M. Tech
CIT/ME

Course name : Advanced Misoprocessor – Electrive

Topic – 1

* computer system performance and its evaluation : A brief account
* CPU performance equation and differentiation of CIS & RISC architectures

Performance sumarization :

Exaple 1

Suppose we have two programs or process – P1 and P2 and three computors : A, B, C

Suppose
A is 10 t's faste than B for P1
B is 10 t's faste than A for P2
A is 20 t's faste than C for P1
C is 50 t's faste the A for P2
C is 5 t's faster the B for P2 and so on

From here : Performance of a Program on a m/c depends on type of program (ie Application)

: Taking individually, one statement may be of some use; but relative performance is unclear wrsto m/cs.

Exaple 2

	Comp A	Comp B	Comp C
P1 (secs)	1	10	20
P2 (secs)	1000	100	20
Total t's (secs)	1001	110	40

(Execution t's of 2 Progs on 3 m/s)

A summary of this table is as!

C is $\frac{110}{40}$ times faster than B for P1 and P2.

C is $\frac{1001}{40}$ times faster than A for P1 and P2.

The above table can give relative performance of machines if P1 and P2 run equal no. of times.

In that case C is faster for P1 and P2

Suppose : Time taken for P1 on m/c A = T_1

" " P1 " " B = T_2

" " P1 " " C = T_3

Av. Execution time = $\frac{T_1 + T_2 + T_3}{3}$;

Similarly Av. execution time of P2 can be found

Thus the av. execution time that tracks total execution time is the arithmetic mean :

$$\frac{1}{n}\sum_{i=1}^{n} \text{Time}_i$$ (Time$_i$ is execution time of ith Program of total of n work load)

Weighted execution time : As shown above, for equal mix i.e. if programs run equal no. of times execution time is governed by arithmetic mean.

When there is unequal mix of programs in the workload, then we assign weightage (w_i) to each program to indicate relative frequency of program in workload

eg. If 20% of tasks in work load are P1 ⇒ weightage = 0.2
80% of tasks in work load are P2 ⇒ weightage = 0.8

Weightage Arithmetic mean $\sum \text{weightage}_i \times \text{Time}_i$ (Weightage$_i$ is freq. of ith Prog in work load.

Time$_i$ is Execution time of Prog

Exaple 3.

Data from Exaple 2 with three diffsent weightings, each proportinal to exeution tie of work load with a given mix:

	Progs			Weightages		
	A	B	C	w(1)	w(2)	w(3)
P1(sec)	1	10	20	0.5	0.909	0.999
P2 (sec)	1000	100	20	0.5	0.091	0.001

Weighted arithmetic mean of three m/cs (exeution times)
Machins are A, B, C and two progs P1, P2 use three weightings w(1), w(2) and w(3)

Soln: For weightage (w1), the arithmetic mean (weighted) for mix of progms P1 and P2 on m/c A is given by

$1 \times 0.50 + 1000 \times 0.5 = 500.50$

For weightage (w2), the arithmetic mean (weighted) for mix of programs P1 & P2 on m/c A is given by

$1 \times .0909 + 1000 \times 0.091 = 91.91$

|||ly for w(3): $1 \times 0.999 + 1000 \times 0.001 = 1.999 \simeq 2$

Thus for diffeent mix of programs (diff. weightages): The execution tie's ~~(weig~~ are diffsent. You can compute for other m/cs using w(1), w(2), w(3)

~~Arithmetic mean (w1)~~	Progs A	B	C
Arithmetic mean (W1)	500.5	55	20
Arithmetic mean (W2)	91.91	18.19	20
Arithmetic mean (W3)	2.0	10.09	20

∴ ~~For a part~~ This leads us to an important design principle for performance enhancement

“MAKE COMMON CASE FAST”

In general purpose computers used for all typs of applications; having invariably conflicting requirements attainment of ths design principle is not possible.

~~Therefore~~ This has led us to design ~~computers~~ different computers for different applications:

e.g. For I/O application

For computational problems

and ~~&~~ Embedded ~~processors for~~ Systems/ ~~for~~ V. specific applications.

Amdhal's law

The law states, “The performance improvement to be gained by using some faster mode of execution is limited by fraction of time the faster mode can be used.”

The law defines speed up that can be gained by using a particular feature.

Suppose enhancement has been made

$$\text{Speed up} \triangleq \frac{\text{Performance of entire task using enhancement when possible}}{\text{Performance of entire task without using enhancement}}$$

$$= \frac{\text{Execution time for entire task without enhancement}}{\text{Execution time of entire task using enhancement when possible}}$$

Speed up tells us how much faster ~~m/c~~ task will run using the machine as opposed to original m/c.

Execution enhancement depends on two factors:

1. The Fraction of computation time in orginal machine that can be converted to take adventage of enhancement

   eg. If unutalized time of m/c is 60 secs & suppose '10' secs of m/c are utalized out of 60 secs. ~~out~~ ~~of~~ by virtue of enhancent

   ∴ Fraction enhanced = $10/60$

2. Speed up Enhanced:

   This is improvement gained by enhanced exeution mode, i.e, how much faster the task would run if the enhanced mode was used for a program. This may be given as:

$$\frac{\text{Time of orginal mode (Ex. time)}}{\text{Time of Enhanced mode (Ex. tie)}}$$

eg. Suppose ~~org~~ in its orginal mode 5 secs are required to run a program. If the m/c is enhanced and in enhanced mode the progm runs only in 2 secs

Then improvent (Enhancement) = $5/2 > 1$

Exaple: Suppose using enhancement processor is 10 tis faste the orginal processor. Assume orginal processor is busy with computati 40% tie and waiting for I/O 60% tie.

