ECSE 425 – Tutorial 2

Performance and Benchmarks

Performance

- Which of those cars has the best performance?

Credit: {Brian Snelson, IFCAR}, Wikipedia

Performance

- How do you measure performance?
  - Throughput : number of operations/sec?
  - Latency : time before a result is obtained?
  - Execution time : wall-clock time taken to perform a task?
- Different types of users expect different things
  - When measuring performance, must take those different needs into account
    - Database? Web server? Rendering farm?

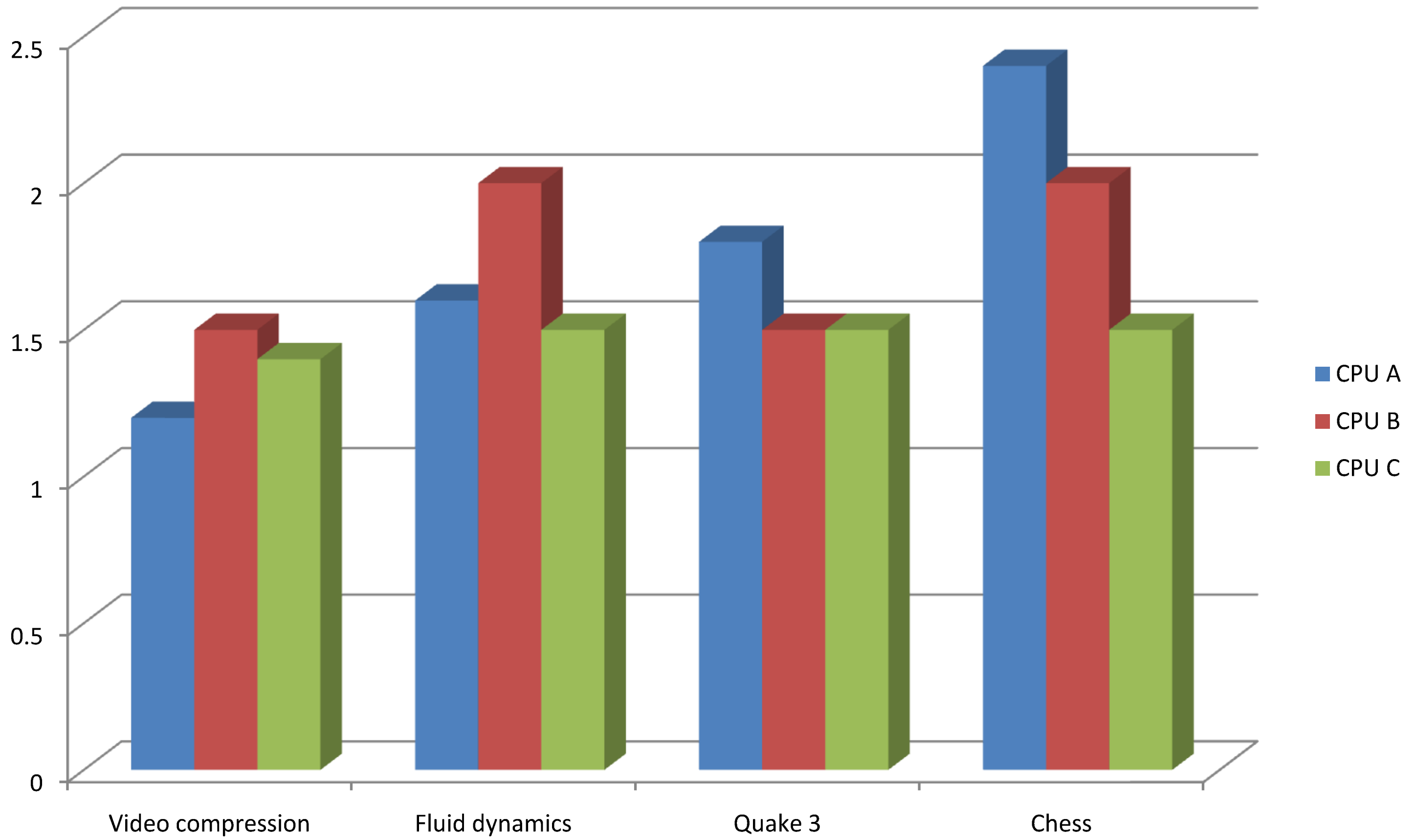
Performance

- How do you measure performance?
  - **Benchmarks!**

  - Good benchmark suites perform calculations using real-life programs, across a wide variety of domains
    - Video compression, games, …
    - XML parsing, linear algebra, …
    - Speech recognition, molecular dynamics, …
    - …

Performance

- Sample benchmark results



Performance

<u>Benchmark scores</u>

- Often need to compute an average score
  - If the scores are expressed as the execution time, we use the arithmetic mean:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} score_i$$

