Lecture #3

****
Review -- 1 min
****
Unifying idea: quantitative basis for architecture

Engineering Methodology
    tech trends, quantitative approach
Bottom line: performance
    throughput or latency
    execution time is the real measure of computer performance
benchmarking
    o games people play, mistakes people make

    o good benchmarks --> good products

    o bad benchmarks create dilemma -- improve
    real performance or improve benchmark number.
    Marketing almost always wins.

****
Outline - 1 min
****
summarizing performance
Amdahl's law
CPU performance

<project discussion>

cost
****
Lecture - 20 min
****

How to summarize performance

Mean -- how to summarize several numbers

Arithmetic/Harmonic -- track total execution time

arithmetic: 1/n * sum(time_1, time_2, ... time_n)

use harmonic for rates

harmonic:

$$\frac{n}{\text{sum}(1/\text{rate\_1}, 1/\text{rate\_2}, \ldots 1/\text{rate\_n})}$$

example: suppose you send 10MB @ 1MB/s
then 10MB @ 5 MB/s
What is avg rate?
XXX: 1 + 5 / 2 = 3 MB/s WRONG

correct: first transfer took 10 seconds, second took
2 seconds --> total time 12 seconds for 20 MB
--> avg rate 1.7 MB/s

(also weighted arithmetic, weighted harmonic)

Geometric
nth root of product of n samples

Arithmetic v. Geometric
Arithmetic: tracks time

Geometric: doesn't matter what machine you normalize to

Problem with geometric mean: encourages spending time to improve
simplest programs v. improving programs where time is spent

e.g. 2 seconds --> 1 second gives same impact as 10000
seconds --> 5000 seconds
(and small programs easier to "crack")

Example: SPEC89 on IBM 550 (slide)

	Ratio to VAX			Time		Weighted Time		
Program		Befo	After	Befr	After	Before	After	
gcc		30	29	49	51	8.91	9.22	
espreso		35	34	65	67	7.64	7.86	
spice		47	47	510	510	5.69	5.69	
doduc	46	49	41	38	5.81	5.45		
nasa7		78	144	258	140	3.43	1.86	
li		34	34	183	183	7.86	7.86	
eqntott		40	40	28	28	6.68	6.68	
matrix300		78	730	58	6	3.43	0.37	<---!!!
fpppp		90	87	34	35	2.97	3.07	
tomcatv		33	138	20	19	2.01	1.94	
man		54	72	124	108	54.42	49.99	
		Geometric		Aritmetic		Weighted Arithmetic		
		Ratio 1.33		1.16		1.09		

Story -- matrix300 spent >90% of its time on one line
cracked it -- got 10x improvement

moral -- geometric gives incentive to do unrealistic optimizations that crack benchmark

Amdahl's Law -- law of diminishing returns

Willie Sutton "Why do I rob banks? Cause that's where the money is."
You should do research same way!
(Depressing how many people don't!)

$$\text{speedup(enhancement)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{performance with}}{\text{performance without}}$$

suppose enhancement speeds up fraction F of program and leaves rest unchanges?

ExTime new = ExTime old * (1-Fraction enhanced + FractionEnhaced/SpeedupEnhanced)

$$\text{Speedup overall} = \frac{1}{\text{1-fraction enhanced} + \frac{\text{fraction enhanced}}{\text{speedup enhanced}}}$$

Question: suppose program spends 25% of its time doing floating point
What is MAX speedup I can get by improving floating point?
A: 1/0.75 = 1.33

(Remember definition of “faster than” or “speedup”

CPU Performance

Remember TIME is a measure of performance
What you care about is how long does it take to run my program?

```
**************************************
***           Wake Up!             ***
*** Most useful equation in chapters 3 and 4 ***
**************************************
```

CPU time = # instructions * cycles per instructin * clock cycle time

[time] = [instructions] * [cycles]/[instructions] * [time]/[cycle]

problem with "MIPS" and "MHz" as peformance metrics

Beware techniques that talk about improvments in one or 2 of the three only
e.g. optimizing compiler reduces number of instructions
increases cycles per instruction

Question: why?

****

Admin - 3 min

****

does eveyone have homework partner?
FRI – HW1
FRI – proj interests

project discussion

Goal:

1) learn to do systems research

Good science = skills + attitude

work independently -- from topic selection to execution
(diferent from undergrad projects)

quantitative methodology
(don't just measure a bunch of stuff, test hypothesis
--> better results with less effort

2) Learn about an area of architecture in depth
3) Advance state of the art

approach

1. select a topic
2. state hypothesis

3. design experiment to test hypothesis
    - ask yourself: what graph would I draw at the end of the experiment to convince you that I'm right?
    - ask yourself: suppose I'm wrong, will experiment tell me?

4. execute plan

Checkpoints for class

1) Jan 28 – topic interests – mail to jingzhao@cs
- your name, email
- project interests
- whether you have a partner or are looking for one

We'll post the list – you can use it to find people that might be good project partner

2) Feb 6 – topic selection
- partner
- topic:
    - hypothesis you plan to test
    - general approach

Selecting a topic

- General approach -- different ways to go
    - reproduce the results of an important paper (other fields do this a lot)
        - ♦ harder than you think -- papers omit important details
        - ♦ this type of project will be graded very much on stringent scientific procedures

    - take a paper and extend its ideas just abit
    - a new idea

- Topic ideas – see handout for a long list. Highlight a few here.
- Factors to consider
    - time limits -- make progress in 2 months

    (important to have at least some part of it completely working hard for e to evaluate a simulator that didn't quite work)

- interest -- work on something you care about

tie it into your research interests if possible (bring special insight and expertise that others may not have)

- impact -- work on something others will care about

goal is something that, if successful, would make a good conference
    paper

3) Feb 27 – proposal
- literature review/critique
- detailed plan of attack

4) April 10 – checkpoint
- initial results

    Show me 1 graph!!! I will be impressed with anyone that has some initial results, no matter how minor.

