CSC346/646 Parallel Computation Spring 2018

Speedup, Amdahl's Law, Gustafson's Law

Whenever we run a parallel program, we hope that more available processors will result in faster program execution. *Speedup* refers to the ratio of the time taken by a sequential program to the time taken by a corresponding parallel program. To be fair, we should compare our best sequential implementation with our best parallel implementation. In practice, the sequential time is often measured by running the parallel code with one processor ; that avoids having to write and optimize a separate sequential code.

Notation:

We define:

$$\begin{aligned} N &= \text{problem size} \\ T_s &= \text{time to perform a computation sequentially} \\ T_{\text{par}}(p) &= \text{time to perform the same computation with } p \text{ processors} \end{aligned}$$

Then speedup is given by:

$$S(p) = \frac{T_s}{T_{\text{par}}(p)}$$

If there is no overhead, and no contention for resources, we might hope a perfect parallel code to have linear speedup. I.e., $S(p) = p$. That's rarely the case. Usually, there is some synchronization, or limited resource that causes sub-linear speedup as more processors are added.

Super-linear Speedup: Occasionally, it is possible in practice to see a program exhibit super-linear speedup. I.e. $S(p) > p$ for some values of p and problem size N. This is counter-intuitive, and seems unlikely on the face of it.

The answer (in practice) is that when you add more processors on a traditional parallel computer, you also add one very important additional resource: **more cache memory**. Having a larger portion of the code and data stored in cache memory can cause additional (super-linear) speedup.

Amdahl's Law:

Let W denote the time needed for the total amount of work to be done. If we assume α represents the fraction of the code that must be performed sequentially, and $1 - \alpha$ the fraction that can be performed in parallel. Then, we expect the sequential time to be:

$$T_s \;=\; W$$

and the parallel time to be:

$$T_{\text{par}}(p) \;=\; \alpha W + \frac{(1-\alpha)W}{p}$$

and speedup is:

$$S(p) \;=\; \frac{W}{\alpha W + \frac{(1-\alpha)W}{p}} \;=\; \frac{1}{\alpha + \frac{(1-\alpha)}{p}} \quad .$$

If we allow the number of processors p to go to infinity, we see:

$$S_{\max} \;=\; \lim_{p\to\infty} \frac{1}{\alpha + \frac{(1-\alpha)}{p}} \;=\; \frac{1}{\alpha}$$

Under these assumptions, speedup is limited to $1/\alpha$ no matter how many processors we have. For example, if 5% of the work must be done sequentially, the best we can hope for is a speedup $S = 20$, regardless of how many processors we use.

Gustafson's Law

Amdahl's Law gives a rather pessimistic prediction regarding the speedup that we might hope to achieve. A key assumption in Amdahl's law is that the fraction of sequential work is fixed, regardless of the size of the problem. In many practical problems, that assumption is not true. More often, the fraction of work that can be done in parallel grows as the problem size increases; Gustafson's law takes this observation into consideration.

Consider a parallel program running with p processors, and let A and B be the amounts of time spent running the sequential and parallel portions of the program respectively.

We have:

$$T_{\text{par}}(p) \;=\; A + B$$

and

$$T_s \;=\; A + pB$$

since it takes a single processor p times longer to execute the parallel portion of the program. We then have speedup:

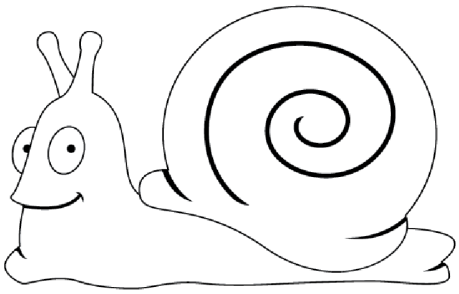
$$S \;=\; \frac{A + pB}{A + B}$$

If we let β be the ratio of time spent doing sequential operations to the total time, i.e., $\beta = A/(A + B)$ then we have:

$$\begin{aligned} S &= \beta \;+\; p(1 - \beta) \\ &= p \;-\; \beta(p - 1) \end{aligned}$$

This last expression gives us a more optimistic view of what might be achieved in parallel. If β is small (e.g. near zero), then speedup will be close to p (perfect linear speedup). For some problems, β becomes smaller as the problem size N becomes larger.

Later in the semester (when we discuss iso-efficiency), we will examine the relationship between problem size N and the number of processors p using an analysis of the asymptotic sequential time (big O) and an analysis of the asymptotic parallel time.



Trying to go faster ...