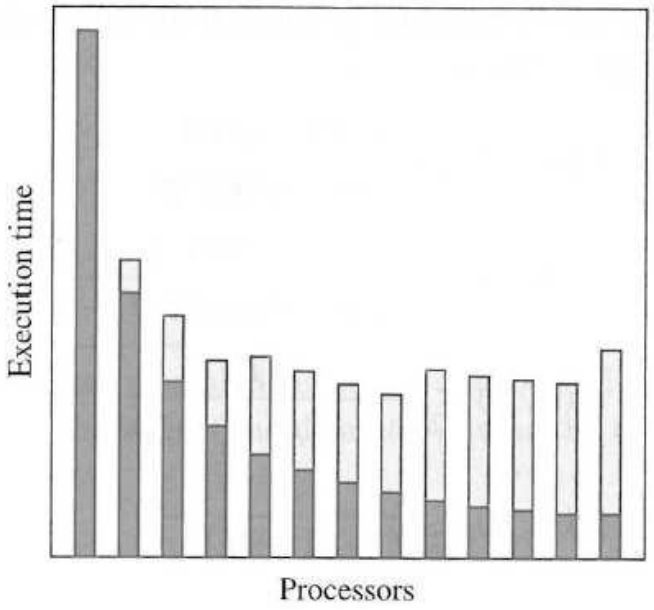
Lecture 6: Performance Analysis

1. Amdahl's law
   - Analyze whether a program merits parallelization
2. Gustafson-Barsis's law
   - Evaluate performance of a parallel program
3. Karp-Flatt metri
   - Decide whether the principle barrier to speedup is due to inherently sequential code or parallel overhead
4. Isoefficiency metric
   - Evaluate the scalability of a parallel program executing on a parallel computer

Typical Time Measurements



Dark grey: time spent on computation, decreasing with # of processors

White: time spent on communication, increasing with # of processors

Operations in a parallel program:

1. Computation that must be performed sequentially
2. Computations that can be performed in parallel
3. Parallel overhead including communication and redundant computations

Basic Units

- n problem size
- p number of processors
- $\sigma(n)$ inherently sequential portion of computation
- $\varphi(n)$ portion of parallelizable computation
- $\kappa(n,p)$ parallelization overhead
- Speedup $\Psi(n,p) = \frac{sequential\ execution\ time}{parallel\ execution\ time}$
- Efficiency

$$\varepsilon(n,p) = \frac{sequential\ execution\ time}{processors\ used \times parallel\ execution\ time}$$

- Sequential execution time $T(n,1) = \sigma(n) + \varphi(n)$

Assume that the parallel portion of the computation that can be executed in parallel divides up perfectly among p processors

- Parallel execution time $T(n,p) \geq \sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)$

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

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

Example 5.6 Edge detection on images

Given an $n \times n$ pixel image, the problem of detecting edges corresponds to applying a3× 3 template to each pixel. The process of applying the template corresponds to multiplying pixel values with corresponding template values and summing across the template (a convolution operation). This process is illustrated in Figure 5.4(a) along with typical templates (Figure 5.4(b)). Since we have nine multiply-add operations for each pixel, if each multiply-add takes time t_c, the entire operation takes time $9t_cn^2$ on a serial computer.

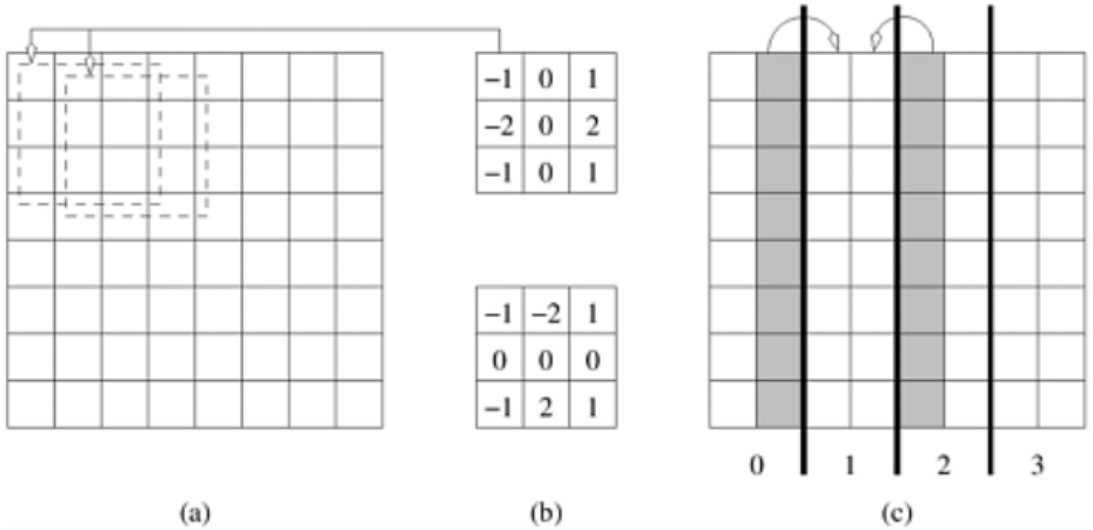


Figure 5.4. Example of edge detection: (a) an 8 × 8 image; (b) typical templates for detecting edges; and (c) partitioning of the image across four processors with shaded regions indicating image data that must be communicated from neighboring processors to processor 1.