5) May 4-8 – in-class presentations
6) May 11 – written report

****
Lecture - 24 min
****

Costs
----

cost/performance is driving force in CS today

others in department might look down on considering costs
    "transient engineering factors" not "scientific principles"

    wrong for 3 reasons
    1. cost will always be part of design

    (correct trade-offs may change from year-to-year, but your approach to making those trade-offs will last)

2. cost/performance factors change at exponential rates
--> exert qualitative,lasting impact on design
just as algorithm design would't ignore an exponential factor,
neither should computer design

3. some cost/performance rules are close to laws
e.g. "smaller is faster," "smaller is cheaper"

IC Cost
----

slide
Details in book
I'm assuming that most people have seen math in UG class

Not going to cover in detail, but you may want to spend some time working through an example problem (this is stuff you should know how to do)

$$\text{IC cost} = \frac{\text{die cost} + \text{testing cost} + \text{packaging cost}}{\text{final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{dies per wafer} * \text{die yield}}$$

$$\text{Dies per wafer} = \frac{\text{pi} * (\text{wafer\_diam}/2)\text{\^{}}2}{\text{die area}} - \frac{\text{pi} * \text{wafer\_diam}}{\text{sqrt}(2*\text{die area})} - \text{Test dies}$$

/* area/area - correction factor - test dies */

Die yield = WaferYield*{1+(defects per unit area * die area) / alpha} ^(-alpha)

- empirical formula
- alpha depends on complexity of process (more layers --> larger alpha)
- also learning curve --> alpha falls over time for a chip
- larger chip --> **double whammy**: fewer dies per wafer, more defects per die

SLIDE: Real World Examples (Aug 1993)

Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	mm^2	Area wafer	Dies/	Yield Cost	Die
386DX		2	.9	900	1.0	43	360	71%	$4
486DX2		3	.8	1200	1	81	181	54%	$12
PowerPC 601		4	.8	1700	1.3	121	115	28%	$53
HP PA 71003		.8	1300	1	196	66	27%	$73	
DEC Alpha 3		.7	1500	1.2	234	53	19%	$149	
SuperSPARC		3	.7	1700	1.6	256	48	13%	$272
Pentium		3	.8	1500	1.5	296	40	9%	$417

source: "Estimating IC Manufacturing Costs" Linley Gwennap _Microprocessor Report_ Aug 2 1993 p15

Cost v. Price

- Component cost: 15% - 33% of list price
- Direct costs (6-8%) -- recurring costs: labor, purchasing, scrap, warrenty
- Gross Margin (34-39%) -- nonrecurring costs: R&D, marketing, sales, equipment maintenence, rental, financing, cost, pretax profits, taxes

--> Avg selling price

- Average Discount (25%-40%) -- volume discounts/retailor markup

--> List price

SLIDE:

Chip Prices (Aug 1993) (10K units price)

Chip	Area mm^2	Mfg cost	Price	Multiplier	Comment
386DX	43	$9	$31	3.4	Intense COmpetition
486DX2	81	$35	$245	7.0	No Competition
PowerPC 601	121	$77	$280	3.6	
DEC Alpha	234	$202	$1231	6.1	Recoup R&D?
Pentium	296	$473	$965	2.0	Early in shipments

Volume v. Cost

Learning Curve: Rule of thumb "When volume doubles, cost reduces 10%"

- _A DEC View of Computer Engineering_ C. G. Bell, J. C. Mudge, and J. E. McNamara. Digital Press, Bedford, MA 1978.

example: NOW v. custom MPP

40MPPs @ 200 nodes = 8000 nodes/year

100,000 Workstations/year

12.5X + 2^3.6 ==> 0.9^3.6 ==> 0.68

--> cost should be 1/3 less for same components

examples: PCs v. Workstations

	1990	1992	1994	1997
PC	24M	33M	44M	65M
WS	0.41M	0.58M	0.68M	0.98M
Ratio	59	57	65	67

65x ~ 2^6 ==> 0.9^6.0 = 0.53

High Margins on High-end Machines

- o R&D considered return on investment (ROI) = 10%
  - --> every $1 in R&D must generate $7-$13 in sales
- o High end machines need more R&D
- o Sell fewer high end machines
  - --fewer to amortize R&D
  - -- much higher margins

o example: price of $1MB Memory (Jan 1994)

PC $40 (Mac Quadra)
WS $42 (SS-10)

QUESTION: Guess how much $1MB of mainframe memory costs?

Mainframe $19200 (IBM 3090)
Supercomputer $600 (M90 DRAM)
$1375 (C90 15ns SRAM)

o example: recouping development costs on low-volume microprocessors

MIPS R4000 cost $30M to develop (Hennessy)
SGI/MIPS sells 300K R4000s over product lifetime
--> $100/chip

80486 cost $100M (rumor)
Intel sells 50M?
--> $2/chip

Price/performance -- gross margin v. market segment

BARGRAPH the following

	Mini	Workstation	PC
Avg Discount	1.2	1.3	.3
Gross Margin	2.2	1.3	0.4

Direct Costs	0.3	0.2	0.1
Component Costs	1.0	1.0	1.0

Impact of Margin Shrink on Society/Industry

Gross Margin ==> entire research and development budget

What does this mean to your jobs?

****

Summay - 1 min

****

summarizing performance
Amdahl's law
CPU performance

<project discussion>

cost