EE 457 Unit 4

Computer System Performance

Motivation

- An individual user wants to:
  - Minimize single program execution time
- A datacenter owner wants to:
  - Maximize number of ___________________________
  - Minimize _______ (______________________)

http://e-telligentinternetmarketing.com/website/frustrated-computer-user-2/
http://www.intomobile.com/2010/11/02/opera-iceland-clean/

Performance Depends on View Point?!

- What's faster:
  - A 777 passenger airliner
  - An F-22 fighter jet
- If you are an individual interested in getting from point A to point B, then the F-22
  - This is known as ______________________
  - Time from the _____ of an operation until it ______
- If you are trying to ________________ number of people, the _______________
  - This is known as ______________________________

Throughput vs. Latency

- If Latency is the Time it takes for a Job to complete & Throughput = Jobs / Time…
- …Is Throughput = 1 / Latency?
  - ________________
  - Latency is from the perspective of a ___________
  - Throughput is from the perspective of ___________
  - ___________ is the great friend of throughput!
- We will see many times in this course (pipelining, memory org., etc.) that there is often not much we can do about ______ but there are lots of ways to improve ________
  - Hopefully without _________ latency too much, if at all

Metrics

- What are the metrics?

Execution Time

- Key Point: When comparing different systems, ______________________ is the ultimate criterion (metric)
- Using a _____ as a metric can often be misleading metrics
  - Often not comparing apples to apples
  - Often not normalized

What's Wrong with Rates

- Two trains take two different routes from City A to City B and leave at the same time. Train 1 travels at 60 MPH, while train 2 travels at 75 MPH. Which one arrives first?
- ____
- Example 1 (MIPS):
  - You may hear that Computer 1 executes 500 MIPS while Computer 2 executes 750 MIPS. Which one executes a given program faster?
  - Train speed = MIPS & Routes = Program (how many instructions)
  - MIPS is only useful for the same ______________________________
- Example 2 (Clock Rate):
  - You may hear that CPU1 runs at 2 GHz and CPU2 runs at 3 GHz, which one executes a program faster (assume same instruction set)
  - CPU1 may have CPI=___ while CPU2 has CPI=___
  - CPU1 Time = ________ < CPU2 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 or waiting for I/O, etc.)
  - Wall Clock Time: Real time it took from when the user submitted the job until it was completed
  - CPU Time (User Time + System Time): Actual time the program used the CPU either in the application code (User Time) or in the OS (System Time)
    - Doesn't include I/O time
  - Linux/Unix: `$time` *`executable`*
    - real 0m16.019s
    - user 0m12.840s
    - sys 0m0.180s

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
  - ____
    - IC = Dynamic Instruction Count not Static Instruction Count
  - ________________________: Average number of clock cycles to execute each instruction
  - ________________________

$$\text{Exec. Time} =$$

$$=$$

Dynamic vs. Static Instruction Count

- Static instruction count is the number of written instructions
- Dynamic instruction count (or "trace" count) is how many instruction were executed at run time
- Would you prefer either:
  - Small Static IC & Large Dynamic IC … or …
  - Large Static IC & Small Dynamic IC

Static IC

Dynamic IC

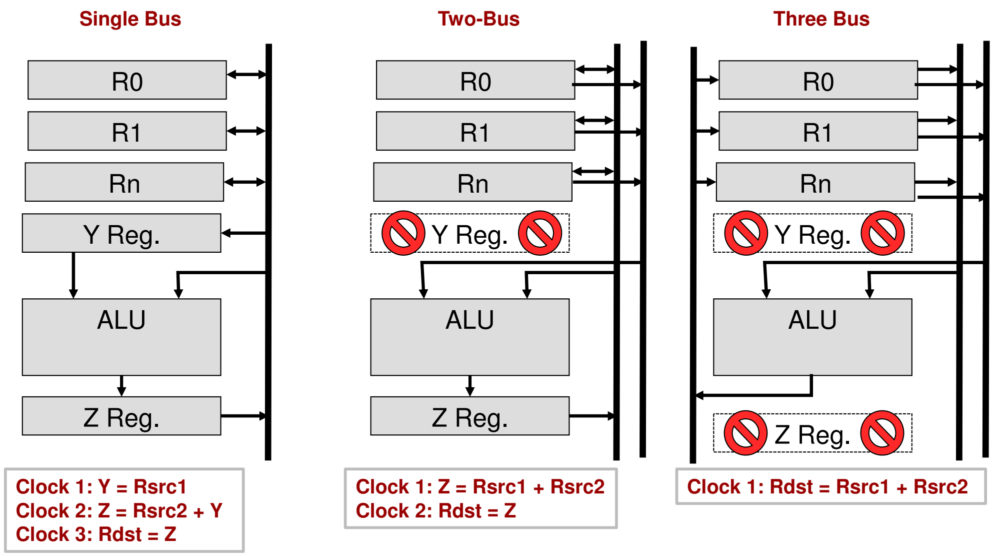
```
    ----
LP: ----
    ----
    ----
    BNE LP
    ----
THN:----
    ----
ELS:----
    ----
```


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.

