Lecture 2 notes

Moore's law suggests that the number of transistors per IC doubles every year or so.

Moore's law has been used to characterize the evolution and performance of most components in modern computer systems, including processors, memories, disks, etc.

Today's lecture discussion shows that while Moore's law has been correct in predicting exponential performance improvements in computer system components, these performance improvements are not uniform

- In particular, while bandwidth has improved at much faster rates than latency in networks, disks, processors and memories

Why does latency lag bandwidth?

- First, more and faster transistors help improve bandwidth but do not help as much improving latency. Transistors become smaller and faster. However, the distance (relative to the size of the transistor) between transistors that communicate grows. The size of the dies on which we etch microprocessors also grows
- Distance limits latency and signals can not travel faster than the speed of light
- Bandwidth sounds cooler when it comes to marketing, and the industry seems to eager to throw more resources at improving bandwidth but not latency
- When we improve latency, we often improve bandwidth: For example, multiplying rotational speed in disks by a factor of X improves also bandwidth by a factor of X. Or, lowering DRAM latency by a factor of X, improves bandwidth by a factor of X by allowing more accesses per second
- On the contrary, some techniques that improve bandwidth, hurt latency: For example, putting queues in an interconnection network allows the network to process more requests per unit of time by queuing some requests. However, queuing waiting time increases overall latency. Another example: adding banks to widen a memory module helps bandwidth, but the increases fan-out hurts latency. Yet another example: Software overhead hurts latency more than bandwidth, when processors communicate. The rule of thumb here is that sending/receiving long messages makes communication bandwidth-bound, whereas sending/receiving short messages makes communication latency-bound. Most of this latency is attributed to software initiating and processing message transfers

How do we measure power?

- Power is proportional to capacitive load, voltage (squared) and frequency of switching
- Slowing the clock rate reduces power consumption, but does not reduce energy!
    - Energy is the integral of power over time. Slowing the clock rate increases time.
- Capacitive load depends on number and technology of transistors
- Dropping voltage clearly helps power, but typically voltage and frequency need to be adjusted together to stabilize the processor
- Further reductions in power are possible through dynamically turning off (or throttling) components of the microprocessor (e.g. execution units, cores in a multi-core processor)
- Unfortunately, most of the well known and easily applicable power reduction techniques attack dynamic power consumption. Transistors leak current and this leakage contributes static power consumption, which is a significant part of overall power consumption (can be 40% in current designs). Designers use techniques such as voltage gating (using additional logic to prevent activation signals and the clock signal from reaching inactive components), to control the loss of leakage. Nevertheless, even inactive components leak

Dependability and reliability

- Systems now typically operate under Service-Level Agreements, specifying quantitative guarantees for continuous services, resource availability, etc. and what happens if service is interrupted
- The reliability of hardware modules is characterized by the Mean Time to Failure (MTTF), or failures in time (1/MTTF=FIT)
- Given a failure, the modules are also characterized by mean time to repair (MTTR) and Mean Time Between Failures (MTBF=MTTF+MTTR)
- Module availability is MTTF/MTTF+MTTR=MTTF/MTBF
- With exponentially distributed lifetimes of modules, system failure rate is the sum of the failure rates of the components

Performance

- Performance has been the one and only metric that we were concerned with in this course for the past several years
- Power and reliability are now as important as performance in most domains of computing
- Execution time is the absolute performance metric
- We usually rely on benchmarks for measuring the performance of a system. Benchmarks need to evolve over time to reflect changes in:

applications and the ways we are using computing systems; technology and the capabilities/capacities of computing systems

- Current standards for measuring performance (SPEC) use a ratio of time on a reference (does not really matter what reference) computer over time on the computer being rated
- Since computers are compared in ratios, the choice of the reference computer does not really matter
- Summarizing performance measured in ratios is done best with a geometric mean. We discuss in class whether the geometric mean is a good or bad indicator of "average" performance of a computing system