# CS147 - Lecture 13

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Performance Measurement
- CPU Performance Factor
- Improving CPU Performance

*Reference Books:*

*1) Chapter 4 of 'Computer Organization & Design' by Patterson & Hennessy*

Performance Measurement ...

2

# Performance Metrics

- Two types of performance metrics are measured for a computing system including processor.
  - **Response Time**
    - The time between start and completion of a task (or program). a.k.a Execution time.
  - **Throughput**
    - Total amount of work done in a given time.

3

- Once we have built-up our first processor (in lecture-11) the target is to improve its performance. To improve performance, we first need to understand what do we mean by performance. To different person, performance may mean different metric. There are two types of metric those are popular within computer architecture domain – response time and throughput.
- Response time is more common and easy to understand from a user perspective. It is the time difference between program / task submission time and the completion time. User needs to wait for some finite time before execution of a program / task is done. This waiting time is the response time of the system for that specific program. Often the response time is expressed as the average response time taking average between response time of a wide variety of standard programs (a.k.a. Benchmark programs).
- Throughput on the other hand is measured from service point of view. It is a measurement of total amount of work (task) done in a given time. For example, MIPS (million instruction per second – not another MIPS as processor name – Microprocessor without Interlocked Pipelined Stages) is one of the processor performance (throughput) measurement unit. A system with 0.5 MIPS is clearly of lower throughput performance than 0.8 MIPS system. From web service perspective, throughput for a web system is how many request a website can serve within a specified amount of time.

# Throughput & Response Time

- What does the following action affect which type of performance?
  - Replacing the processor in a computer with a faster version.
  - Adding additional processors to a system that uses multiple processor for separate task.

4

- Using faster processor will always reduce the response time. However, decreasing response time always improve throughput. If individual tasks takes less time to complete, within a specific time duration more tasks will be done, improving throughput of the system. Hence, replacing processor with a faster version will improve both response time and throughput.
- Adding multiple processor will increase throughput, because multiple tasks can be completed in parallel, hence in a given amount of time more tasks will be done. It also affects the response time indirectly. For large number of tasks requested on a system (e.g. billion web request to a site server), tasks will queue up and wait till they get a turn to be executed on a system. As a result, response time (from user perspective) will decrease, if more processor is added to the system.
- Hence, both the cases will increase performance of a system from response time and throughput perspective.

# Comparing Two Systems

- Performance 'P' of a system is reciprocal of execution time (E) for a given program.

$$P = \frac{1}{E}$$

- For a given two system X and Y, the performance comparison will be as following.

$$\frac{P_X}{P_Y} = \frac{E_Y}{E_X}$$

5

# Comparing Two Systems

- If computer A runs a program in 10 sec and computer B runs the same program in 15 sec, how much faster is A than B?

$$\frac{P_A}{P_B} = \frac{E_B}{E_A} = \frac{15}{10} = 1.5$$

- Hence computer A is 1.5 times faster than computer B.

# Measuring Performance

- The following are measured for performance.
  - Wall clock time (or response time or elapsed time)
  - CPU time
    - User CPU time
    - System CPU time
- System performance refers to the wall clock time
- CPU performance refers to the user CPU time.

7

- In general wall clock time is the measure for system's performance. This is the time difference between start of a program to end of the program. This is also known as response time or elapsed time. However, elapsed time always does not reflect the processor or CPU performance well, especially in multi-user / multi-program environment. This is because, a program can wait for different reason in a system for multi-user/multi-program scenario. There can be a wait time to be scheduled by operating system which will increase the wall clock time. The IO operation duration (writing to a disk for example) will add up to the wall clock time (IO operations are usually much slower to the CPU operations). Hence measuring wall clock time is not a true reflection of a CPU performance. However, wall clock time is a true measure of a system performance.
- CPU performance is measured with CPU time, which is the total time span that a CPU did work on a specific program or task (not including the waiting time in queue to be scheduled). However, any modern application program involves software tasks (or routine) belongs to system software (for example printf in C). Thus, CPU time consists of two parts - 'User CPU time' (time spent on user code) and 'System CPU time' (time spent on system code as a part of user program). Sometimes, it will not be true reflection of CPU performance if we include system CPU time, because standard or benchmark program does not have any control over the system codes (it belongs to the OS – hence different OS may result in different total CPU time since it affects the System CPU time). Hence, to analyze the CPU performance, the user CPU time will be the right metric for measuring CPU / processor performance.

