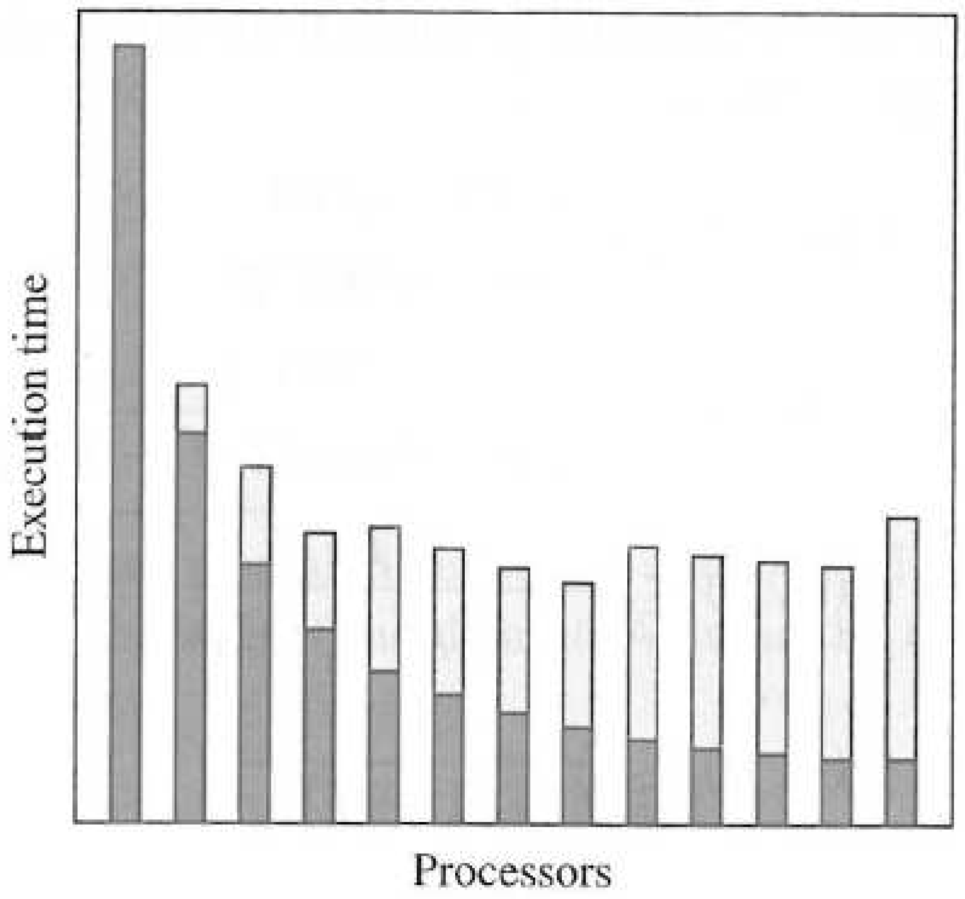
Performance analysis

An example of time measurements

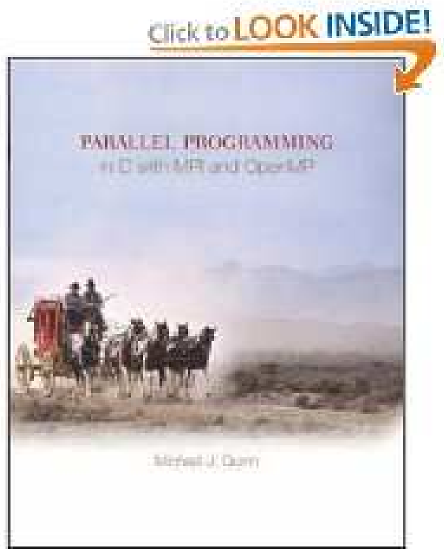


Dark grey: time spent on computation, decreasing with p
White: time spent on communication, increasing with p

Objectives of the lecture

- How to roughly predict computing time as function of p?
- How to analyze parallel execution times?
- Understand the limit of using more processors

Chapters 7 from *Michael J. Quinn*, **Parallel Programming in C with MPI and OpenMP**



Notation

n	problem size
p	number of processors
$\sigma(n)$	inherently sequential computation
$\varphi(n)$	parallelizable computation
$\kappa(n,p)$	parallelization overhead

$$\text{Speedup } \Psi(n,p) = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

$$\text{Efficiency } \varepsilon(n,p) = \frac{\text{Sequential execution time}}{\text{Processors used} \times \text{Parallel execution time}}$$

Simple observations

- Sequential execution time $= \sigma(n) + \varphi(n)$
- Parallel execution time $\geq \sigma(n) + \varphi(n)/p + \kappa(n,p)$

$$\text{Speedup } \Psi(n,p) \leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n)/p + \kappa(n,p)}$$

$$\text{Efficiency } \varepsilon(n,p) \leq \frac{\sigma(n) + \varphi(n)}{p\sigma(n) + \varphi(n) + p\kappa(n,p)}$$

Amdahl's Law

If the parallel overhead $\kappa(n,p)$ is neglected, then

$$\text{Speedup } \Psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

Suppose we know the inherently sequential portion of the computation,

$$f = \frac{\sigma(n)}{\sigma(n)+\varphi(n)}$$

Can we predict the speedup $\Psi(n,p)$?

