Computer Metrics

- We will introduce some of the ways of comparing the performance of various computers.
  - We begin by introducing some of the terminology of performance.

- William Thomson (1824 – 1907), a British scientist whose work on transmission lines said:
  - *"When you can measure what you are talking about, and express it in numbers, you know something about it. But when you cannot measure it, and when you cannot express it in numbers, your knowledge is unsatisfactory kind."*

Throughput

- The ***throughput*** of a computer is a measure of *the* ***amount of work*** *it performs per unit time*.
- The ***upper limit*** **to a system's throughput** can normally be determined from basic system parameters.
  - For example, if a computer has a 500 MHz clock and it can execute up to two instructions in parallel (dual processors) and each instruction takes 1, 2, or 4 clock cycles, then the upper limit on throughput occurs when all instructions are being executed in parallel in one cycle; that is 10^9 instructions/s.
- Note that the definition of ***throughput*** includes the term *amount of work* because instruction execution is *meaningful* only if the instructions are performing *useful* calculations;
  - A computer executing an endless stream of NOPs (no operations) may be operating at its peak rate but is achieving nothing other than to wait.
- *Instructions per second* is a *very poor indicator* of the actual performance of a computer.

Efficiency

- A computer, of course, always executes instructions unless it is in a halt state or a suspended state.
- A computer may not always execute *useful* instructions because, for example, it may be repeatedly doing a busy loop waiting for data.
- The ***efficiency*** of a computer is an indication of the fraction of time that it is doing *useful* work.

$$Efficiency = \frac{Total\ time\ executing\ useful\ work}{Total\ time} = \frac{Optimal\ time}{Actual\ time}$$

- **Example**, if a computer takes 20s to perform a computational task and 5s is taken waiting for a disk that has been idle to spin up to speed, the efficiency is 20s/(20s + 5s) = 20/25 = 80%.

Latency

- *Latency is the delay* between activating a process (for example, a memory write or a disk read, or a bus transaction) and the start of the operation.
  - latency is the *waiting time*.
- Some define latency as the time until *finishing* a process.

- In some computer applications, the effects of latency may be negligible in comparison with processing time.

- In other systems, the effects of latency may have an important effect on system performance.

Relative Performance

- We are interested in how one computer performs with respect to another.

$$Performance_{A\_to\_B} = \frac{Performance_{ComputerA}}{Performance_{ComputerB}}$$

- A computer performance is *inversely* proportional to its execution time

$$Performance \propto \frac{1}{ExecutionTime}$$

- The relative performance of computers A and B is the *inverse* of their execution times; that is

$$Performance_{A\_to\_B} = \frac{Performance_{ComputerA}}{Performance_{ComputerB}} = \frac{ExecutionTime_{ComputerB}}{ExecutionTime_{ComputerA}}$$

- **Example,** If system A executes a program in 105s and system B executes the same program in 125s, we can calculate the A to B relative performance as 125/105 = 1.19. You can say that machine A is 19% faster than B.

Relative Performance

- When you are trying to improve a system, you are often most interested in how much better the new system is in comparison with the old system.
  - The *old* system may be
    - a previous machine,
    - the same machine without the improvement, or
    - even a competitor's machine,
  - It is called the *reference* machine or *baseline* machine.

- The speedup ratio is a measure of *relative performance* and it is defined as

$$Speedup\ ratio = \frac{Performance\ of\ improved\ machine}{Performance\ of\ reference\ machine}$$

$$Speedup\ ratio = \frac{Execution\ time\ on\ reference\ machine}{Execution\ time\ on\ improved\ machine}$$

- **Example,** If a reference machine takes 100s to run a program and the test machine takes 50s, the speedup ratio is 100/50 = 2.

Time and Rate

- Benchmarks can be expressed
  - as the *time* required to execute a task **or**
  - as the *rate* at which tasks are executed.

- For example, one benchmark may yield a time of 20s, whereas another benchmark may yield a rate of 12 tasks/s.

- They say people feel *more comfortable with measures that increase numerically with performance* (i.e., rate) rather than those that reduce with performance (i.e., time).