What is overall speed up gained by incorporating enhancement.

1. Execution time : This can be defined in different ways depending on what we count. Most straight forward definition is: Wall clk time, Response tie or Elapsed time This time is latency to complete a task.

Elapsed time includes:

a) Disk access

b) Memory access

c) I/O activities

d) Operating System overheads etc.

Thus elapsed time is tie is tie as seen by the user.

2. CPU tie : This is tie during which CPU is executing this tie does not include waiting tie etc. (as may be case in elapsed tie).

CPU Time < CPU tie spent on program (User CPU tie)
CPU tie spent on O.S.

Exaple: If we give a comand in Unix : Time

Time : 90.7u 12.9s 2:39 65%.

90.7u	12.9s	2:39	65%
User CPU tie	System CPU tie	Elapsed tie	% of CPU tie actually used (excluding waiting tie)

So more than $\frac{1}{3}$rd of tie was spent on waiting for I/O or running other progrs (in multiprogrg envi) or both. Many ties we ignore CPU tie spent on O.S. because of inaccuracies in O.S.

∴ System Performance = Elapsed tie on unloaded System

CPU Performance = User CPU tie on an unloaded system.

CPU Performance Equation :

Computes are constructed using a clk runing at a constant rate. These discrete time events are called clock periods, clk cycls etc.

Clk Period is designated by

a : Duration (n.s.) or

b : Rate (GHz)

$$\text{CPU Time} = \text{CPU clk cycls per program} \times \text{clk cycle time}$$

$$= \frac{\text{CPU clk cycls for progam}}{\text{Clk rate}}$$

In addition to ~~clk cycls~~ numbes of clk cycls needed to execute a program, we can also count the no. of instructions executed; This is also called Path length or Instruction Count (IC)

If we know no. of clk cycls & IC we can calculate No. of clk cycls per Instruction (CPI). Designers sometimes use Instructions per clk cycle (IPC) which is inverse of CPI

$$CPI = \frac{\text{CPU clk cycles for a program}}{IC}$$

↳ Figure of merit, provides insight into different styles of Instruction sets & implementations.

Total clk cycls = IC × Av. no. of clk cycls per instructn

CPU te = Instruction Count × clk cycle te × cycls per inst.

$$\text{CPU te} = \frac{\text{Instruction cnt} \times \boxed{\text{clk cycle te}}}{\text{clk rate (IPC)}} \qquad \text{cycls/inst}$$

Expanding the formula in the units of measurement

$$\underset{(\text{Ic})}{\left(\frac{\text{Inst}}{\text{Progm}}\right)} \times \underset{(\text{CPI})}{\left(\frac{\text{clk cycls}}{\text{Inst}}\right)} \times \underset{\left(\frac{1}{\text{clk rate}}\right)}{\left(\frac{\text{Seconds}}{\text{clk cycle}}\right)} = \frac{\text{Seconds}}{\text{Progm}} = \text{CPU te}$$

or $\cancel{\text{CPU te} = \frac{\text{time}}{\text{Progm}}} \Rightarrow \cancel{\frac{\text{time}}{\text{cy}}}$

$$\text{or CP Time} = \frac{\text{Time}}{\text{Progm}} = (\text{Ic})\,(\text{CPI})\left(\frac{1}{\text{clk rate}}\right) \qquad \text{cycls/sec}$$

$$\frac{\text{Inst}}{\text{Progm}} \times \frac{\text{sec}}{\text{cycls}}$$

⇓ Reduce for CISC ⇓ Reduce for RISC.

CPU te or CPU performance is dependent on

a) clk cycls or clk rate (Processor specific)
b) clk cycls per inst
c) Instruction count

~~A 10%~~ A x% improvement in any one of them leads to x% improvement in CPU te.

It is difficult to change one of the parameters in isolation because basic technologies involved in changing each characteristic are independent

- clk cycle te : HW technology & organizati
- CPI : organization — inst. set architecture
- IC : Inst set architecture & compiler tech.

Examples to Solve and Submit

1. Suppose we have two implementations of some instructions set architecture. Computer A has clk cycle tie of 250 ps and CPI of 2.0 for some program. Computer B has CLK cycle tie of 500 ps. and CPI of 1.2 for same program. Which computer is faster for this program and by how much?

2. A program runs in 10 sec on computer A which has 4 GHZ CLK. We are trying to help computer designer to build a computer B that will run this progra in 6 Sec. The designer has ~~desined~~ determined that a substantial increase in CLK rate is possible; but this increase will effect rest of CPU design, causing computer B to require 1.2 times as many CLK cycles 9 computer A for this program What CLK rate should designer target?

3. Suppose we are considering enhancement of server systes. The new CPU is 10 tims faster ~~on computation~~ ~~in~~ than orginal. Assuming That the orginal CPU is busy with computation 40% of tie and is waiting for I/O 60% of the tie. What is overall speedup gained by incorpating enhancement.

4. What is significance of MIPS rate? A 40 MHZ processor was used to execute a bench mark program with following inst. mix

Inst. type	Inst count	CLK cycle count
Integer Arithmetic	45000	1
Data Tx	32000	2
FP op	15000	2
Control Tx	8000	2

Determine: CPI, MIPS rate and Execution tie for program.

5. What are bench marks? Discuss SPEC bench mark

6. Dyssentiate: ~~Price~~ Peformance, Reliablity Peformance, Power peformance, & Green ~~[illegible]~~