  - If the scores are performance ratios, we use the geometric mean:

$$\mu = \sqrt[n]{\prod_{i=1}^{n} ratio_i}$$

Performance

<u>Benchmark scores</u>

- What is the mean score μ of a benchmark suite where relative scores with respect to a reference machine are (ratio = t_{new}/t_{ref}):
  - $ratio_1=4$, $ratio_2=2$, $ratio_3=8$

- What is the mean execution time μ of a benchmark suite where execution times are :
  - $t_1=4s$, $t_2=2s$, $t_3=8s$

Amdahl's Law

- How can you determine how much better a program performs given improvements in certain parts of this program?
  - This metric is known as the *speedup*

- Amdahl's law!
  - Speedup : $n = ExecutionTime_{old} / ExecutionTime_{new}$

Amdahl's Law

System with single enhancement:

$$\texttt{time}_{new} = (1 - f_{enh})time_{old} + \frac{f_{enh}time_{old}}{s}$$

$$systemspeedup = \frac{time_{old}}{time_{new}} = \frac{1}{(1-f_{enh})+\frac{f_{enh}}{s}}$$

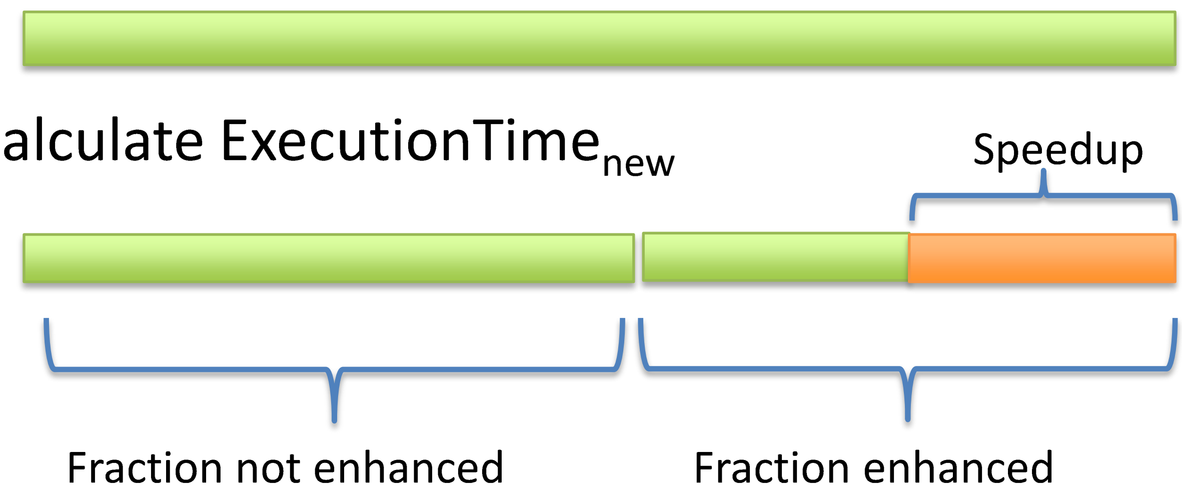
System with multiple enhancements:

$$\texttt{time}_{new} = \frac{f_1 time_{old}}{s_1} + \frac{f_2 time_{old}}{s_2}$$

$$systemspeedup = \frac{time_{old}}{time_{new}} = \left(\frac{f_1}{s_1} + \frac{f_2}{s_2}\right)^{-1}$$

