Performance of Computers

Which computer is fastest?

Not so simple

- scientific simulation - FP performance
- program development - Integer performance
- commercial work - I/O

Performance of Computers

Want to buy the fastest computer for what you want to do

- workload is important

Want to design the fastest computer for what they want to pay

- BUT cost is an important criterion

Forecast

Time and performance

Iron law

MIPS and MFLOPS

Which programs and how to average

Amdahl's law

Defining Performance

What is important to who

Computer system user

- minimize elapsed time for program = time_end - time_start
- called response time

Computer center manager

- maximize completion rate = #jobs/second
- called throughput

Response Time vs. Throughput

Is throughput = 1/av. response time?

- only if NO overlap
- with overlap, throughput > 1/av.response time
- e.g., a lunch buffet - assume 5 entrees
- each person takes 2 minutes at every entree
- throughput is 1 person every 2 minutes
- BUT time to fill up tray is 10 minutes
- why and what would the throughput be, otherwise?
  because there are 5 people (each at 1 entree) simultaneously; if there is no such overlap throughput = 1/10

© 2000 by Mark D. Hill

What is Performance for us?

For computer architects

- CPU execution time = time spent running a program

Because people like faster to be bigger to match intuition

- performance = 1/X time
- where X = response, CPU execution, etc.

Elapsed time = CPU execution time + I/O wait

We will concentrate mostly on CPU execution time

© 2000 by Mark D. Hill

Improve Performance

Improve (a) response time or (b) throughput?

- faster CPU
  - both (a) and (b)
- Add more CPUs
  - (b) but (a) may be improved due to less queueing

© 2000 by Mark D. Hill

Performance Comparison

Machine A is n times faster than machine B iff

- perf(A)/perf(B) = time(B)/time(A) = n

Machine A is x% faster than machine B iff

- perf(A)/perf(B) = time(B)/time(A) = 1 + x/100

E.g., A 10s, B 15s

- 15/10 = 1.5 => A is 1.5 times faster than B
- 15/10 = 1 + 50/100 => A is 50% faster than B

© 2000 by Mark D. Hill

Breaking down Performance

A program is broken into instructions

- H/W is aware of instructions, not programs

At lower level, H/W breaks intructions into cycles

- lower level state machines change state every cycle

E.g., 500 MHz PentiumIII runs 500M cycles/sec, 1 cycle = 2 ns

E.g., 2 GHz PentiumX will run 2G cycles/sec, 1 cycle = 0.5 ns

Iron law

Time/program = instrs/program x cycles/instr x sec/cycle

sec/cycle (a.k.a. cycle time, clock time) - 'heartbeat' of computer

- mostly determined by technology and CPU organization

cycles/instr (a.k.a. CPI)

- mostly determined by ISA and CPU organization
- overlap among instructions makes this smaller

instr/program (a.k.a. instruction count)

- instrs executed NOT static code
- mostly determined by program, compiler, ISA

Our Goal

Minimize time which is the product, NOT isolated terms

Common error to miss terms while devising optimizations

- E.g., ISA change to decrease instruction count
- BUT leads to CPU organization which makes clock slower

Other Metrics

MIPS and MFLOPS

MIPS = instruction count/(execution time x 10^6)

= clock rate/(CPI x 10^6)

BUT MIPS has problems

Problems with MIPS

E.g., without FP H/W, an FP op may take 50 single-cycle instrs

with FP H/W only one 2-cycle instr

Thus adding FP H/W

- CPI increases (why?) The FP op goes from 50/50 to 2/1
    - but instrs/prog decreases more (why?) each of the FP op reduces from 50 to 1, factor of 50
    - total execution time decreases
- For MIPS
    - instrs/prog ignored
    - MIPS gets worse!

Problems with MIPS

Ignore program

Usually used to quote peak performance

- ideal conditions => guarantee not to exceed!!

When is MIPS ok?