Parallel program:

1. Exchange a layer of n pixels with each of the two adjoining processing elements.

   Time takes for message passing: $2(t_s + t_w n)$

2. Apply template on local sub-image.

   Time takes for computing: $9t_c n^2/p$

   Speed up: $$\Psi(n,p) = \frac{9t_c n^2}{\frac{9t_c n^2}{p} + 2(t_s + t_w n)}$$

Amdahl's Law (1)

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

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

Let f be the percentage of inherently sequential portion of the computation, i.e.,

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

Amdahl's Law (3)

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

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

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

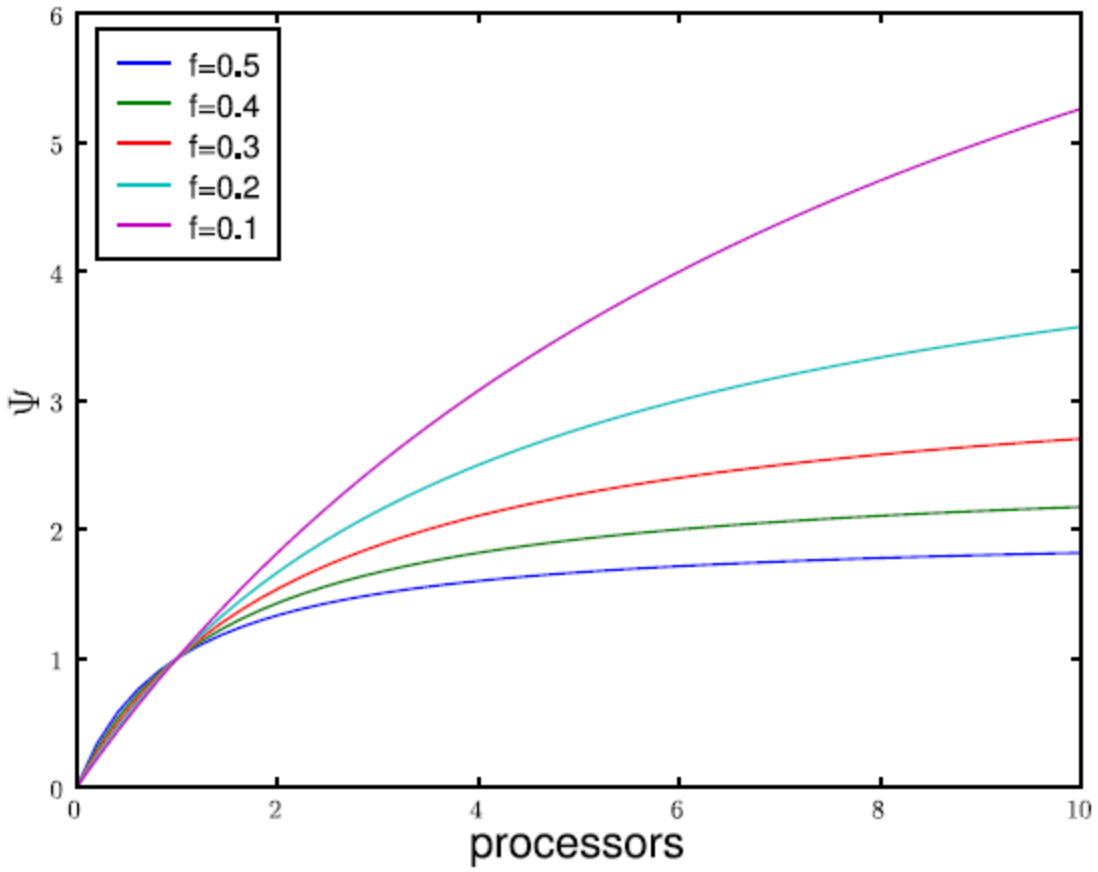
$$\Psi(n,p) \leq \frac{1}{f+\ (1-f)/p}$$

Amdahl's Law: Let f be the fraction of operations in a computation that must be performed sequentially, where $0 \leq f \leq 1$. The maximum speedup $\Psi(n,p)$ achieved by a parallel computer with p processors performing the computation is $\Psi(n,p) \leq \frac{1}{f+(1-f)/p}$

Upper limit: as $p \rightarrow \infty, \Psi(n,p) \leq \frac{1}{f+\frac{1-f}{p}} < \frac{1}{f}$

Speedup vs. f

Amdahl's law assumes that the problem size is fixed. It provides an upper bound on the speedup achievable by applying a certain number of processors.



Example 1

If 90% of the computation can be parallelized, what is the max. speedup achievable using 8 processors?

