CMSC 132: Computer Architecture

Asst. Prof. Reginald Neil C. Recario
rncrecario@gmail.com
Institute of Computer Science
University of the Philippines Los Baños

Summarizing Performance Results

- One must evaluate a number design choices for their relative quantitative benefits across a suite of benchmarks believed to be relevant.

- Consumers trying to choose a computer will rely on performance measurements from benchmarks.

Summarizing Performance Results

- In the ideal case, the suite resembles a statistically valid sample of the application space, but such a sample requires more benchmarks than are typically found in most suites and requires a randomized sampling, which essentially no benchmark suite uses.

Summarizing Performance Results

- With the chosen measure of performance with a benchmark suite, we would like to be able to summarize the performance results of the suite in a single number.
- Straight forward solution is to compare arithmetic means of different programs.
  - Problem with arithmetic mean?
  - Some programs run longer. Equal weights.

Summarizing Performance Results

- An alternative would be adding weighting factor to each benchmark and use the weighted arithmetic mean as the single number to summarize performance.
  - Problem? How do you decide on the weights?
  - Each company may have a set of weights they favor

Summarizing Performance Results

- Another solution is to normalize execution times to a reference computer.
- Example:
  - SPECRatio of Computer A is 1.25 higher than Computer B

Summarizing Performance Results

- The ratio can be expressed as

$$1.25 = \frac{\text{SPECRatio A}}{\text{SPECRatio B}}$$

$$= \frac{\dfrac{\text{Execution Time}_{\text{reference}}}{\text{Execution Time}_{\text{A}}}}{\dfrac{\text{Execution Time}_{\text{reference}}}{\text{Execution Time}_{\text{B}}}}$$

Summarizing Performance Results

- And can expressed as

$$= \frac{\text{Execution Time}_B}{\text{Execution Time}_A}$$

$$= \frac{\text{Performance}_A}{\text{Performance}_B}$$

Summarizing Performance Results

- SPECRatio is a ratio rather than an absolute execution time
- The mean must be computed using the geometric mean

$$Geometric\ mean = \sqrt[n]{\prod_{i=1}^{n} sample_i}$$

Summarizing Performance Results

- Sample_i is the SPECRatio for program_i
- Geometric mean ensures two important properties:
  - The geometric mean of the ratios is the same as the ratio of the geometric means.
  - The ratio of the geometric means is equal to the geometric mean of the performance ratios, which implies that the choice of the reference computer is irrelevant.

Summarizing Performance Results

- Does a single mean that summarizes performance suite well?
  - Variability Distribution
  - Can possibly use bell-shaped Normal Distribution
  - Can also use LogNormal Distribution

Quantitative Principles of Computer Design

- Take advantage of parallelism
  - Improve throughput performance by using multiple processors (on independent processes of course!)

Quantitative Principles of Computer Design

- Principle of Locality
  - 90/10 Principle [Based on Pareto Principle (80/20)]
  - A program spends 90% of its execution time in only 10% of the code
  - Types of Locality
    - Temporal Locality
    - Spatial Locality

Quantitative Principles of Computer Design

- Focus on the Common Sense
  - In making a design trade-off, favor the frequent case over the infrequent case.

Quantitative Principles of Computer Design

- Amdahl's Law
  - The performance gain that can be obtained by improving some portion of a computer.
  - States that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Quantitative Principles of Computer Design

- Amdahl's Law defines the speedup that can be gained by using a particular feature.
- Speedup is the ratio
- Speedup = A/B

  where

  A = Performance for entire task using the enhancement when possible

  B = Performance for entire task without using the enhancement

Quantitative Principles of Computer Design

- Alternatively, Speedup can also be expressed as
  - Speedup = C/D

where

C = Execution time for entire task without using the enhancement

D = Execution time for the entire task using the enhancement when possible

Quantitative Principles of Computer Design

- Speedup tells us how much faster a task will run using the computer with the enhancement as opposed to the original computer.

Quantitative Principles of Computer Design

- Amdahl's Law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

Quantitative Principles of Computer Design

- The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement
  - Less than 1
- The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program
  - Greater than 1

Quantitative Principles of Computer Design

- The execution time using the original computer with the enhanced mode will be the time spent using the unenhanced portion of the computer plus the time spent using the enhancement:

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times$$

$$((1 - \text{Fraction}_{\text{enhanced}}) + (\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}))$$

Quantitative Principles of Computer Design

- The overall speedup is the ratio of the execution times:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}}$$

$$\frac{1}{((1 - \text{Fraction}_{\text{enhanced}}) + (\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}))}$$

Quantitative Principles of Computer Design

- Example
  - Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

Quantitative Principles of Computer Design

- Answer:

$\text{Fraction}_{\text{enhanced}} = 0.4$

$\text{Speedup}_{\text{enhanced}} = 10$

$$\begin{aligned}\text{Speedup}_{\text{overall}} &= 1/(0.6 + (0.4/10)) \\ &= 1/(0.64) \\ &= \mathbf{1.56}\end{aligned}$$

Overall speedup is 1.56x.

Quantitative Principles of Computer Design

- Amdahl's Law expresses the law of diminishing returns: The incremental improvement in speedup gained by an improvement of just a portion of the computation diminishes as improvements are added.

Quantitative Principles of Computer Design

- Example
  - A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10.