Amdahl's Law

Note that $f = \frac{\sigma(n)}{\sigma(n)+\varphi(n)}$ means

$$1 - f = \frac{\varphi(n)}{\sigma(n) + \varphi(n)}$$

Therefore

$$\begin{aligned} \Psi(n,p) &\le \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n)/p} \\ &= \frac{1}{\frac{\sigma(n)}{\sigma(n)+\varphi(n)} + \frac{\varphi(n)}{\sigma(n)+\varphi(n)}\frac{1}{p}} \\ &= \frac{1}{f + (1-f)/p} \end{aligned}$$

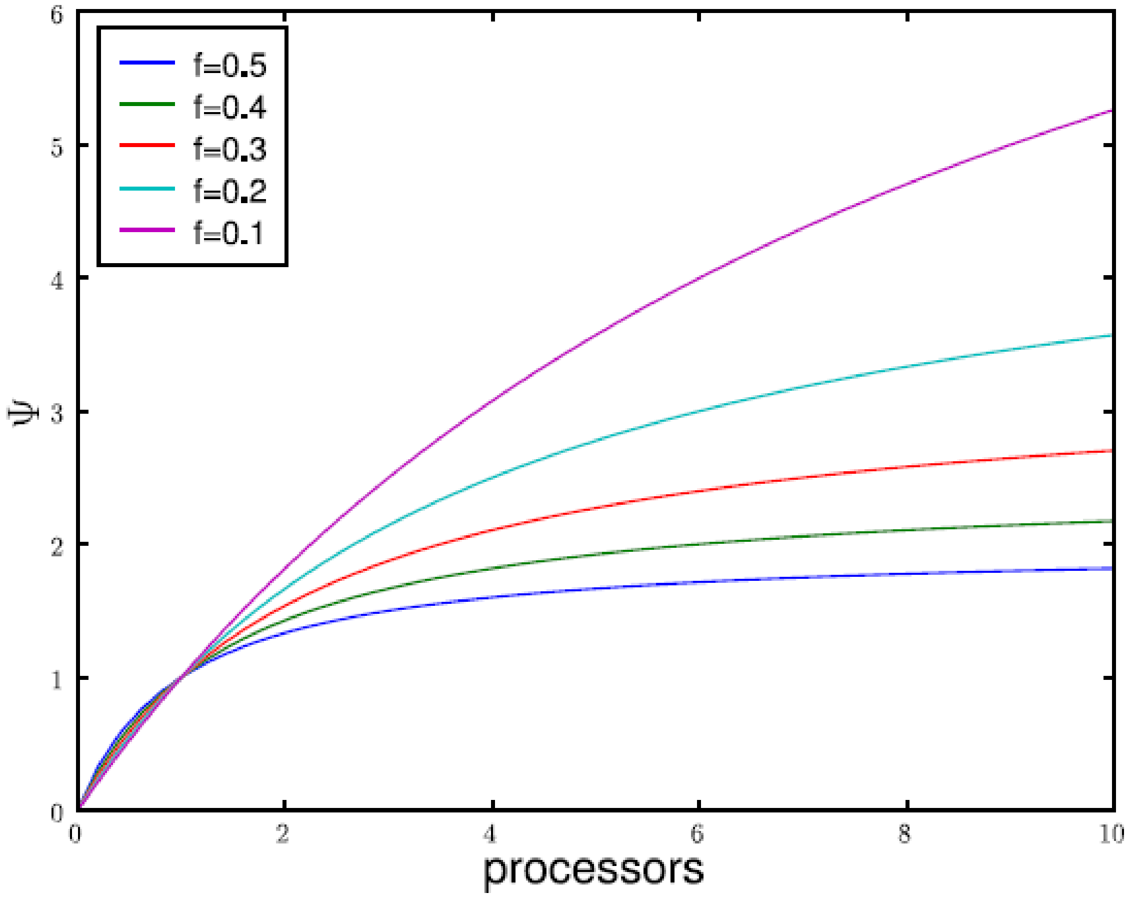
Amdahl’s Law

Amdahl’s Law: if $f = \sigma/(\sigma + \varphi)$ is known, then the best achievable speedup can be estimated as

$$\Psi \leq \frac{1}{f + (1 - f)/p}$$

Upper limit (when p goes to infinity): $\Psi \leq \frac{1}{f+(1-f)/p} < \frac{1}{f}$

Amdahl’s Law



Example 1

If we know that 90% of the computation can be parallelized, what is the maximum speedup we can expect from using 8 processors?

