CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020
Batch: 2016-17
(LECTURE # 3)

FAKHRA AFTAB
LECTURER
DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 2

Types of Models

Performance Metrics

Characteristics of a Good Performance Metric

Workload
(Explanation of Throughput & Mean Response Time vs. Load)

Chapter # 1 (Cont'd)

COMPUTER SYSTEMS PERFORMANCE MODELING AND EVALUATION

Amdahl’s Law

Amdahl's law states that in parallelization, if *f* is the proportion of a system or program that can be made parallel, and *1 - f* is the proportion that remains serial, then the maximum speedup that can be achieved using ‘n’ number of processors is given by:

$$S = \frac{1}{1 - f + f/n}$$

$$S_{ideal} = \lim_{n \to \infty} \frac{1}{1 - f}$$

Question: Suppose that enhancement is achieved which runs 10 times faster than the original machine but is only usable 40% of time. Determine the speed up.

Amdahl's Law - Pitfall

- Improving an aspect of machine's performance and expecting a proportional improvement in overall performance.

$$T_{\text{improved}} = \frac{T_{affected}}{Improvement\ factor} + T_{\text{unaffected}}$$

- Example: Suppose that a program runs in 100 seconds on a machine with multiply responsible of 80 seconds of this time.
  - How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?
  - How about making it 5 times faster? State your conclusion as well.

Solution (a)

- To be 4 times faster the program should run in 100/4 (i.e. 25 seconds)
- T_{improved} = 25 seconds
- $T_{affected}$ = 80 seconds
- $T_{unaffected}$= 20 seconds

- By substituting the values in the equation,

Improvement Factor = 16

Solution (b)

Means VS End Based Metric

Reliability is one of the most important characteristics of performance metric.

i) **Means-based:** Measures what was done whether or not it was useful. For E.g. NOP instruction, multiply by 0 etc. It results in an unreliable metric.

ii) **End-based:** Measures progress towards a goal. Only counts what is actually accomplished. It results in reliable metric.

Example (a)

Consider the given vector dot-product routine:

```
s = 0;
for (i = 1; i < N; i++)
      s = s + x[i] * y[i];
```

Let

t_A = # of clock cycles for one addition floating point operation

t_M = # of clock cycles for one multiplication floating point operation

i) Total time required to execute this program:

$$t_1 = N\,(t_A + t_M)$$

ii) Execution Rate:

$$R_1 = \frac{2N}{N\,(tA + tM)}\ \text{FLOPS/cycle}$$

Example (b)

Consider the given vector dot-product routine:

```
s = 0;
for (i = 1; i < N; i++)
if (x[i] != 0 && y[i] != 0)
     s = s + x[i] * y[i];
```

Let

t_{if} = # of C.C. to execute one 'if' instruction

f = fraction of N for which both x[i] and y[i] are non-zero.

i) Total time required to execute this program:

$$t_2 = fN\,(t_A + t_M) + Nt_{if}$$

$$t_2 = \{f\,(t_A + t_M) + t_{if}\}N$$

ii) Execution Rate:

$$R_2 = \frac{2Nf}{\{f\,(t_A + t_M) + tif\,\}N}\ \text{FLOPS/cycle}$$

$$R_2 = \frac{2f}{f\,(t_A + t_M) + tif}\ \text{FLOPS/cycle}$$

Question:

Suppose that:

t_{if} = 4 , t_{+} = 5, t_{*} = 10, Clock Rate = 250 MHz and f = 10%.

i) Calculate the execution time and execution rate for both examples (a) and (b).

ii) Compare the results using mean-based and end-based metric and verify that end-based metric is reliable.

Solution

- t_1 = 60N ns
- t_2 = 22N ns
- $S_{2,1}$ = 60N/22N = 2.73
- R_1 = 33 MFLOPS
- R_2 = 9.09 MFLOPS

- On the basis of execution time, Program 2 is ((60/22)-1) * 100 = 172% faster than Program 1.
- On the basis of execution rate, Program 2 is (1 – 9.09/33) * 100 =72% slower than Program 1.
- Which one is end based?

Performance Measures

- Computer systems architect and designers look for configurations of computer systems elements so that system performance meets desired measures of the user applications.

- All performance measures deal with three basic issues:
  - how quickly a given task can be accomplished,
  - how well the system can deal with failures and other unusual situations, and
  - how effectively the system uses the available resources.

Categories of Performance Measures

1) Responsiveness – Measures intended to evaluate how quickly a given task can be accomplished by the system.
2) Usage Level – Evaluate how well the various components of the system are being used.
3) Mission-ability – The measures indicate if the system would remain continuously operational for the duration of a mission.
4) Dependability – Indicate how reliable the system is over the long run.
5) Productivity – These measures indicate how effectively a user can get his or her work accomplished.

Classification of Computer Systems

Classification of CS	Relevant Measures	Examples
General Purpose Computing	Responsiveness, Usage level and Productivity	All generic systems
High Availability	Responsiveness & Dependability. Data corruption or destruction is unacceptable	Banks, airline, or telephone databases, switching systems, etc.
Real Time Control	Responsiveness and Dependability	Air Traffic Control Systems, Command Control Systems etc.
Mission Oriented	extremely high levels of reliability over mission time	fly-by-wire airplanes, battlefield systems, and space-crafts.
Long Life	Highly dependable; require considerable intelligence for diagnostics and repair either automatically or by remote control from ground station	Systems used for unmanned spaceships without manual diagnostics and repairs.