CPU Performance Factor ...

8

# Clock Cycles & Frequency

- For a synchronous CPU, the execution time (E) is related to number of clock cycles (N) used by the program and the clock period (T => Frequency F = 1/T)

$$E = N \times T = \frac{N}{F}$$

- Hence the execution time can be improved, either by decreasing number of clock cycles for a program or by decreasing clock period (== increasing clock frequency).

9

- Execution time is the time spent by processor on a user program. The user program needs to be executed within a finite number of clock cycle (N). This N varies from program to program. A large program needs more clock cycle than a small program. Hence this execution time is program dependent. There has been some standard programs, agreed upon by different processor design houses, that are used to measure a processor performance. These programs are known as benchmark programs (e.g. SPEC CPU, SPECWeb99, SPEC INT).

- Since execution time directly proportional to number of cycle (N) and clock period (T), execution time will be decreased if either of them is decreased. However, since processor design houses does not have any control on the benchmark programs, they try to increase the processor speed (by increasing the clock frequency or decreasing the clock period – but often that is not a very straightforward to achieve).

- On the other hand, compiler development companies target to reduce number of instructions (thus the number of cycle needed to execute a program) per program. However, this is not also a very easy job to reduce the number of instructions per program. It is often tied to ISA of the target processor.

- Number of clock cycle also dependent on processor implementation (e.g. using combinational multiplier vs. sequential multiplier).

# Improving Performance

- A program runs in 10 sec on computer A with 4GHz clock frequency. A new computer B needs to be built to run the same program in 6 sec. It has been determined that the clock rate (F) can be increased a lot, but it will need the program to run 1.2 times of the number of clock cycle (N) that was required on computer A. For this program, what will be the target clock rate for the computer B?

$$E_A = \frac{N_A}{F_A},\ E_B = \frac{N_B}{F_B}$$

$$\frac{E_A}{E_B} = \frac{N_A}{N_B} \times \frac{F_B}{F_A}$$

$$\Rightarrow \frac{10}{6} = \frac{1}{1.2} \times \frac{F_B}{4\text{GHz}}$$

$$\Rightarrow F_B = \frac{10 \times 1.2 \times 4\text{GHz}}{6}$$

$$= 8\text{GHz}$$

10

# Clock Per Instruction (CPI)

- CPI or clock per instruction is an average number of clock cycle required to complete an instruction.
- Number of clock cycle (N) for a program is product of number of instructions in the program (or instruction count I) and CPI.

$$N = I \times CPI$$

- CPU time (E) can be expressed in terms of CPI as following (where F is the clock frequency).

$$E = \frac{I \times CPI}{F}$$

11

- Now, how to relate the number of clock cycle needed (N) to finish a program to the number of instructions (machine code) of the program (I). CPI or cycle per instruction is a metric that represent how many clock cycle is needed in average to complete an instruction. Hence, if there is total I instructions for a program executing on a processor with average clock cycle per instruction as CPI then number of clock cycle N is (I*CPI).

# Clock Per Instruction (CPI)

- Computer A and B implements same ISA. Computer A has clock cycle of 250 ps and CPI of 2.0. Computer B has 500 ps cycle time with CPI of 1.2. Which computer is faster and how much?

$$\frac{P_A}{P_B} = \frac{E_B}{E_A} = \frac{N_B \times T_B}{N_A \times T_A}$$
$$= \frac{I \times CPI_B \times T_B}{I \times CPI_A \times T_A}$$
$$= \frac{1.2 \times 500}{2.0 \times 250} = 1.2$$

Computer A is 1.2 times as fast as computer B



 

# Heterogeneous ISA

- All the instruction may not be taking same amount of clock cycles. Taking average of all type of instructions may lead into less accurate performance analysis.