- same compiler and same ISA
- e.g., same binary running on Pentium Pro and Pentium
- why? instrs/prog is constant and may be ignored

Other Metrics

MFLOPS = FP ops in program/(execution time x 10^6)

Assuming FP ops independent of compiler and ISA

- Assumption not true
    - may not have divide in ISA
    - optimizing compilers

Relative MIPS and normalized MFLOPS

- adds to confusion! (see book)

Rules

Use ONLY Time

Beware when reading, especially if details are omitted

Beware of Peak

Iron Law Example

Machine A: clock 1 ns, CPI 2.0, for a program

Machine B: clock 2 ns, CPI 1.2, for same program

Which is faster and how much

Time/program = instrs/program x cycles/instr x sec/cycle

Time(A): N x 2.0 x 1 = 2N

Time(B): N x 1.2 x 2 = 2.4N

Compare: Time(B)/Time(A) = 2.4N/2N = 1.2

So, Machine A is 20% faster than Machine B for this program

Iron Law Example

Keep clock of A at 1 ns and clock of B at 2 ns

For equal performance, if CPI of B is 1.2, what is CPI of A?

Time(B)/Time(A) = 1 = (N x 2 x 1.2)/(N x 1 x CPI(A))

CPI(A) = 2.4

Iron Law Example

Keep CPI of A 2.0 and CPI of B 1.2

For equal performance, if clock of B is 2 ns, what is clock of A?

Time(B)/Time(A) = 1 = (N x 2.0 x clock(A))/(N x 1.2 x 2)

clock(A) = 1.2 ns

Which Programs

Execution time of what

Best case - you run the same set of programs everyday

- port them and time the whole “workload”

In reality, use benchmarks

- programs chosen to measure performance
- predict performance of actual workload (hopefully)

+ saves effort and money

– representative? honest?

How to average

Example (page 70)

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100
Total	1001	110

One answer: total execution time, then B is how much faster than A? 9.1

© 2000 by Mark D. Hill

How to average

Another: arithmetic mean (same result)

Arithmetic mean of times: $\left\{\sum_{1}^{n} time(i)\right\} / n$ for n programs

AM(A) = 1001/2 = 500.5

AM(B) = 110/2 = 55

500.5/55 = 9.1

Valid only if programs run equally often, so use “weight” factors

Weighted arithmetic mean: $\left\{\sum_{1}^{n} (weight(i) \times time(i))\right\} / n$

© 2000 by Mark D. Hill

Other Averages

E.g., 30 mph for first 10 miles

90 mph for next 10 miles. averatge speed?

Average speed = (30+90)/2  WRONG

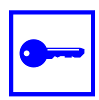
 Average speed = total distance / total time

- (20 / (10/30+10/90))
- 45 mph

© 2000 by Mark D. Hill

Harmonic Mean

Harmonic mean of rates = $\dfrac{1}{\left\{\sum_{1}^{n} \dfrac{1}{rate(i)}\right\} / n}$

 Use HM if forced to start and end with rates

Trick to do arithmetic mean of times but using rates and not times

© 2000 by Mark D. Hill

Dealing with Ratios

E.g.,

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100

If we take ratios, with respect to Machine A

	Machine A	Machine B
Program 1	1	10
Program 2	1	0.1

Dealing with Ratios

average for machine A is 1, average for machine B is 5.05

If we take ratios, with respect to Machine B

	Machine A	Machine B
Program 1	0.1	1
Program 2	10	1

average for machine A = 5.05, average for machine B = 1

can't both be true!

Don't use arithmetic mean on ratios (normalized numbers)

Geometric Mean

Use geometric mean for ratios

geometric mean of ratios = $\sqrt[n]{\prod_{1}^{n} ratio(i)}$

Use GM if forced to use ratios

Independent of reference machine (math property)

In the example, GM for machine A is 1, for machine B is also 1

- normalized with respect to either machine

But..

