Performance (from Chapter 1)

Some charts based on lecture slides provided by Morgan/Kauffman

Definition: Performance

$$\text{performance}(x) = \frac{1}{\text{execution\_time}(x)}$$

" X is n times faster than Y" means

$$n = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution\_time}(Y)}{\text{Execution\_time}(X)}$$

Some charts based on lecture slides provided by Morgan/Kauffman

2

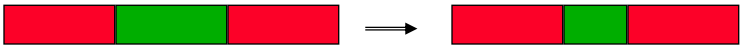
Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Best you could ever hope to do:

$$Speedup_{maximum} = \frac{1}{(1 - Fraction_{enhanced})}$$



Some charts based on lecture slides provided by Morgan/Kauffman

Amdahl's Law example

- **Application: Network Server, 60% of time is spent waiting on I/O**
- **We replace the server's processor with one that is 10 times faster**

$$Speedup_{overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

$$= \frac{1}{(1 - 0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

The new processor improves only the 40% of the execution time that is not I/O

10 times faster processor provides only 56% speedup

Some charts based on lecture slides provided by Morgan/Kauffman

Performance of a computer for a specific program (that you know a lot about)

"Processor performance equation"

CPU time of a program
= # of instructions executed by the program
x Average Clock Cycles per instruction (CPI)
x Clock cycle time

Example:

- A program executes 1000 instructions on a particular processor

- CPI of this processor = 2

- Processor has a 4GHz clock (cycle time = 250psec)

CPU time = 10^3 x 2 x 250x10^{-12}

= 500 nsec

Some charts based on lecture slides provided by Morgan/Kauffman

5

To get a more general idea of a computer's performance

- **Usually rely on benchmarks vs. real workloads**
- **To increase predictability, collections of benchmark applications, called *benchmark suites*, are popular**
- **SPECCPU: popular desktop benchmark suite**
  - **CPU only, split between integer and floating point programs**
  - **SPECint2000 has 12 integer, SPECfp2000 has 14 integer pgms**
  - **SPECSFS (NFS file server) and SPECWeb (WebServer) added as server benchmarks**
- **Transaction Processing Council measures server performance and cost-performance for databases**
  - **TPC-C Complex query for Online Transaction Processing**
  - **TPC-H models ad hoc decision support**
  - **TPC-W a transactional web benchmark**
  - **TPC-App application server and web services benchmark**

Some charts based on lecture slides provided by Morgan/Kauffman

6

How Summarize Suite Performance

- Arithmetic average of execution time of all pgms?
  - But they vary by 4X in speed, so some would be more important than others in arithmetic average
- Could add a weights per program, but how pick weight?
  - Different companies want different weights for their products
- SPECRatio: Normalize execution times to reference computer, yielding a ratio proportional to performance =

$$\frac{\text{time on reference computer}}{\text{time on computer being rated}}$$

How Summarize Suite Performance

- If program SPECRatio on Computer A is 1.25 times larger than Computer B, then

$$1.25 = \frac{SPECRatio_A}{SPECRatio_B} = \frac{\frac{ExecutionTime_{reference}}{ExecutionTime_A}}{\frac{ExecutionTime_{reference}}{ExecutionTime_B}}$$

$$= \frac{ExecutionTime_B}{ExecutionTime_A} = \frac{Performance_A}{Performance_B}$$

Dependability / Reliability

Some charts based on lecture slides provided by Morgan/Kauffman

Define and quantity dependability

- How decide when a system is operating properly?
- Infrastructure providers now offer Service Level Agreements (SLA) to guarantee that their networking or power service would be dependable
- Systems alternate between 2 states of service with respect to an SLA:

1. Service accomplishment, where the service is delivered as specified in SLA
2. Service interruption, where the delivered service is different from the SLA

- Failure = transition from state 1 to state 2
- Restoration = transition from state 2 to state 1

Some charts based on lecture slides provided by Morgan/Kauffman

Define and quantity dependability

- *Module reliability* = measure of continuous service accomplishment (or time to failure).
  2 metrics
1. *Mean Time To Failure* (*MTTF*) measures Reliability
2. *Failures In Time* (*FIT*) = 1/MTTF, the rate of failures
   - Traditionally reported as failures per billion hours of operation
- *Mean Time To Repair* (*MTTR*) measures Service Interruption
  - *Mean Time Between Failures* (*MTBF*) = MTTF+MTTR
- *Module availability* measures service as alternate between the 2 states of accomplishment and interruption (number between 0 and 1, e.g. 0.9)
- *Module availability = MTTF / (MTTF + MTTR)*

Some charts based on lecture slides provided by Morgan/Kauffman

11