Universiti Malaysia PAHANG
Engineering • Technology • Creativity

BCN1043

COMPUTER ARCHITECTURE & ORGANIZATION

By
Dr. Mritha Ramalingam

Faculty of Computer Systems & Software Engineering
mritha@ump.edu.my

AUTHORS

- **Dr. Mohd Nizam Mohmad Kahar** (mnizam@ump.edu.my)
- **Jamaludin Sallim** (jamal@ump.edu.my)
- **Dr. Syafiq Fauzi Kamarulzaman** (syafiq29@ump.edu.my)
- **Dr. Mritha Ramalingam** (mritha@ump.edu.my)

Faculty of Computer Systems & Software Engineering

Learning outcomes

- Explain how the components of system architecture contribute to improving its performance.
- Describe Amdahl's law and discuss its limitations.
- Design and conduct a performance-oriented experiment.
- Use software tools to profile and measure program performance.
- Explain the importance of locality in determining performance.
- Describe why things that are close in space take less time to access.
- Calculate average memory access time and describe the tradeoffs in memory hierarchy performance in terms of capacity, miss/hit rate, and access time.

Computer Performance Evaluation

- Computer Performance Evaluation
- Performance definition
- Benchmark
- Summarizing performance
- Amdahl’s law
- Cycles Per Instruction

What Does Performance Mean?

- Response time
  - A simulation program finishes in 5 minutes
- Throughput
  - A web server serves 5 million request per second
- Other metrics
  - MIPS (million instruction per second)
  - MFLOPS
  - Clock frequency

Execution Time

- Processor design is concerned with processor consumed by program execution. Shorter execution time=>
  - Shorter response time
  - Higher throughput
- **Execution time = #inst×CPI×Cycletime**
  - What affects #inst, CPI, and cycle time?
  - Almost all designs can be interpreted
- Any other metrics is meaningful only if consistent with execution time

Performance of Computers

Performance is defined for ***a program and a machine***.

How to compare computers? Need benchmark programs:

- Real applications: scientific programs, compilers, text-processing software, image processing
- Modified applications: providing portability and focus
- Kernels: good to isolate performance of individual features
  - Lmbench: measure latency and bandwidth of memory, file system, networking, etc.
- Toy benchmarks
- Synthetic benchmarks: matching average execution profile

Performance Comparison

"X is n times faster than Y":

$$\frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution time}_y}{\text{Execution time}_x} = n$$

- *n*: speedup if we are considering an enhancement, optimization, etc.
- What does "improving" mean?
  - Improve performance: decrease execution time, increase throughput
  - Improve execution time: decrease execution time
  - Degrade performance: the reverse of the above; brings negative speedup

Benchmark Suite

- Benchmark suite is a collection of benchmarks with a variety of applications
  - Alleviating weakness of a single benchmark
  - More representative for computer designers to evaluate their design
  - ***Benchmarks test both computer and compilers, and OS in many cases***
- Desktop benchmarks: CPU, memory, and graphics performance
- Sever benchmarks: throughput-oriented, I/O and OS intensive
- Embedded benchmarks: measuring the ability to meet deadline and save power

Summarizing Performance

Given the performance of a set of programs, how to evaluate the performance of machines?

	A	B	C
P1 (secs)	1	10	20
P2 (secs)	1000	100	20
Total (secs)	1001	110	40

- Which computer is the “best” one?

Arithmetic Mean

- Total execution time / (number of programs)

$$\frac{1}{n}\sum_{i=1}^{n}\text{Time}_{\text{i}}$$

  - Simple and intuitive
  - Representative if the user run the programs an equal number of times

Weighted Arithmetic Mean

- Give (different) weights to different programs

$$\sum_{i=1}^{n} \text{Weight}_i \times \text{Time}_i, \qquad \sum_{i=1}^{n} \text{Weight}_i = 1$$

  - Considering the frequencies of programs in the workload

Geometric Means

- Based on relative performance to a reference machine

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

- Relative performance is consistent with different reference machines

$$\frac{\text{Geometric mean}(X_i)}{\text{Geometric mean}(Y_i)} = \text{Geometric mean}\left(\frac{X_i}{Y_i}\right)$$

  - If C is 2x faster than B (using B as the reference), B is 2x faster than A (A as the reference), then C is 4x faster than A (A as the reference)

Harmonic Mean

- Given speedups s1, s2, …, s_n, the average speedup by harmonic mean is

  n / (1/s1 + 1/s2 + … + 1/s_n)

Why not arithmetic mean?

WHAT IS PERFORMANCE?

Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

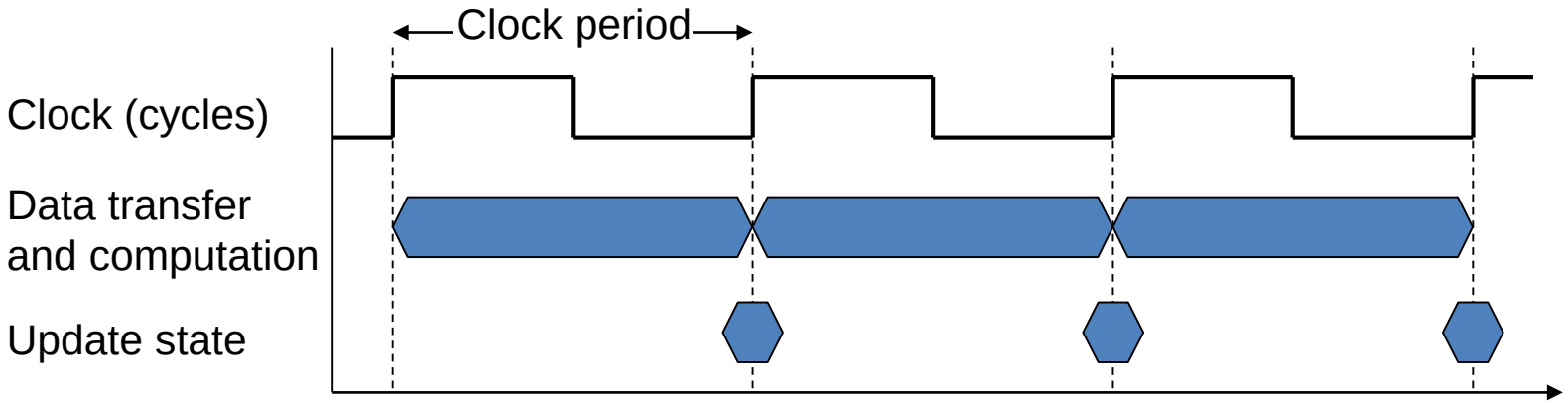
- Example: time taken to run a program
  - 60s on A, 30s on B
  - Execution Time_B / Execution Time_A= 30s / 60s = 0.5
    So A is 0.5 times faster than B
  - or B is 2 times faster than A

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

- Will continue