Solution

Since f=10%, Amdahl's Law tells us for $p = 8$

$$\Psi \le \frac{1}{0.1 + \frac{(1-0.1)}{8}} \approx 4.7$$

Example 2

If 25% of the operations in a parallel program must be performed sequentially, what is the maximum speedup achievable?

Solution

The maximum speedup is

$$\lim_{p \to \infty} \frac{1}{0.25 + \frac{(1-0.25)}{p}} = \frac{1}{0.25} = 4$$

Example 3

Suppose

$$\begin{aligned} \sigma(n) &= 18000 + n \\ \varphi(n) &= \frac{n^2}{100} \end{aligned}$$

What is the maximum speedup achievable on a problem of size $n = 10000$?

Example 3 (cont'd)

Solution

Since

$$\Psi(n,p) \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

and we know

$$\begin{aligned} \sigma(10000) &= 28,000 \\ \varphi(10000) &= 1,000,000 \end{aligned}$$

Therefore

$$\Psi(10000,p) \leq \frac{28,000+1,000,000}{28,000+1,000,000/p}$$

Comments about Amdahl’s Law

- Parallelization overhead $\kappa(n,p)$ is ignored by Amdahl’s Law
  - Amdahl’s Law gives a too optimistic estimate of Ψ
- The problem size n is constant for $p = 1$ and increasing p values
  - Amdahl’s Law doesn’t consider solving larger problems with more processors
- The inherently sequential portion f may greatly decrease when n is increased
  - Amdahl’s Law ($\Psi < \frac{1}{f}$) can be unnecessarily pessimistic for large problems

Gustafson–Barsis's Law

What if we want to solve larger problems when the number of processors p is increased?

That is, we may not know the computing time needed by a single processor, because the problem size is too big for one processor.

However, suppose we know the fraction of time spent by a parallel program (using p processors) on performing inherently sequential operations, can we estimate the speedup Ψ?

Gustafson–Barsis's Law

Definition: s is the fraction of time spent by a parallel computation using p processors on performing inherently sequential operations.
More specifically,

$$s = \frac{\sigma(n)}{\sigma(n) + \varphi(n)/p}$$

and

$$1 - s = \frac{\varphi(n)/p}{\sigma(n) + \varphi(n)/p}$$

Gustafson–Barsis's Law

We note

$$\begin{aligned}\sigma(n) &= (\sigma(n)+\varphi(n)/p)s \\ \varphi(n) &= (\sigma(n)+\varphi(n)/p)(1-s)p\end{aligned}$$

Therefore

$$\begin{aligned}\Psi(n,p) &\leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p} \\ &= \frac{(s+(1-s)p)(\sigma(n)+\varphi(n)/p)}{\sigma(n)+\varphi(n)/p} \\ &= s+(1-s)p \\ &= p+(1-p)s\end{aligned}$$

Gustafson–Barsis's Law

Given a parallel program solving a problem of size n using p processors, let s denote the fraction of total execution time spent in serial code. The maximum speedup Ψ achievable is

$$\Psi \leq p + (1 - p)s$$

Comments about Gustafson–Barsis's Law

- Gustafson–Barsis's Law encourages solving larger problems using more processors. The speedup obtained is thus also called scaled speedup.
- If n is large enough for p processors, n is probably too large (with respect to memory) for a single processor. However, this doesn't prevent Gustafson–Barsis's Law from predicting the best achievable speedup Ψ, when s is known.
- Since parallelization overhead $\kappa(n, p)$ is ignored, Gustafson–Barsis's Law may also overestimate the speedup.
- Since $\Psi \leq p + (1 - p)s = p - (p - 1)s$, so the best achievable speedup is $\Psi \leq p$. The smaller s the better Ψ.
- If $s = 1$, then there is no speedup at all, because $\Psi \leq p + (1 - p) = 1$.

Example 1

An application executing on 64 processors uses 5% of the total time on non-parallelizable computations. What is the scaled speedup?