Geometric mean of ratios is not proportional to total time

AM in example says machine B is 9.1 times faster

GM says they are equal

If we took total execution time, A and B are equal only if

- program 1 is run 100 times more often than program 2

Generally, GM will mispredict for three or more machines

Summary

Use AM for times

Use HM if forced to use rates

Use GM if forced to use ratios

Better yet, use unnormalized numbers to compute time

Benchmarks: SPEC95

System Performance Evaluation Cooperative

Latest is SPEC2K but Text uses SPEC95

8 integer and 10 floating point programs

- normalize run time with a SPARCstation 10/40
- GM of the normalized times

SPEC95

Benchmark	Description
go	AI, plays go
m88ksim	Motorola 88K chip simulator
gcc	Gnu compiler
compress	Unix utility compresses files
li	Lisp Interpreter
ijpeg	Graphic (de)compression
perl	Unix utility text processor
vortex	Database program

Some SPEC95 Programs

Benchmark	INT/FP	Description
m88ksim	Integer	Motorola 88K chip simulator
gcc	Integer	Gnu compiler
compress	Integer	Unix utility compresses files
vortex	Integer	Database program
su2cor	FP	Quantum physics; Monte carlo
hydro2d	FP	Navier Stokes equations
mgrid	FP	3-D potential field
wave5	FP	Electromagnetic particle simulation

Amdahl's Law

Why does the common case matter the most?

Speedup = old time/new time = new rate/old rate

Let an optimization speed f fraction of time by a factor of s

Spdup = $[(1-f)+f] \times oldtime / ([(1-f) \times oldtime] + f/s \times oldtime)$

$= 1/(1-f+f/s)$

Amdahl's Law Example

Your boss asks you to improve Pentium Posterior performance by

- improve the ALU used 95% of time, by 10%
- improve the memory pipeline used 5%, by a factor of 10

Let f = fraction sped up and s = the speedup on that fraction

- new_time = (1-f)*old_time + (f/s)*old_time
- Speedup = new_rate / old_rate = old_time / new_time
- Speedup = old_time / ((1-f)*old_time + (f/s)*old_time)

Amdahl's Law: Speedup = 1 / ((1-f) + (f/s))

Amdahl's Law Example, cont.

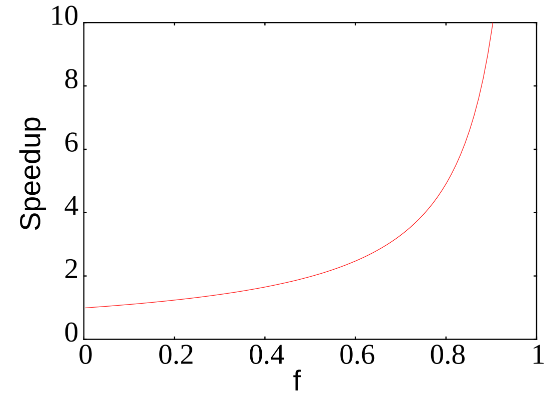
Your boss asks you to improve Pentium Posterior performance by

- improve the ALU used 95% of time, by 10%
- improve the memory pipeline used 5%, by a factor of 10

f	s	Speedup
95%	1.10	1.094
5%	10	1.047
5%	∞	1.052

Amdahl's Law: Limit

$\lim_{s \to \infty} \left(\frac{1}{1-f+f/s} \right) = \frac{1}{1-f}$ => Make common case fast



Summary of Chapter 2

Time and performance: Machine A n times faster than Machine B

- iff Time(B)/Time(A) = n

Iron Law: Time/prog = Instr count x CPI x Cycle time

Other Metrics: MIPS and MFLOPS

- Beware of peak and omitted details

Benchmarks: SPEC95

Summarize performance: AM for time, HM for rate, GM for ratio

Amdahl's Law: Speedup = $1/(1-f+f/s)$ - common case fast