EE 357 Unit 12

Performance Modeling

An Opening Question

- An Intel and a Sun/SPARC computer measure their respective rates of instruction execution on the same application written in C
  - Computer A achieves 160 MIPS (Millions of Instructions Per Second)
  - Computer B achieves 200 MIPS
- Which computer executes the program faster?
  - It depends on the instruction set and compiler (ultimately, the instruction count). Computer B and its compiler may use many more simpler (faster) instructions to implement the program thereby increasing its instruction execution rate but saying nothing of overall execution time

Another Question

- A Pentium 3 has a clock rate of 1 GHz while a Pentium 4 has a clock rate of 2 GHz.
  - They implement the same instruction set
  - They are tested on the same executable program.
- Is the Pentium 4 twice as fast as the Pentium 3?
  - Since they both use the same instructions and the same instruction count (same executable), we may think that the Pentium 4 would be twice as fast
  - However, the microarchitectural implementation of the processor may mean that the Pentium 3 executes instructions in 2 clocks on average while the Pentium 4 executes instruction in 4 clocks on average thus making the execution time exactly the same.

Execution Time

- Execution time is the only valid metric for comparing performance
- Two possible performance goals
  - Execution time: Measured for a single program's execution
  - Throughput: Total jobs performed per unit time

Wall Clock Time vs. CPU Time

- Even execution time can be hard to measure accurately because the OS may allocate a percentage of compute cycles to other programs (also, part of a programs execution is spent in OS calls for I/O, etc.)
  - Wall Clock Time: Real time it took from when the user submitted the job until it was completed
  - CPU Time: Actual time the program took to execute when it was running

Performance

- Performance is defined as the inverse of execution time

$$\text{Performance} = \frac{1}{\text{Execution Time}}$$

- Often want to compare relative performance or speedup (how many times faster is a new system than an old one)

$$\text{Speedup} = \frac{\text{Performance}_{\text{New}}}{\text{Performance}_{\text{Old}}} = \frac{\text{Execution}_{\text{Old}}}{\text{Execution}_{\text{New}}}$$

Performance Equation

- Execution time can be modeled using three components
  - Instruction Count: Total instructions executed by the program
  - Clocks Per Instruction (CPI): Average number of clock cycles to execute each instruction
  - Cycle Time: Clock period (1 / Freq.)

$$\text{Exec. Time} = \text{Instruc. Count} * \frac{\text{Clocks}}{\text{Instruction}} * \frac{\text{Time}}{\text{Clock}}$$

$$= \text{Instruc. Count} * \text{CPI} * \text{Cycle Time}$$

Example

- Processor A runs at 200 MHz and executes a 40 million instruction program at a sustained 50 MIPS
- Processor B runs at 400 MHz and executes the same program (w/ a different compiler) which yields a count of 60 million instructions and a CPI of 6
- What is the CPI of the program on Proc. A?
- Which processor executes the program faster and by what factor?
- What is the MIPS rate of Proc. B?

$$CPI_A = \frac{200*10^6\,cycles}{second} * \frac{second}{50*10^6\,instrucs}$$

$$ExecTime_A = 40*10^6\,instrucs.* \frac{second}{50*10^6\,instrucs.} = 0.8\,\text{sec}$$

$$ExecTime_B = 60*10^6\,instrucs.* \frac{6cycles}{instruc.} * \frac{second}{400*10^6\,cycles} = 0.9\,\text{sec}$$

$$Speedup = \frac{ExecTime_B}{ExecTime_A} = \frac{0.9}{0.8} = 1.125$$

$$MIPS_B = \frac{60*10^6\,instrucs}{0.9 seconds} = 66.67 MIPS$$

What Affects Performance

Component	SW/HW	Affects	Description
Algorithm	SW	Instruc. Count & CPI	Determines how many instructions & which kind are executed
Programming Language	SW	Instruc. Count & CPI	Determines constructs that need to be translated and the kind of instructions
Compiler	SW	Instruc. Count & CPI	Efficiency of translation affects how many and which instructions are used
Instruction Set	HW	Instruc. Count, CPI, Clock Cycle	Determines what instructions are available and what work each instruction performs
Microarchitecture	HW	CPI, Clock Cycle	Determines how each instruction is executed (CPI, clock period)

Source: H&P, Computer Organization & Design, 3rd Ed.