Solution

Since $s = 0.05$, the scaled speedup on 64 processors is

$$\Psi \leq p + (1-p)s = 64 + (1-64)(0.05) = 64 - 3.15 = 60.85$$

Example 2

If we want to achieve speedup $\Psi = 15000$ using $p = 16384$ processors, what can the maximum allowable value of the serial fraction s be?

Solution

Since

$$\Psi \leq p + (1-p)s = p - (p-1)s$$

then

$$s \leq \frac{p - \Psi}{p - 1} = \frac{16384 - 15000}{16384 - 1} \approx 0.084$$

Karp–Flatt Metric

Both Amdahl's Law and Gustafson–Barsis's Law ignore the parallelization overhead $\kappa(n,p)$, they may therefore overestimate the achievable speedup.

We recall

$$\begin{aligned} \text{Parallel execution time } T(n,p) &= \sigma(n) + \varphi(n)/p + \kappa(n,p) \\ \text{Sequential execution time } T(n,1) &= \sigma(n) + \varphi(n) \end{aligned}$$

Karp–Flatt Metric

If we consider the parallelization overhead as another kind of “inherently sequential work”, then we can use Amdahl’s law to experimentally determine a “combined” serial fraction e, which is defined as

$$e(n,p) = \frac{\sigma(n) + \kappa(n,p)}{\sigma(n) + \varphi(n)}$$

This experimentally determined serial fraction $e(n,p)$ may either stay constant with respect to p (meaning that the parallelization overhead is negligible) or increase with respect to p (meaning that parallelization overhead deteriorates the speedup).

Karp–Flatt Metric

If we know the actually achieved speedup $\Psi(n,p)$ using p processors, how can we determine the serial fraction $e(n,p)$?

Since

$$T(n,p) = T(n,1)e + \frac{T(n,1)(1-e)}{p}$$

and we know the value of $\Psi(n,p)$, which is defined as

$$\Psi(n,p) = \frac{T(n,1)}{T(n,p)} = \frac{T(n,1)}{T(n,1)e + \frac{T(n,1)(1-e)}{p}} = \frac{1}{e + \frac{1-e}{p}}$$

Therefore

$$\frac{1}{\Psi} = e + \frac{1-e}{p} \quad \Rightarrow \quad e = \frac{1/\Psi - 1/p}{1 - 1/p}$$

Example 1

Benchmarking a parallel program on $1, 2, \ldots, 8$ processors produces the following speedup results:

p	2	3	4	5	6	7	8
Ψ	1.82	2.50	3.08	3.57	4.00	4.38	4.71

What is the primary reason for the parallel program achieving a speedup of only 4.71 on eight processors?

Example 1 (cont'd)

Solution

We can use Karp–Flatt Metric to experimentally determine the values of $e(n,p)$ as

p	2	3	4	5	6	7	8
Ψ	1.82	2.50	3.08	3.57	4.00	4.38	4.71
e	0.10	0.10	0.10	0.10	0.10	0.10	0.10

Since the experimentally determined serial fraction e is not increasing with p, the primary reason for the poor speedup is the large fraction (10%) of the computation that is inherently sequential. In other words, parallel overhead is not the reason for the poor speedup.

Example 2

Benchmarking a parallel program on $1, 2, \ldots, 8$ processors produces the following speedup results:

p	2	3	4	5	6	7	8
Ψ	1.87	2.61	3.23	3.73	4.14	4.46	4.71

What is the primary reason for the parallel program achieving a speedup of only 4.71 on eight processors?

Example 2 (cont'd)

Solution

We can use Karp–Flatt Metric to experimentally determine the values of $e(n, p)$ as

p	2	3	4	5	6	7	8
Ψ	1.87	2.61	3.23	3.73	4.14	4.46	4.71
e	0.070	0.075	0.080	0.085	0.090	0.095	0.1

Since the experimentally determined serial fraction e is steadily increasing with p, parallel overhead is also a contributing factor to the poor speedup.

Exercises

- Write an MPI program that first uses a master process to read in a JPEG image. (The existing C code collection `http://heim.ifi.uio.no/xingca/inf-verk3830/simple-jpeg.tar.gz` can be used.) Then the master process divides the image and distributes the pieces to the other MPI processes. The purpose is to involve all the MPI processes to calculate the average pixel value of the entire image, and the standard deviation value.
- Suppose a 2D uniform grid is divided into $P \times Q$ rectangular pieces. Each piece is assigned to one MPI process, and that two and two neighboring processes (in both x- and y-direction) need to exchange the values on their outermost boundary-layer points. Can you make use of the following two MPI functions:
  `MPI_Type_vector` and `MPI_Type_commit`
  to help simplify communications in the x-direction?
  (Hint: you should make use of an online MPI manual.)