Lecture 1

Moore's Law

"The number of transistors in a dense integrated circuit will double exponentially every 2 years" -Gordon Moore 1965

- Not technically a "law" but an observation $\rightarrow$ prediction.
- True more-or-less until 2012, now slowing down.
- Processor clock rates stopped increasing in the early 2010s due to heat dispersion issues.
- As we cannot increase performance via clock rate, we instead increase transistor count to put multiple processors on the same chip to ger more work done via parallelism.
    - e.g.
        - Intel i9 Extreme i9-7980XE: 18 cores, 36 threads
        - AND Ryzen Threadruipper 2990WX: 32 cores, 64 threads

Cores vs (Hyper)threads

- A core contains an ALU, FPU, several caches + misc. registers
- There are enough transistors on a modern chip that we can double up registers & program counter on a core, while sharing the ALU and FPU. This, therefore, allows 2 'threads' of execution on the same core.
- Caches
    - Muli-level
        - L1 is closest to the core, shared between 2 threads
        - L2 can be shared between 2 cores
        - L3 is shared by **all** cores

Measuring Parallel Speedup

- Latency: Time from initiation $\rightarrow$ computation.
- Work: a measure of what has to be done for a particular task.
- Throughput: Work done per unit time

Speedup & Efficiency

- Speedup$_p$, S_p : ratio of latency for solving a problem with 1 hardware unit to latency of solving with P hardware units.

    - $S_p = \frac{T_1}{T_p}$
    - Perfect linear speedup = $S_p = p$

- Efficiency$_p$, E_p : Ratio of latency for solving a problem with 1 hardware unit to P times the latency of solving it on P hardware units.

    - Measures how well the individual hardware units are contributing to the overall solution.
    - $E_p = \frac{T_1}{p \times T_p} = \frac{S_p}{p}$

Interpreting Speedup & Efficiency

- Sub-linear speedup and efficiency is **normal** due to overhead in parallelising a problem.
- Super-linear speedup is possible, but usually due to special conditions.
  - e.g. serial problem doesn't dit in a CPU cache, but parallelised versions do.
- Important to compare the **best** serial solution to the parallel version.

Strong Scalability

Amdahl's Law

- Gene Amdahl, in 1967 argued that the time to execute a program is comprised of:
  - Time doing non-parallelisable work +
  - Time doing parallelisable work
    - $T_1 = T_{ser} + T_{par}$
  - Therefore, if speedup on P units of the parallel part is S
    - $T_p = T_{ser} + \frac{T_{par}}{S}$
  - Therefore, overall speedup, given the speedup of the parallel part is S is:
    - $S_p = \frac{T_{ser}+T_{par}}{T_{ser}+\frac{T_{ser}}{S}}$
  - If let let f be te function of the program that is parallelisable then:
    - $T_{ser} = (1-f)T_{par} $ & $ T_{par} = fT_1$
    - $\therefore$

$$S_p = \frac{(1-f)T_1 + fT_1}{(1-f)T_1+\frac{fT_1}{S}} = \frac{1}{1-f+\frac{f}{S}}$$

- This law says that there is a limit to parallel speedup

$$ \lim_{S\rightarrow\infin}\frac{1}{1-f+\frac{f}{S}}= \frac{1}{1-f}$$

or alternatively,

$$\lim_{S\rightarrow\infin}\frac{T_1}{T_p} = \frac{1}{1-f}\implies \lim_{S\rightarrow\infin} T_p = T_{ser}$$

- Assuming $P \rightarrow \infin \implies S \rightarrow \infin$

Weak Scalability

- John Gustafson & Edwin Barsis, in 1988, argued that Amdahl's law did not give the whole picture
  - Amdahl kept the task fixed and considered how much you could shorten the processing time by running in parallel.
  - Gustafson & Barsis kept the processing time fixed and considered how much larger a task one could handle in that time by running in parallel
  - This approach was motivated by observing that as computers increase in power, the problems that they are applied to also increase in size.

Gustafson - Barsis Law

- Assume that W is the workload that can be executed without parallelism in time T. If f is the fraction of W that is parallelisable then:

  - $W = (1-f)W+fW$

- With speedup, S, we can run S times the parallelisable part in the same time, although, we don't change the amount of work done in the non-parallelisable part:
    - $W_s = (1-f)W + SfW$
- If we do W_S in time T, we are, on average, doing W amount of work in time $\frac{TW}{W_S} \therefore$ the total speedup is:

$$S_S = \frac{\frac{T}{TW}}{W_s} = \frac{W_s}{W} = 1-f+fS$$

Conclusion

- Both Amdahl & Gustafson$\cdot$Barsis are correct
- Together, they give guidance on which tasks can benefit from parallelisation & how
- You can only go so much faster ona fixed problem using parallelisation, **one cannot avoid Amdahl's limit on fixed problem size**
- However, when growing the size of the task, you can increase te size of the parallelised part of the problem faster than you increase the size of the non-parallelisable part, then Gustafson-Barsis gives opportunities for speedups that are not available otherwise.