 MIPS!"**

1988 : EE Times + 5 companies band together to perform Systems Performance Evaluation Committee (SPEC) in 1988

- **Sun, MIPS, HP, Apollo, DEC**

Create standard list of programs, inputs, reporting:

- **some real programs, includes OS calls, some I/O**

Currently SPEC is more than 40 computer companies:

- **Compaq, Cray, DEC, HP, Hitachi, IBM, Intel, Motorola, Netscape, SGI, Sun**

SPEC Benchmarks

New incarnations required every three years:

- **SPEC89, SPEC92, SPEC95.**

Causes of benchmark obsolescence:

- **increasing processor speed**
- **increasing cache sizes**
- **increasing levels of caches**
- **increasing application code size**
- **library code dependences**
- **aggressive benchmark engineering**

SPEC95 integer benchmarks

benchmark	*description*
go	**plays a game of go**
m88ksim	**Motorola 88k chip simulator**
gcc	**Gnu C compiler**
compress	**in-memory LZW file compression**
li	**Lisp interpreter**
jpeg	**spectral based image compression/decompression**
perl	**Perl program that manipulates strings and primes**
vortex	**database program**

SPEC95 floating point benchmarks

benchmark	*description*
tomcatv	**mesh generation program**
swim	**513x513 shallow water finite difference model**
su2cor	**Monte Carlo simulation**
hydro2d	**2D Navier-Stokes solver**
mgrid	**3D multigrid solver**
applu	**parabolic/elliptic PDE solver**
turb3d	**turbulence model**
apu	**air pollution model**
fppp	**quantum chemistry model**
wave5	**electromagnetic particle model**

SPEC CPU performance measures

SPECfp = $(NT_1 \times NT_2 \times ... \times NT_n)^{1/n}$

Each NT_k is a normalized time:

- **(reference time for benchmark k) / (measured time for benchmark k)**
- **reference times are measured on a Sparcstation 10/40 (40 MHz Supersparc with no L2 cache)**

Problem: SPEC performance measures don't predict execution time!!!

system	total time	SPECfp95
166 MHz Pentium Pro	6470	5.47
180 MHz Pentium Pro	6440	5.40

Means and Ratios

	frames	sys A	sys B	sys C
prog 1	320	20 secs	10 secs	40 secs
prog 2	320	40 secs	80 secs	20 secs
total		60 secs	90 secs	60 secs

Total running time is the ultimate performance measure.

Means and Ratios (Cont.)

		seconds		
	frames	sys A	sys B	sys C
prog 1	320	20	10	40
prog 2	320	40	80	20
total		60	90	60
normalized to A		1	1.5	1
normalized to B		.67	1	.67
normalized to C		1	1.5	1

Normalized total running time is OK too. It tracks with total running time.

Means and Ratios (Cont.)

Arithmetic mean (AM) = $(T_1 + T_2 + ... + T_n) / n$

	frames	seconds sys A	sys B	sys C
prog 1	320	20	10	40
prog 2	320	40	80	20
total		60	90	60
AM		30	45	30
normalized to A		1	1.5	1
normalized to B		.67	1	.67
normalized to C		1	1.5	1

Normalized and unormalized arithmetic means predict running time.

Means and Ratios (Cont.)

		seconds (normalized to A)(normalized to B)		
	frames	sys A	sys B	sys C
prog 1	320	20 (1.0) (2.0)	10 (0.5) (1.0)	40 (2.0) (4.0)
prog 2	320	40 (1.0) (0.5)	80 (2.0) (1.0)	20 (0.5) (0.25)
total		60 (2.0) (2.5)	90 (2.5) (2.0)	60 (2.5) (4.25)
AM		30 (1.0) (1.25)	45 (1.25) (1.0)	30 (1.25) (2.13)

Sums of normalized times and arithmetic means of normalized times do NOT predict running time!!!

Means and Ratios (Cont.)

Geometric mean (GM) = $(T_1 \times T_2 \times \ldots \times T_n)^{1/n}$

		seconds (normalized to A)(normalized to B)		
	frames	**sys A**	**sys B**	**sys C**
prog 1	**320**	**20 (1.0) (2.0)**	**10 (0.5) (1.0)**	**40 (2.0) (4.0)**
prog 2	**320**	**40 (1.0) (0.5)**	**80 (2.0) (1.0)**	**20 (0.5) (0.25)**
total		**60 (2.0) (2.5)**	**90 (2.5) (2.0)**	**60 (2.5) (4.25)**
GM		**28.3 (1) (1)**	**28.3 (1) (1)**	**28.3 (1) (1)**

The geometric means are consistent (i.e. independent of the system they are normalized to), but they are consistently wrong!!!

This is why the SPECfp95 numbers don't always predict running time.

Means and Ratios (Cont.)

The harmonic mean (HM) is a measure for rates (and ratios in general) that predicts running time:

Suppose rate for each program k is W_k/T_k, where

W_k = work for program k

T_k = running time for program k.

Then

$HM = sum_{k=1..n}(W_k) / sum_{k=1..n}(T_k)$

Means and Ratios (Cont.)

		frames/sec		
	frames	sys A	sys B	sys C
prog 1	320	16	32	8
prog 2	320	8	4	16
total frames/sec		24	36	24
AM		12	18	12
GM		11.3	11.3	11.3
HM		10.7	7.1	10.7
(total time)		(60)	(90)	(60)

HM is the only measure for rates (and ratios in general) that predicts running time.

Alternate formulation of HM

If $W_k = W_j = W$, for all k and j, and $R_k = W / T_k$

Then

$HM = n / sum_{k=1...n}(1/R_k)$

Summary

1. Total running time is the true performance measure.
2. A performance metric should track total running time.
3. AM can be used to summarize performance expressed as an unnormalized time.
4. AM should NOT be used to summarize performance expressed as a ratio (i.e. a rate or normalized time)
5. GM should NOT be used for summarizing performance expressed as a time or a rate.
6. HM should be used for summarizing any performance expressed as a ratio.
7. If you want to normalize, compute the aggregate measure first, then normalize.