Computer Architecture

TECHNOLOGICAL TRENDS AND MEASURINGPERFORMANCE

PROFESSOR SAMIR ABOU EL-SEOUD

Objectives

1. Understand overall system performance.
2. Understand common methods of evaluating computer performance.
3. Be able to calculate how changes to one part of a computer system will affect overall performance

Technological Trends

- Since the early 1980s, computer performance has been driven by improvements in the capabilities of the integrated circuits used to implement microproceesors, memory chips and other computer components.
- Over time, integerated circuits improved in **density** (how many transistors and wires can be placed in a fixed area on a silicon chip), **speed** (how quickly basic logic gates and memory devices operate), and **area** (the physical size of the largest integerated circuit that can be fabricated).
- Computer architects have been able to take advantage of the increasing denisty of integrated circuits to add features to microprocessors and memory systems that improve performance over and above the improvements in speed of the underlying transistors.

Measuring Performance

- Generally, performance describes how quickly is a given system can execute a program or programs.
- Systems that execute programs in less time are said to have higher performance.
- The best measure of computer performance is the **execution time** of the program or programs that the user wants to execute, but it is generally impractical to test all the programs that will run on a given system before deciding which computer to purchase or when making design decisions.
- Instead, computer architects have come up with a variety of metrics to describe computer performance.

MIPS

- An early measure of computer performance was the rate at which a given machine executed instructions.
- This is calculated by dividing the number of instructions executed in running a program by the time required to run the program and is typically expressed in millions of instructions per second (MIPS).
- MIPS has fallen out of use as a measure of performance, mainly because it does not account for the fact that different systems often require different numbers of instruction to implement a given program.
- A computer's MIPS rating doesn't tell you anything about how many instructions it requires to perform a given task, making it less useful than other metrics for comparing the performance of different systems.

CPI/IPC

- Another metric used to describe computer performance is ***the number of clock cycles required to execute each instruction***, known as cycles per instruction, ***or CPI***.
- The CPI of a given program on a given system is calculated by dividing the number of clock cycles required to execute the program by the number of instruction executed in running the program.
- For the systems that can execute more than one instruction per cycle, ***the number of instructions executed per cycle, or IPC***, is often used instead of CPI.

CPI/IPC

- IPC is calculated by dividing the number of instructions executed in running a program by the number of clock cycles required to execute the program.
- These two metrics give the same information, and the choice of which one to use is generally made based on which of the values is greater than the number 1.
- When using IPC and CPI to compare systems, **it is important to remember that high IPC values indicate that the reference program took fewer cycles to execute than low IPC values, while high CPI values indicate that more cycles ere required than low CPI values.**
- Thus, a large IPC tends to indicate good performance, while a large CPI indicates poor performance.

CPI/IPC Example

- A given program consists of a 100-instruction loop that is executed 42 times. If it takes 16,000 cycles to execute the program on a given system, what are that system's CPI and IPC values for the program?

- Solution:

The 100-instruction loop is executed 42 times, so the total numbers of instructions executed is 100x42 = 4200. It takes 16,000 cycles to execute the program, so the CPI is 16,000/4200 = 3.81. To compute the IPC, we divide 4200 instructions by 16,000 cycles, getting an IPC of 0.26

Benchmark SUITES

- Both MIPS and CPI/IPC have limitations as measures of computer performance. *Benchmark suites* are a third measure of computer performance
- A benchmark suite consists of a set of programs that are believed to be typical of the programs that will be run on the system.
- A system's score on the benchmark suite is based on how long it takes the system to execute all of the programs in the suite.

Geometric versus Arithmetic mean

- Many benchmark suites use the geometric mean rather than the arithmetic mean to average the results of the programs contained in the benchmark suite.
- Using the geometric mean makes it harder for a system to achieve a high score on the benchmark suite by achieving good performance on just one of the programs in the suite, making the systems' overall score a better indicator of its performance on most programs.
- The geometric mean of n values is calculated by multiplying the n values together and taking the nth root of the product.
- The arithmetic mean, or average, of a set of values is calculated by adding all of the values together and dividing by the number of values.

Example

- What are the arithmetic and geometric means of the values 4,2,4,82?

  - Solution:

The arithmetic mean of this series is

$$\frac{4+2+4+82}{4}=23$$

The geometric mean is

$$\sqrt[4]{4\times 2\times 4\times 82}=7.16$$

Note that the inclusion of one extreme value in the series had a much greater effect on the arithmetic mean than on the geometric mean.

Speed up

- Computer architects often use the term speedup to describe how the performance of an architecture changes as different improvements are made to the architecture.
- Speedup is simply the ratio of the execution times before and after a change is made, so:

$$Speedup = \frac{Execution\ time\ _{before}}{Execution\ time\ _{after}}$$

For example, if a program takes 25 seconds to run on one version of an architecture and 15 seconds to run on a new version, the overall speedup is 25 seconds/15 seconds =1.67

Amdahl's law

- The most important rule for designing high performance computer systems is make the common case fast.
- Qualitatively, this means that the impact of a given performance improvement on overall performance is dependent on both how much the improvement improves performance when it is in use and how often the improvement is in use.
- Quantitatively, this rule has been expressed as Amdahl's law, which states

$$\text{Execution time}_{new} = \text{Execution time}_{old} \times \left[\text{Frac}_{unused} + \frac{Frac_{used}}{Speedup_{used}}\right]$$

In this equation, Frac_{unused} is the fraction of time that the improvement is not in use, Frac_{used} is the fraction of time that the improvement is in use, and Speedup_{used} is the speedup that occurs when the improvement is used (this would be the overall speedup if the improvement were in use at all times).

Note that Frac_{used} and Frac_{unused} are computed using the execution time before the modification is applied.

Amdahl's law

- Amdahl's Law can be rewritten using the definition of speedup to give:

$$\text{Speedup} = \frac{Execution\ time_{old}}{Execution\ time_{new}} = \frac{1}{Frac_{unused} + \frac{Frac_{used}}{Speedup_{used}}}$$

Amdahl's' law example

- Suppose that a given architecture that does not have a hardware support for multiplication, so multiplications have to be done through repeated addition. If it takes 200 cycles to perform a multiplication in software and 4 cycles to perform a multiplication in hardware.

  a) What's the overall speedup from hardware for multiplication if a program spends 10% of its time doing multiplications?

  b) What's the overall speedup from hardware for multiplication if a program spends 40% of its time doing multiplications?

Amdahl's' law example

Solution:

Speed up = 200/4 = 50 (ratio of time to do a multiplication without the hardware to time with the hardware).

Using Amdahl's law

a) $Frac_{used} = 0.1$

$Frac_{unused} = 1 - 0.1 = 0.9$

Speedup = 1 / [0.9 + (0.1/50)] = 1.11

b) $Frac_{used} = 0.4$

$Frac_{unused} = 1 - 0.4 = 0.6$

Speedup = 1 / [0.6 + (0.4/50)] = 1.64