CPI & Microarchitecture



General Implications: Less Resources => More Clock Cycles (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?

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

Example

Instruction Type	CPI P1
A	1
B	2
C	3

If CLK=1 MHz what is PEAK Inst./Sec. = ________

Average CPI = ____________ = __________

Instruction Type	CPI P1	
A	1	
B	2	
C	3	

Average CPI =

Example

- Calculate CPI of this snippet of code using the following CPI's for each instruction type

```
        add     $s0,$zero,$zero
        addi    $t1,$zero,4
loop:   lw      $t2,0($t0)
        add     $t2,$t2,$t1
        addi    $t0,$t0,4
        addi    $t1,$t1,-1
        bne     $t1,$zero,loop
        sw      $t2,0($t2)
```

Instruction Type	CPI
add / addi	1
lw / sw	4
bne	2

Dynamic Instruction Count = __________

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

Instruction Type	Dynamic Count
add	
lw / sw	
bne	

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
- ___________________________________

Energy Proportional Computing

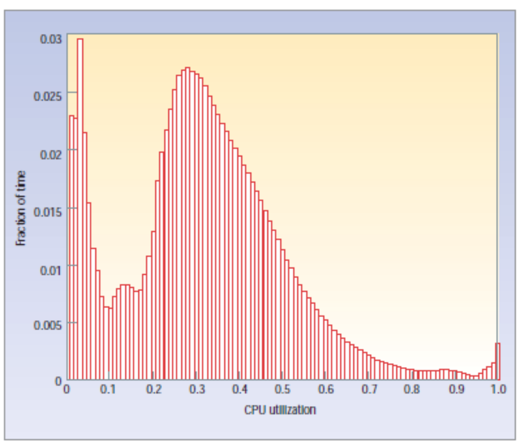
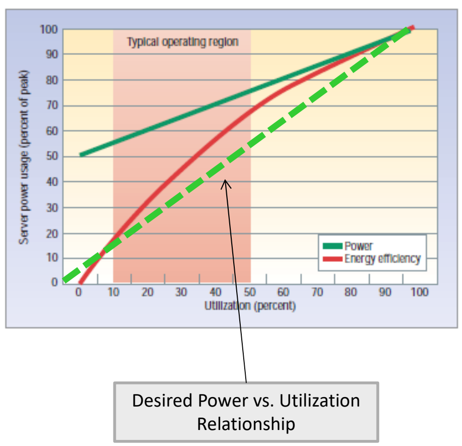


Figure 1. Average CPU utilization of more than 5,000 servers during a six-month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.



"The Case for Energy-Proportional Computing", Luiz André Barroso, Urs Hölzle, *IEEE Computer*, vol. 40 (2007).

What should I optimize?

AMDAHL'S LAW

Amdahl's Law

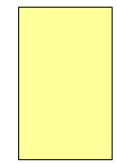
- Where should we put our effort when trying to enhance performance of a program
- Amdahl's Law = How much performance gain do we get by improving only a part of the whole

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

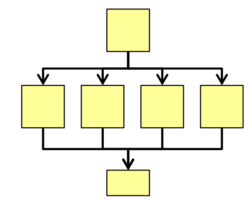
$$Speedup = \frac{ExecTimeOld}{ExecTimeNew} =$$

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

Parallelization Example

- A programmer parallelizes a function in his program to be run on 8 cores. The function accounted for 40% of the runtime of the overall program. What is the speedup of the enhancement?

$$Speedup \quad =$$

Example

- What if we improve only class B instrucs.

Instruction Type	CPI P1	Freq.
A	1	30%
B	2 => 1	20%
C	3	50%

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

$$Speedup = \frac{1}{2/3 + (1/3/2)} = \frac{1}{5/6} = 6/5 = 1.2$$

???

Profiling

- How do you know where time is being spent?
- From a software (programming for performance) perspective, profilers are handy tools
  - Instrument your code to take statistics as it runs and then can show you what percentage of time each function or even line of code was responsible for
  - Common profilers
    - 'gprof' (usually standard with Unix / Linux installs) and 'g++'
    - Intel VTune
    - MS Visual Studio Profiling Tools
- From a hardware perspective, simulators can help
  - SimpleScalar
  - Simics
  - Your own simulation model developed in Verilog/SystemC/etc.

gprof Output

```
 %   cumulative   self              self     total
time   seconds   seconds    calls   s/call   s/call  name
42.96      4.48     4.48 56091649     0.00     0.00  Board::operator<(Board const&) const
 6.43      5.15     0.67  2209524     0.00     0.00  std::_Rb_tree<...>::_M_lower_bound(...)
 5.08      5.68     0.53 108211500     0.00     0.00  __gnu_cxx::__normal_iterator<...>::operator+(...)
 4.51      6.15     0.47  4419052     0.00     0.00  Board::Board(Board const&)
 4.32      6.60     0.45  1500793     0.00     0.00  void std::__adjust_heap<...>(...)
 3.84      7.00     0.40 28553646     0.00     0.00  PuzzleMove::operator>(PuzzleMove const&) const
```


Credits

- These slides were derived from Gandhi Puvvada's EE 457 Class Notes