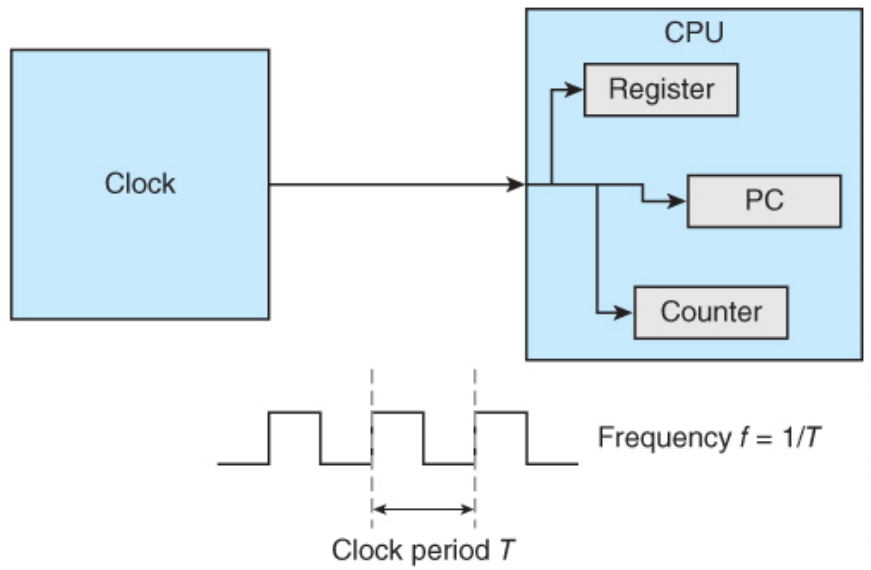
Time and Rate

- ***Average performance:***
  - *Time* and *rate* benchmarks don't behave in the same way with respect to averaging.
  - Suppose we benchmark a computer and get execution times for tasks *A* and *B*, respectively, of 2 and 4 seconds.
  - We can also say that the rates at which tasks *A* and *B* execute are 0.5 tasks/s and 0.25 tasks/s, respectively.
  - The average *execution time* is ½ × (2 + 4) = 3s
    The average *rate* is 1/3 tasks/s = 0.333 tasks/s
  - This average *rate of execution* is ***not*** the average of the two rates
    0.333 tasks/s ≠ ½ × (0.5 tasks/s + 0.25 tasks/s).

Clock Rate

- The obvious indicator of a computer's performance is its *clock rate*, the speed at which internal operations are carried out within the computer.
- Figure 6.9 illustrates the CPU clock.
  At each clock cycle, the processor carries out an internal operation.
- At first sight, it is tempting to think that the processor's performance is directly proportional to its clock rate and therefore clock rate is a precise metric.

FIGURE 6.9 The CPU's clock



Clock Rate

- Yet, using the clock rate as a metric to compare processors is probably the worst metric by which to judge computers.
  - There is *no single clock in most computers* (i.e., there's a separate clock generator for each functional part such as the CPU, the bus, and the memory).
  - Some systems have a single master clock that generates pulses at the highest rate required by any circuit and all other clocks run at a sub-multiple of this frequency.
  - Some processors operate at variable clock rates. For example, *mobile* processors designed for use in laptops can reduce the clock rate to conserve power.
  - Some processors switch to a lower clock speed if the core temperature rises and the chip is in danger of overheating.

Clock Rate

- Since about 2008, clock speed has ceased to increase dramatically because the limits of power dissipation have been reached (power dissipation is directly proportional to the *square* of the clock frequency).
- Clock speeds may rise again if power consumption falls because of the introduction of new semiconductor materials or because of circuit innovations.
- *Manufacturers have directed their efforts towards multicore processors rather than faster processors.*

MIPS

- A slightly better metric than clock rate is **MIPS**, or *millions of instruction per second*.
- This metric removes the discrepancy between systems by measuring *instructions per second* rather than *clocks per second*.
- For a given computer

$$MIPS = \frac{n / 10^6}{t_{execute}}$$

where n is the number of instructions executed and
$t_{execute}$ is the time taken to execute them.

MIPS

- The MIPS rating is a poor metric that fails for the same reason as the clock rate, i.e., *it doesn't account for the efficiency of the instructions*
- MIPS tells you only how fast a computer executes instructions, but *doesn't tell you what is actually achieved by the instructions being executed*.