QUICK REVIEW OF PERFORMANCE DEFINITION

For some program running on machine X,

- Performance = 1 / Execution time$_X$

"X is n times faster than Y"

- Performance$_X$/ Performance$_Y$= n

Computing CPU Time

- The time to execute a given program can be computed as:

  CPU time = CPU clock cycles x clock cycle time ….. (i)

- Since clock cycle time and clock rate are reciprocals

  CPU time = CPU clock cycles / clock rate

- The number of CPU clock cycles can be determined by

  CPU clock cycles = (instructions/program) x (clock cycles/instruction)

  = Instruction count x CPI

- which gives

  CPU time = Instruction count x CPI x clock cycle time CPU time

  = Instruction count x CPI / clock rate

Computing CPI

- The CPI is the average number of cycles per instruction.
- If for each instruction type, we know its frequency and number of cycles need to execute it, we can compute the overall CPI as follows:

$$\underline{\mathbf{CPI = \Sigma CPI \times F}}$$

Operation	Frequency	CPI	CPI * F
ALU	50%	1	0.5
Load	20%	5	1
Store	10%	3	0.3
Branch	20%	2	0.4
Total	**100%**		**2.2**

Question:

You are lead designer for a company that produces microprocessors for a graphics accelerator. The performance analysis department reported the following instruction usage profile for the popular benchmark suite *SmartBench:*

Instruction Class	CPI	Usage
Integer/Logical	2	30%
Load/Store	2	25%
Floating Point	6	15%
Branches	2	10%
Others	4	20%

a) Given that *SmartBench* executes in 2 seconds on 500 MHz processor, how many machine instructions were executed ?

b) How long will *SmartBench* take to execute if floating point instructions are made 4 times faster?

Answers to Part (a) and (b)

a) 333.33 M Instructions
b) 1.775 seconds

EVENTS

- Event is an action of interest in our system.

- Events of interest to computer engineers/architect are:
  - The beginning or end of a clock cycle
  - The beginning or end of a computer's instruction execution cycle
  - The reading of a memory location
  - The initiation of a block data transfer from a secondary storage device
  - The initiation of a process or task.

- In terms of performance assessment, the system analyst must have understanding of the relationship of events to each other.

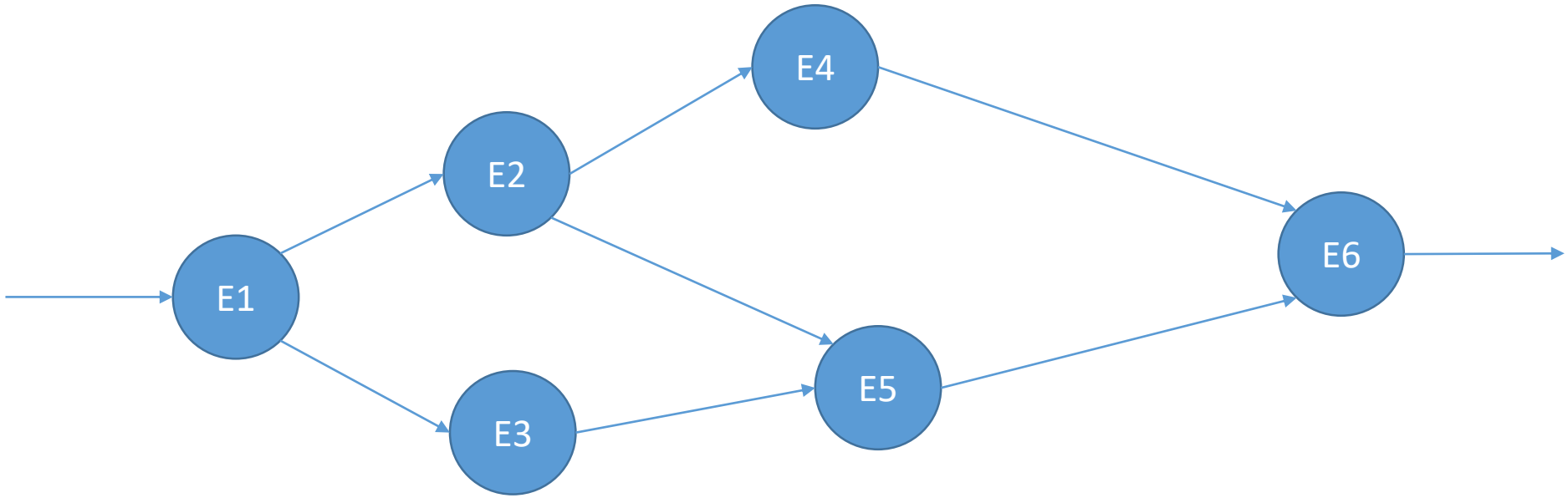

Fig 1: Event Partial Ordering