Amdahl's Law

- Simpler method:
  - $n = ExecutionTime_{old} / ExecutionTime_{new}$
  - Assume $ExecutionTime_{old} = 1$
  - Calculate $ExecutionTime_{new}$



Amdahl's Law

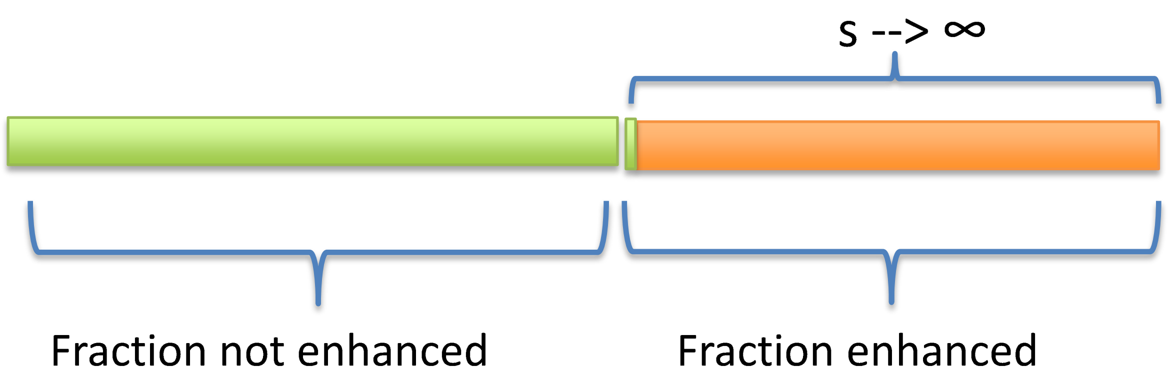
- Assume a processor spends 90% of its time performing integer arithmetic and 10% doing floating point calculations.

- If it were possible to speed-up integer calculations by 12% (s=1.136) OR floating point calculations by 85% (s=6.666), which one should you choose?

Amdahl's Law

- What is the best speedup achievable for a given f_{enh}:

$$\lim_{s\to\infty}\left(\frac{1}{(1-f_{enh})+\frac{f_{enh}}{s}}\right)=\frac{1}{1-f_{enh}}$$



Amdahl's Law

- Assume a processor spends 90% of its time performing integer arithmetic and 10% doing floating point calculations.

- What is the best theoretical speed-up that you can obtain by improving the performance of the floating point unit infinitely?

Processor Performance

- CPU time = IC* CPI * t_{cc}

- IC (instruction count) = number of instructions in your program
- CPI (cycles per instruction) = number of clock cycles needed to perform an instruction
- CC (clock cycle time) = time taken by a single clock cycle

- <u>Those metrics are tied to one another!</u>

Processor Performance

- Another way to view speedups:
  - CPU time = IC* CPI * t_{cc}
  - s = $Time_{old}$ / $Time_{new}$

    = IC_{old}* CPI_{old} * t_{cc_old} / IC_{new}* CPI_{new} * t_{cc_new}

Processor Performance

Assume a CPU with running a program:

– IC=20M, CPI=2.6, t_{cc}=1ns

Suppose the following scenarios:

– Decrease t_{cc} to 0.9ns and increase CPI to 2.7

– Increase t_{cc} to 1.1ns and decrease CPI to 2.1

What is the best alternative?

Processor Performance

Assume a CPU and workload with the following characteristics:

– 70% integer operations : 2 CC/instr. avg.

– 15% fp operations : 10 CC/instr. avg.

– 14.9% branches : 1 CC/instr. avg.

– 0.1% : SHA1 operation : 70 CC/instr

What is the average CPI for this CPU?

Is it better to improve CPI_{fp} to 9CC or CPI_{SHA1} to 20 CC?