David Merrick
Heather Warman
May 24, 2014

Assignment 6

6.1: What is performance in the context of computer systems and why is it so difficult to define?

Few people seem to agree on how to measure performance or how to interpret the results of testing it. The dictionary definition of performance is how successful something is at a certain task. In the context of computers, this usually refers to how fast the computer can perform a task. This, however, can be nebulous. The book gave an example of a desktop computer that can very quickly execute any program run on it. It compared this to an iPad with a much less powerful processor, but that could boot up faster, and noted that when using a simple calendar application the iPad usurped the raw processing power of the desktop by it's speedier boot time. In short, performance is nebulous and really depends on the application of the computer. In some cases, like the iPad calendar example, a slower computer in terms of raw processing power may be able to outperform one with much greater processing power if performance is defined as the speed of doing something for the end user.

6.2: A system consists of a CPU, cache memory, main store, and hard disk drive. Where are time and effort best spent improving the system's performance? What factors affect your answer?

It depends on the application. If the user needs raw processing power, obviously the improvements should be made to the CPU. If the user needs a faster boot time, then the hard disk drive should be improved to read data more quickly. The factors that affect the answer to that question are cost of improvements and what the application is for the computer.

6.4: A data transmission system transmits data in the form of a master frame containing 16 sub-frames. Each sub-frame includes a 1024-bit data word and a 12-bit error-correcting code. The master frame itself contains a 32-bit error correcting code. What is the efficiency of this system?

$Efficiency = (total\ time\ spent\ executing\ useful\ work)/(total\ time) = (optimal\ time)/(total\ time)$

Master frame:

16 sub-frames	32-bit error correcting code

Sub frame:

1024-bit word	12-bit error correcting code

Total bits: 1024 * 16 + 32 = 16,416
Bits for error correcting code = 12 * 16 + 32 = 224

$Efficiency = bits\ for\ useful\ work/total\ bits = (16,416 - 224)/16,416 = .986$

6.6: Why is clock rate a poor metric of computer performance? What are the relative strengths and weaknesses of clock speed as a performance metric?

Clock rate is a poor metric for performance because it is only applicable in certain situations and largely meaningless otherwise. It can, for example, be useful in comparing two models of CPUs from the same generation, such as the Intel i7-980 vs the 970. But it really only measures the oscillation speed, which isn't necessarily the speed at which the processor executes instructions or performs operations. For this reason, it doesn't mean very much when comparing different CPUs.

6.7: Math Problem

a. 150ns
b. 120ns
c. 120ns
d. 100ns
e. 80ns

6.9: Can you think of a better metric than MIPS or clock speeds that gives a good impression of the power of a processor (without having to use benchmarks).

According to the book, there are certain attributes that a good metric should have. A good metric should be repeatable (always yielding the same result under different conditions), easy to measure, universal across CPU architectures, general, and independent of commercial influences. A benchmark is the time required to execute a task or the rate at which tasks are executed. MIPS and clock speed are unreliable benchmarks. Clock rate is a poor indicator of performance because there is no simple relationship between clock rate and system performance across different platforms. MIPS is a poor indicator of performance because it only shows how fast a computer executes instructions, not the meaningful work done by those instructions. If using benchmarks, MFLOPS is more reliable than clock speed or MIPS because it indicates the actual work done rather than instruction throughput. The better indicator of performance is using fine-grained benchmarks; testing specific fragments of code across different platforms. The benchmark should match the workload being evaluated.

6.11: Overclocking a computer means operating it at a higher clock rate than that specified by its manufacturer; for example, a 2 GHz chip might be clocked at 2.1 GHz to squeeze more performance out of it. Does overclocking disprove the famous aphorism "There's no such thing as a free lunch," or is there a hidden cost? If so, what is the cost of overclocking?

There are definitely costs to overclocking a system. One of the main costs is that the lifespan of the hardware can be considerably reduced by the increased voltage necessary to overclock, and the large amount of heat that is produced. So you may gain more performance but risk the stability of the system.

6.12: If the clock rate could be reduced by 15%, it would require only 2 cycles to perform a register load. Would that be a good idea?

It might be a good idea if the frequency for register load operations were higher. Since it is only at 20%, they don't occur as often as the all branch instructions or the arithmetic/logical instructions. If we lower the clock rate to improve one thing, it might have a negative effect on another thing, and since arithmetic/logical instructions have a higher frequency and only take 1 cycle, it might be a bad idea to lower the clock rate and potentially increase the needed cycles for that to 2 cycles, which would make the performance much worse since those happen much more often.

6.13: If the average performance of the computer (in terms of its CPI) is to be increased by 20% while executing the same instruction mix, what target must be achieved for the cycles per conditional branch instruction?

Done -- Picture

6.14: A program is run on a computer with the following parameters.
Clock cycle time 10 ns
Instructions with 1 cycle 70%
Instructions with 2 cycles 20%
Instructions with 3 cycles 10%
What is the MIPS rating of this computer?

Done -- Picture

6.16: In a particular system, a CPU is used for 78% of the time and a disk drive for 22% of the time. A designer has two options:
a. Improve the disc performance by 40% and the CPU performance by 20%
b. Improve the disc performance by 10% and the CPU performance by 80%

Done -- Picture

6.17: For the following systems that have both serial and
parallel activities, calculate the speedup ratio.
a. 10 processors $f_s = 0.1$
b. 100 processors $f_s = 0.1$
c. 5 processors $f_s = 0.4$
d. 100 processors $f_s = 0.01$

Done -- Picture

6.18: A system has a single core processor that costs $150. Suppose that adding more cores to the chip costs $10 per additional processor. (Note: For this system, the value of $f_s = 0.1$ .If it is considered worthwhile adding cores until the incremental speedup ratio increases by less than 5% over the original (unmodified) performance, what is the optimum number of processors? What percentage increase in cost is required to achieve this performance?

