Drake Bridgewater & Ryan Phillips
May 31, 2014
CS 472 HW 6

(6.1)

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

> Computer performance is difficult to define and is dependent on the slowest component. As our text book explains "computer's performance are used by the designer to improve the computers architecture and organization by locating bottlenecks and eliminating them or minimizing their effects." [1]

(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?

> Two answers but both very good:
>
> The answer really depends on what kind of performance you are wanting to improve. An example of this is the latest push for SSDs as a better technology for harddrives. This does relatively little in increasing the performance of a lot of classic computing scenarios that are CPU or GPU bound. But for the average consumer's workflow in which many different programs are loaded in and out of memory, the time the user spends waiting to start working is important. So then a new goal could be to try to eliminate any noticeable wait times for basic tasks. And the truth of the matter is that each of the components in a system play a part in accomplishing this, but whatever the current bottleneck is, that is where the focus should be. And this is a constantly shifting target.
>
> The area engineers typically focus on for improving overall system performance is the slowest component and still today the slowest component is the HDD. The HDD has been replaced by SSD which are non-mechanical components allowing the system to be faster but does not allow for huge capacity storage. Therefore, computers are given this artificial appearance of speed by intelligent management of hardware that has unparalleled speed.

(6.4)

A data transmission system transmits data in the 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?

First let's assume that by efficiency we are referring to the ratio of actual data to total data transferred.

```
actual data = 1024 * 16 = 16384 bits
data needed for error correction =
16*12 + 32 = 224 bits

efficiency = 16384 / (16384+224) ~= .987

About 98.7 percent efficient.
```

(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?

Because it's too simple. Clock rate does affect computer performance, but it's merely one of many factors.

Strengths: All other factors being equal (i.e. within the same family and architecture) clockspeed can help you determine the relative performance. And in the purely theoretical/abstract case, there is a direct connection between clockspeed and computing power.

Weaknesses: In real-world scenarios, increasing the speed of the clock means that many operations that used to be able to fit in one clock cycle might now take more than one cycle, negating the performance improvements. This is because "some actions such as accessing memory require a minimum amount of time". [1]

(6.7)

The timing diagram in Figure P6.7 illustrates a system in which operations occur as three consecutive clock cycles. Actions taking place in clock cycle 1 are scalalbe; that is, if the clock cycle time chances, the actions can be speeded up or slowed down correspondingly. In cycle 2, the action process 1 requires 25 ns and in clock cycle 3 the action process 2 requires 32 ns. If the clock cycle is less than the time required for process 1 or processes 2, then one ore more wait cycle have to be inserted for the process to complete.

What is the time to complete an operation if the clock cycle time is:

a 50 ns

$50 * 3 = 150$ ns

b 40 ns

$40 * 3 = 120$ ns

c 30 ns

$30 + 30 + 60 = 120$ ns

d 20 ns

$20 + 40 + 40 = 100$ ns

e 10 ns

$10 + 30 + 40 = 80$ ns

(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)?

> The problem with MIPS is that you don't know "what exactly was achieved by an instruction being executed"; [1] the complexity of instructions varies.
>
> So then we could move up one level of abstraction, and try to come up with a standard which takes these differences into account. You would have an increased number of basic tasks which will be performed by some number of MIPS. For example, table 6.2 in the book compares the instructions needed to perform an expression on two hypothetical computers. Well, the number of times this expression could be executed would become the new metric. This does start to come close to what many benchmarks do, but the 'operations' are much simpler. The problem then is deciding how to weight these operations, and which operations to include in your test set.

(6.11)

Overlclocking a computer means operating at 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?

> One of the main downsides of overclocking is that is may cause some systems to become more unstable. Additionally, a system is designed to dissipate a certain maximum amount of heat. Overclocking effectively increases the amount of heat produced, meaning that the system is more likely to overheat (if the cooling system isn't improved).

(6.12)

The following figures define the typical operating parameters of a processor.

Operation	Frequency	Cycles
Arithmatic/logical instructions	45%	1
Register load operations	20%	3
Register store operations	10%	2
All branch instructions	25%	2

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?

CPU time(v1):
$$= (\#instructions) \times ((0.45 * 1) + (0.20 * 3) + (0.10 * 2) + (0.25 * 2)) \times (clock)$$
$$= 1 \times (1.75) \times clock$$

CPU time(v2):
$$= (\#instructions) \times ((0.45 * 1) + (0.20 * 2) + (0.10 * 2) + (0.25 * 2)) \times (clock \times 1.15)$$
$$= 1 \times (1.55) \times (1.15 \times clock)$$
$$= 1 \times (1.78) \times clock$$

v2 is $\frac{1.78}{1.75} = 2\%$ slower

(6.13)

A computer has the following parameters:

Operation	Frequency	Cycles
Arithmatic/logical instructions	65%	1
Register load operations	10%	5
Register store operations	5%	2
Conditional branch instructions	20%	8

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?

```
.65*1 + .1*5 + .5*2 * .2*8 = 2.75 (base CPI)

We want to increase this metric by 25%,
which would correspond to a CPI of 2.2 (2.87*.8)

2.2 = .65*1 + .1*5 + .5*2 * .2*x

x = 5.25

If cycles per conditional branch instruction could hit 5
instead of 8, we would achieve the peformance goal.
```

(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?