- Better approach is to consider individual CPI for each instruction (or instruction type). The following would be the formula for the number of clock cycle (N) with respect to ISA with instruction categories 1..n

$$N = \sum_{i=1}^{n} I_i \times CPI_i$$



 

# What affects the performance?

| HW or SW Component | Affects What? | How? |
|---|---|---|
| Algorithm | Instruction Count & CPI | Algorithm determines the number of instructions. It may affect overall CPI by choosing slower instruction over faster one. |
| Programing Language | Instruction count & CPI | It certainly affects the program instruction count. It may also affects the CPI by feature. e.g. Java supports data abstraction heavily, which needs indirect calls and thus higher CPI. |
| Compiler | Instruction count & CPI | The efficiency of compiler certainly dictates the choice of instructions and optimization of the program which will affect both instruction count and overall CPI. |
| ISA | Instruction count & CPI & clock rate | ISA impacts all three aspects of CPU performance. It dictates number of instructions to be executed, cost in each cycle of each instruction, and the overall clock rate of the processor. |

# CPI based measurements

- A computer has three categories of instruction A,B and C with CPI as 1,2 and 3 respectively. Two revision of compiler creates different set of instructions count per type of instruction. Revision 1 creates 2,1 and 2 instructions per type A, B and C. Where the revision 2 creates 4, 1 and 1 instructions per type A, B and C. Which revision of the compiler is providing faster implementation?

$$N_{r1} = \sum_{i=A}^{C} CPI_i \times I_i = (1 \times 2) + (2 \times 1) + (3 \times 2) = 10$$

$$N_{r2} = \sum_{i=A}^{C} CPI_i \times I_i = (1 \times 4) + (2 \times 1) + (3 \times 1) = 9$$

- Hence revision 2 is providing faster implementation.

Improving Performance ...

16

## Steps for improvement

- Analyze and break-down performance of a system into performance per independent module.
- Identify the largest performance bottleneck and target to improve that part.
- Try to make common case faster.

17

- Any system (hardware or software) consists of multiple smaller submoudle (e.g. for a software system smaller modules are individual top level functions and procedures). Each of this submodule contributes to overall system performance. For improving performance of any such system, the first step is to analyze the performance of individual submodule with respect to benchmark programs. Once we have complete performance data, we start with slowest submodule and improve its performance. This is because improving slowest module will improve the overall system by an extent more than the improvement can be achieved by improving submodule which is already fast.
- From processor perspective, another approach is to make common case faster. For example, if we find out that the most of the application program is majorly using addition and subtraction over multiplication, significant performance improvement can be done by improving addition / subtraction operation. This is because, even if we improve the multiplication operation instead of addition / subtraction, the improvement will not be seen by the application programs because they mostly execute addition / subtraction.

# Pitfalls and fallacies

- Expecting the improvement of one aspect of a computer to increase performance by an amount proportional to the size of the improvement.

- Using a subset of the performance equation as performance metric.

18

- At the beginning of a performance improvement project, we tend to define very unrealistic goal if we do not analyze the performance problem space very carefully. This majorly happens due to overlook of big picture of the problem space leading to use only sub-set of system's performance equations / factors.

# Amdahl's Law

- Used to determine the theoretical limit for performance improvement. If a 1 is the full execution time of a program and B (<1) is the part which is improved by factor of n then the over all speedup (S) will be as following.

$$S = \frac{1}{(1-B) + \frac{B}{n}}$$

19

- This specific equation is very widely used in industry (at least at the conception level). This equation has two major observations.
  - Improving a sub-module by a factor of n will not make the whole system faster by same factor.
  - The more is B, more is the overall performance improvement.

# Unrealistic Expectation

- A program runs in 100sec on a computer with multiply operation responsible for 80sec of this time. How much does this multiplier needs to be improved to execute this program 5 times faster?

$$S = 5 = \frac{100}{(100-80)+\frac{80}{n}}$$

$$\Rightarrow 20 + \frac{80}{n} = 20$$

$$\Rightarrow \frac{80}{n} = 0$$

$$\Rightarrow n = \infty$$

# CS147 - Lecture 13

Kaushik Patra
(kaushik.patra@sjsu.edu)

21

- Performance Measurement
- CPU Performance Factor
- Improving CPU Performance

*Reference Books:*

*1) Chapter 4 of 'Computer Organization & Design' by Patterson & Hennessy*