Quantitative Principles of Computer Design

- The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Quantitative Principles of Computer Design

- Answer:

  $\text{Speedup}_{\text{FPSQR}} = 1 /((1 - 0.2) + (0.2/10))$

  $= 1/0.82$

  $\mathbf{= 1.22}$

  $\text{Speedup}_{\text{FP}} = 1/((1-0.5)+(0.5/1.6))$

  $= 1/0.81$

  $\mathbf{= 1.23}$

  There is a slightly better performance comparing the two alternative enhancements.

The Processor Performance Equation

- All computers are constructed using a clock running at a constant rate.
- These discrete time events are called *ticks*, *clock ticks*, *clock periods*, *clocks*, *cycles*, or *clock cycles*.

The Processor Performance Equation

- CPU time for a program can then be expressed two ways:

  CPU time = CPU clock cycles for a program x Clock cycle time

  OR

$$\textbf{CPU Time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

The Processor Performance Equation

- We can also count the number of instructions executed—the instruction path length or instruction count (IC). If we know the number of clock cycles and the instruction count, we can calculate the average number of clock cycles per instruction (CPI). Designers sometimes also use instructions per clock (IPC), which is the inverse of CPI.

The Processor Performance Equation

- CPI is computed as

$$\mathbf{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction Count}}$$

The Processor Performance Equation

- By transposing instruction count in the above formula, clock cycles can be defined as IC x CPI. This allows us to use CPI in the execution time formula:

  CPU time = Instruction Count x Cycles per instruction x Clock cycle time

The Processor Performance Equation

- Expanding the first formula into the units of measurement shows how the pieces fit together

$$\frac{\text{Instruction}}{\text{Program}} * \frac{\text{Clock cycle}}{\text{Instruction}} * \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU Time}$$

The Processor Performance Equation

- As the formula demonstrates, processor performance is dependent upon three characteristics: clock cycle (or rate), clock cycles per instruction, and instruction count.

The Processor Performance Equation

- Furthermore, CPU time is equally dependent on these three characteristics: A 10% improvement in any one of them leads to a 10% improvement in CPU time.

The Processor Performance Equation

- Unfortunately, it is difficult to change one parameter in complete isolation from others because the basic technologies involved in changing each characteristic are interdependent:
  - Clock cycle time—Hardware technology and organization
  - CPI—Organization and instruction set architecture
  - Instruction count—Instruction set architecture and compiler technology

The Processor Performance Equation

- Sometimes it is useful in designing the processor to calculate the number of total processor clock cycles as

$$\text{CPU Clock Cycles} = \sum_{i=1}^{n} IC_i * CPI_i$$

The Processor Performance Equation

- The form mentioned in the previous slide can be used to express CPU time as

$$\text{CPU Time} = \left(\sum_{i=1}^{n} \text{IC}_i * \text{CPI}_i \right) * \text{Clock cycle time}$$

The Processor Performance Equation

- And overall CPI as

$$\text{CPI} = \frac{\left(\sum_{i=1}^{n}(\text{IC}_i * \text{CPI}_i)\right)}{\text{Instruction count}} = \sum_{i=1}^{n}\left(\frac{\text{IC}_i}{\text{Instruction count}} * \text{CPI}_i\right)$$

The Processor Performance Equation

- Example:
  - Suppose we have made the following measurements:

    Frequency of FP operations = 25%

    Average CPI of FP operations = 4.0

    Average CPI of other instructions = 1.33

    Frequency of FPSQR= 2%

    CPI of FPSQR = 20

The Processor Performance Equation

- Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using the processor performance equation.

The Processor Performance Equation

- Answer:

  First, observe that only the CPI changes; the clock rate and instruction count remain identical. We start by finding the original CPI with neither enhancement:

$$\mathrm{CPI}_{\mathrm{original}} = \sum_{i=1}^{n}\left(\frac{\mathrm{IC}_i}{\text{Instruction count}} * \mathrm{CPI}_i\right)$$

The Processor Performance Equation

$$\mathrm{CPI}_{\mathrm{original}} = \sum_{i=1}^{n} \left(\frac{\mathrm{IC}_i}{\text{Instruction count}} * \mathrm{CPI}_i \right)$$

- (4 * 25%) + (1.33 * 75%) = 2.0 *(rounded)*

The Processor Performance Equation

- We can compute the CPI for the enhanced FPSQR by subtracting the cycles saved from the original CPI:

$$\mathrm{CPI}_{\text{with new FPSQR}} = \mathrm{CPI}_{\text{original}} - 2\% * \left(\mathrm{CPI}_{\text{old FPSQR}} - \mathrm{CPI}_{\text{new FPSQR only}}\right)$$

- (2.0 - 2% * (20 - 2)) = 1.64

The Processor Performance Equation

- We can compute the CPI for the enhancement of all FP instructions the same way or by summing the FP and non-FP CPIs. Using the latter gives us

  $CPI_{newFP} = (75\% \text{ x } 1.33) + (25\% \text{ x } 2.5) = 1.62$

The Processor Performance Equation

- Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better. Specifically, the speedup for the overall FP enhancement is

$$\text{Speedup}_{\text{newFP}} = \frac{\text{CPU Time}_{\text{original}}}{\text{CPU Time}_{\text{newFP}}}$$

The Processor Performance Equation

$$= \frac{\text{IC} * \text{Clock cycle} * \text{CPI}_{\text{original}}}{\text{IC} * \text{Clock cycle} * \text{CPI}_{\text{newFP}}}$$

$$= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{newFP}}} = \frac{2.00}{1.625} = \mathbf{1.23}$$

Happily, we obtained this same speedup using Amdahl's Law.

Reference(s):

- **Hennessy, J.L., Patterson, D.A**. Computer Architecture: A Quantitative Approach (4th Ed)