Using 100 for n:
$$100/(140 * 10^-9 * 10^6) \ = 715 MIPS$$

(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%

    Which is the better option, and why?

> Option b. Since the CPU is more heavily used that the disk drive, improvements to the CPU carry more weight. In this particular example option b increases the overall performance by over twice that of option a. This is about 65% improvement vs. 30% improvement.

(6.17)

For the following systems that have both serial and parallel activies, calculate the speedup ratio.

> **a** 10 processors, $f_s = 0.1$
> $S \approx 5.26$
>
> **b** 100 processors, $f_s = 0.1$
> $S \approx 9.17$
>
> **c** 5 processors, $f_s = 0.4$
> $S \approx 1.92$
>
> **d** 100 processors, $f_s = 0.01$
> $S \approx 50.25$

(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 is 0.10).

If it is considered worthwhile adding cores until the incremental speedup ratio increases by less than 5% over the orignal (unmodified) performance, what is the optimum number of processors? What percentage increase in cost is required to achieve this performance?

> The difference between 33 and 34 processors is only 4.99%. So according to the specs, 33 is the optimal number of processors.
>
> This corresponds to a total increase in cost of 313% or an incremental change of 6.66% (from 32 to 33 processors).

(6.22)

A coprocessor is added to a computer to speed up 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?

```
x = non-string fraction
s = string-processing instructions fraction
x + s = 1
x*1 + s*3.5 = 1.5
x = 1 - s
1-s + 3.5*2 = 1.5
s = 0.5/2.5 = 0.2

20% of the execution time must use these string-proccessing instructions.
```

(6.25)

A computer spends 25% of its time accessing a hard disk. It spends 20% of the time doing floating point. The hard disk is replaced by two disks operating in parallel and the floating point unit is replaced by one four times faster. The speed up is given by Amdahl's law.

So, the speedup for the disk is S_{disk} - 2/2(2 * 0.75 + 1 - 0.75) = 1/1.75 = 1.429

The speedup for the floating-point unit is $S_{floating-point}$ = 4/(f * 0.80 + 1 - 0.80) = 4/1 = 4/3.4 = 1.176.

The total speedup ratio is the product o the individual speedups which is 1.429 * 1.176 = 1.681.

Is this answer correct?

> No, the total speedup ratio should not be calculated as a product. Assuming the two time fractions given in the problem are independent, the overall speedup would have to weight the increases provided by the two speedup ratios by the fractions. This gives a total speedup ratio less than either of the two speedups, 1.117. The original answer would only make sense if the system was somehow taking advantage of both speedups simultaneously, and then it would only apply at that exact moment.

(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?

```c
for (i = 0; i < 100; i++){
    p = q * s + 12345
    x = 0.0;
    for (j = 0; j < 60000; j++){
     x = x + A[j] * B[j];
    }
}
```

> To test performance the code would need to read and write to each level of cache including the main memory and this may or may not be enough to warrant accessing main memory. On top of ensuring main memory is being accessed you also should attempt to have enough instructions that the instruction cache is not sufficient.

(6.31)

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

The Arithmetric mean is defined as

$$A_{Arithmetric} = \frac{1}{n}\Sigma_{i=1}^{n}X_i \tag{1}$$

The Geometric mean is defined as

$$A_{Geometric} = (\Pi_{i=1}^{n}X_i)^{1/n} \tag{2}$$

To show that the arithmetic mean is always greater we will show that the arithmetic mean is greater then the geometric mean for $n = 2$
    Hypothsis:

$$\frac{x+y}{2} > \sqrt{xy} \tag{3}$$

$$(\frac{x+y}{2})^2 - xy = (\frac{x+y}{2})^2 > 0 \tag{4}$$

The arithemetic mean would be the largest when ever value, α is the same

$$\alpha = \frac{\Sigma\alpha}{n} = \sqrt[n]{\alpha\alpha...\alpha} = (\alpha^n)^{1/n} \geq \sqrt[n]{\Pi_{i=0}^{n}x_i} \tag{5}$$

References

[1] Alan Clements. *Computer Organization and Architecture*. Global Engineering: Christopher M. Shortt, themes and variations edition, 2014.