Calculating CPI

- CPI can be found by taking the expected value (weighted average) of each instruction type's CPI [i.e. CPI for each type * frequency (probability) of that type of instruction]

$$CPI = \sum_i CPI_{Type\_i} * P(InstructionType_i)$$

- In practice, CPI is often hard to find analytically because in modern processors instruction execution is dependent on earlier instructions
  - Instead we run benchmark applications on simulators to measure average CPI.

CPI vs. IPC

- The reciprocal of CPI is IPC (Instructions per Cycle)
- Modern processors have the ability to execute more than one instruction simultaneously (superscalar)
- In the case of a 2-way superscalar, the maximum performance would be 2 instructions per clock cycle yielding a CPI of 0.5
- Thus, CPI is often inverted to IPC (max IPC = 2 instructions per cycle for the 2-way superscalar)

$$\text{Exec. Time} = \text{Instruc. Count} * \text{CPI} * \text{Cycle Time}$$

$$= \text{Instruc. Count} * \frac{1}{\text{IPC}} * \text{Cycle Time}$$

Other Performance Measures

- OPS/FLOPS = (Floating-Point) Operations/Sec.
  - Maximum number of arithmetic operations per second the processor can achieve
  - Example: 4 FP ALU's on a processor running @ 2 GHz => 8 GFLOPS
- Memory Bandwidth (Bytes/Sec.)
  - Maximum bytes of memory per second that can be read/written
- Programs are either memory bound or computationally bound

Amdahl's Law

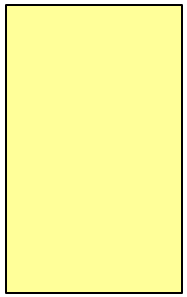
- Where should we put our effort when trying to enhance performance of a program
- Amdahl's Law = How much performance gain do we get by improving only a part of the whole

$$ExecTimeNew = ExecTimeUnaffected + \frac{ExecTimeAffected}{ImprovementFactor}$$

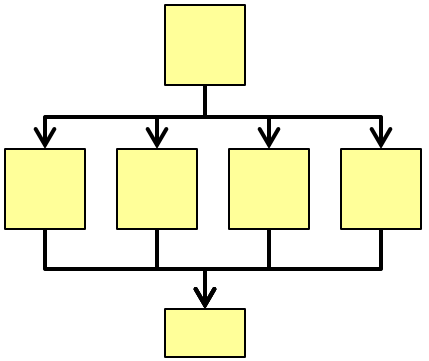
$$Speedup = \frac{ExecTimeOld}{ExecTimeNew} = \frac{1}{Percent_{Unaffected} + \frac{Percent_{Affected}}{ImprovementFactor}}$$

Amdahl's Law

- Holds for both HW and SW
  - HW: Which instructions should we make fast? The most used (executed) ones
  - SW: Which portions of our program should we work to optimize
- Holds for parallelization of algorithms (converting code to run multiple processors)



Original Sequential Program



Parallelized Program

Amdahl's Law Example

- A program consists of a single function with a loop. The loop body executes 10 times and consists of 5 instructions. The rest of the function consists of 50 instructions. Assume all instructions take the same amount of time to execute.
- If we could somehow remove the 50 sequential instructions altogether, how much faster will our program run

$$Speedup = \frac{1}{Percent_{Unaffected} + \frac{Percent_{Affected}}{ImprovementFactor}}$$

$$Speedup = \frac{1}{0.5 + 0} = 2$$

Parallelization Example

- A programmer is parallelizing her code to run on an 8 core system.
  - 40% of the original program will still need to be executed sequentially
  - Another 40% of the code can be parallelized into only 4 independent threads (thread = execution stream of a core)
  - The remaining 20% of the code can be fully parallelized to use all 8 cores.
- What speedup will be achieved assuming all other factors are equal (clock speed, etc.)?

$$Speedup = \frac{1}{0.4 + \frac{0.4}{4} + \frac{0.2}{8}} = \frac{1}{0.525} = 1.905$$