Power

The primary source of power dissipation for CMOS technology is dynamic power, the power consumed during transistor switching:

\[
\text{Power} = \text{Capacitive Load} \times \text{Voltage}^2 \times \text{Switching Frequency} = C \times V^2 \times f
\]

**Power Problem:** We’re going to roll out a new processor. Compared to the previous version, we’ve reduced our capacitive load by 25% and are running it on 2V instead of 3V. How much faster can we clock the processor without exceeding the power consumption of the previous version?

\[
C_1 V_1^2 f_1 \geq C_2 V_2^2 f_2 \implies \frac{f_2}{f_1} \leq \frac{C_1}{C_2} \times \left(\frac{V_1}{V_2}\right)^2 = \frac{4}{3} \times \frac{9}{4} = 3, \text{ so 3 times as fast or 200% increase.}
\]

Parallelism

Software can be classified as sequential, where a single computational process is carried out, or concurrent, where collections of interacting computational processes are executed. Hardware can be classified as serial or parallel, depending on whether one or more computations can be carried out simultaneously. These software and hardware classifications are independent of each other (e.g., a sequential process can run on hardware that may execute multiple instructions at a time).

**Job-level (process-level) parallelism,** where independent programs are run on different processors can take advantage of multiprocessors, computers with more than one processor. In particular, today we use the term **multicore microprocessor** in reference to multiprocessors with multiple processors (cores) on the same IC (chip).

**Thread Level Parallelism (TLP):** Executing different processes (threads) of the same program on different processors. Threads can communicate with each other.

**Instruction Level Parallelism (ILP):** Performing different instructions in a program simultaneously.

**Data Level Parallelism (DLP):** Operating on independent data simultaneously.

Flynn’s Taxonomy

Flynn’s taxonomy is a classification of computer architectures into Single/Multiple Instruction and Single/Multiple Data stream. The following image is taken from [Wikipedia]:

<table>
<thead>
<tr>
<th>SISD</th>
<th>SIMD</th>
<th>MISD</th>
<th>MIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="SISD Diagram" /></td>
<td><img src="image2" alt="SIMD Diagram" /></td>
<td><img src="image3" alt="MISD Diagram" /></td>
<td><img src="image4" alt="MIMD Diagram" /></td>
</tr>
</tbody>
</table>

So far we have been assuming use of SISD, or just a normal uniprocessor machine. MISD is generally unused, so the main areas of interest are SIMD and MIMD.

Amdahl’s Law

In general terms, Amdahl’s Law states (quoting P&H) that “the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.” Let “exec time” stand for execution time:
exec time after improvement = \frac{\text{exec time affected by improvement}}{\text{amount of improvement}} + \text{exec time unaffected}

As stated above, it’s clear to see that this supports the common design principle of *making the common case fast*.

Restated in terms of parallelism, the potential speedup from parallelization is limited by the amount a program can be parallelized and the level of parallelization (here amount of improvement = number of processors). Let $F$ be the fraction of the original execution time affected by the improvement (% of process that can be parallelized) and $N$ be the number of processors. Then the amount of speed-up going from 1 to $N$ processors is given by:

\[
\text{Speedup} = \frac{1}{(1 - F) + F/N}
\]

This equation has the following limits:

\[
\lim_{F \to 1} \text{Speedup} = N, \quad \text{and} \quad \lim_{N \to \infty} \text{Speedup} = \frac{1}{1 - F}
\]

In general terms, what conditions would allow us to achieve the “ideal value” of Speedup = $N$?

A perfectly parallelizable program – program can be split into $N$ equal and independent tasks.

In general terms, what conditions would allow us to achieve the “ideal value” of Speedup = $\frac{1}{1 - F}$?

Infinite parallelization – assume $\infty$ processors with no parallelization overhead.

**Exercise:** In a given program, 95% of the execution time can be parallelized. How many processors are needed to achieve a speed-up of over 10?

First check: $\frac{1}{1 - F} = \frac{1}{0.05} = 20 > 10$, so speed-up of 10 is possible.

$10 \times (1 - 0.95) + 10 \times \frac{0.95}{N} < 1$, so $0.5 + \frac{9.5}{N} < 1$ and $N > 19$.

**Exercise:** A given program runs $x = 55$ floating point operations, 5 of which can’t be parallelized. What is the speed-up for 5 processors? Now if $x = 105$ floating point operations with the same 5 that can’t be parallelized, what is the speed-up for 5 processors? Assume a floating point operation takes $t_{fp}$ time.

\[
\frac{50t_{fp}}{5} + 5t_{fp} = 15t_{fp} \quad \text{for a speed-up of} \quad \frac{55}{15} = \frac{11}{3}. \quad \text{Or, speed-up} = \frac{1}{(1 - \frac{10}{11}) + \frac{10/11}{5}} = \frac{11}{3}.
\]

\[
\frac{100t_{fp}}{5} + 5t_{fp} = 25t_{fp} \quad \text{for a speed-up of} \quad \frac{105}{25} = \frac{21}{5} > \frac{11}{3}. \quad \text{Or, speed-up} = \frac{1}{(1 - \frac{20}{21}) + \frac{20/21}{5}} = \frac{21}{5}.
\]

**Fa 12 Midterm Q3**

d) You have to solve a problem using Amazon EC2 servers. You know the server will finish the problem in an hour using 10 machines, but the deadline for your solution is just over 1 minute away. You attempt to solve the problem quickly by running an instance with 600 machines.

However, even though the cluster magically booted up instantaneously, you were late turning in your project and wasted a lot of money in the endeavor. This scenario indicates your solution lacked which kind of scaling? Circle one:

- **Strong Scaling**
- **Weak Scaling**