Done -- Picture

6.22: A coprocessor is added to a computer to speed the execution time of string-processing instructions by a factor of 3.5. What fraction of the execution time must use these string-processing instructions in order to achieve an average speedup of 1.5?

Done -- Picture

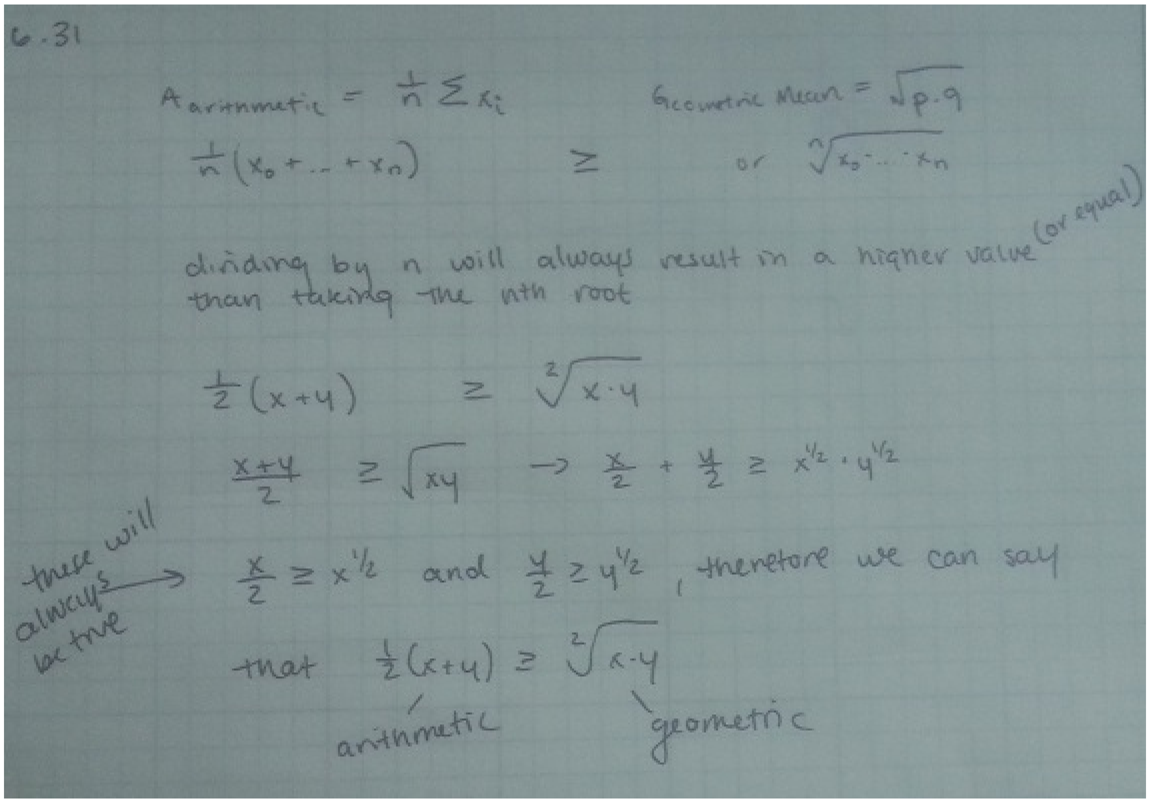
6.25: Math problem

No, this answer is not necessarily correct. It is wrong to assume that just because you have two disks operating in parallel, that the speed will be twice as fast. This doesn't account for operations that may not be able to be parallelized or perhaps are throttled at a certain speed.

6.26: Someone decided to use the following C code as part of a benchmark to determine the performance of a computer including its memory. It has two potential faults. What are they?

The first is that the x variable is never used. A smarter processor could have branch logic to realize this and just skip over that entire loop. This would make the code execute faster, but would not be an accurate indicator of the power of the CPU or the memory. The second is that the inner loop accesses an array. If this array is in the CPU cache already, the CPU will never fetch from memory. Meaning that the benchmark of the memory performance would be inaccurate.

6.31: For two benchmarks, x and y, show that their arithmetic mean is always higher than, or the same as, the geometric mean.



6.31

$A_{arithmetic} = \frac{1}{n}\sum x_i$ Geometric Mean = $\sqrt{p\cdot q}$

$\frac{1}{n}(x_0 + \dots + x_n) \geq$ or $\sqrt[n]{x_0 \cdot \dots \cdot x_n}$

dividing by n will always result in a higher value (or equal) than taking the nth root

$$\frac{1}{2}(x+y) \geq \sqrt[2]{x \cdot y}$$

$$\frac{x+y}{2} \geq \sqrt{xy} \rightarrow \frac{x}{2} + \frac{y}{2} \geq x^{1/2} \cdot y^{1/2}$$

these will always be true $\rightarrow$ $\frac{x}{2} \geq x^{1/2}$ and $\frac{y}{2} \geq y^{1/2}$, therefore we can say

that $\underbrace{\frac{1}{2}(x+y)}_{\text{arithmetic}} \geq \underbrace{\sqrt[2]{x \cdot y}}_{\text{geometric}}$