Solution:

$$f = 10\%,$$

$$\Psi(n,p) \leq \frac{1}{0.1+\frac{1-0.1}{8}} \approx 4.7$$

Example 2

Suppose $\sigma(n) = (18000 + n)\mu sec$

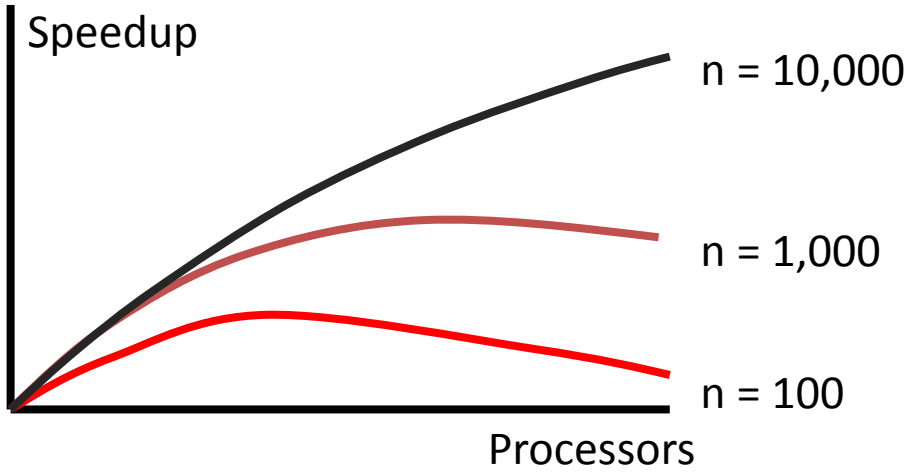
$$\varphi(n) = \left(\frac{n^2}{100}\right)\mu sec$$

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

Solution: $\Psi(n,p) \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}} \leq \frac{28000+1000000}{28000+1000000/p}$

Remark

- Parallelization overhead $\kappa(n,p)$ is ignored by Amdahl's law
  - Optimistic estimate of speedup
- The problem size n is constant for various p values
  - Amdahl's law shows how execution time decreases as number of processors increases.
- **Amdahl effect**
  - Typically $\kappa(n,p)$ has lower complexity than $\varphi(n)/p$. For a fixed number of processors, speedup is usually an increasing function of the problem size.
  - As n increases, $\varphi(n)/p$ is much larger than $\kappa(n,p)$
  - As n increases, speedup increases



- The inherently sequential portion f may decrease when n increases
  - Amdahl's law ($\Psi(n,p) < \frac{1}{f}$) can under estimate speedup for large problems

Gustafson-Barsis's Law

- Amdahl's law assumes that the problem size is fixed and show how increasing processors can reduce time.
- Let the problem size increase with the number of processors.
- Let s be the fraction of time spent by a parallel computation using p processors on performing inherently sequential operations.

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

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

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

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

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

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

$$= s + (1-s)p$$

$$= p + (1-p)s$$

Gustafson-Barsis's law: Given a parallel program of size n using p processors, let s be the fraction of total execution time spent in serial code. The maximum speedup $\Psi(n,p)$ achieved by the program is

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

Remark

- Gustafson-Barsis's law allows to solve larger problems using more processors. The speedup is called scaled speedup.
- Since parallelization overhead $\kappa(n,p)$ is ignored, Gustafson-Barsis's law may over estimate the speedup.
- Since $\Psi(n,p) \leq p + (1-p)s = p - (p-1)s$, the best achievable speedup is $\Psi(n,p) \leq p$.
- If $s = 1$, then there is no speedup.

Example

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

Solution: $s = 0.05$,

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

Karp-Flatt Metric

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

Recall:

- Parallel execution time $T(n,p) = \sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)$
- Sequential execution time $T(n,1) = \sigma(n) + \varphi(n)$

- Define **experimentally determined serial fraction** e of parallel computation:

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

- **experimentally determined serial fraction** e
  - Takes into account parallel overhead
  - Detects other sources of overhead or inefficiency ignored in speedup model
    - Process startup time
    - Process synchronization time
    - Imbalanced workload
    - Architectural overhead

- **experimentally determined serial fraction** e 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 dominates the speedup*)

- Given $\Psi(n, p)$ using p processors, how to determine $\mathrm{e}(n, p)$?

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

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

Therefore, $\frac{1}{\Psi} = e + \frac{1-e}{p}$

$$\rightarrow \qquad e = \frac{\frac{1}{\Psi} - \frac{1}{p}}{1 - \frac{1}{p}}$$

Example 1

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

p	2	3	4	5	6	7	8
$\Psi(n,p)$	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 8 processors?

Solution: Compute $e(n,p)$ corresponding to each data point:

p	2	3	4	5	6	7	8
$\Psi(n,p)$	1.82	2.50	3.08	3.57	4.00	4.38	4.71
$e(n,p)$	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Since the experimentally determined serial fraction $e(n,p)$ is not increasing with p, the primary reason for the poor speedup is the 10% of the computation that is inherently sequential. Parallel overhead is not the reason for the poor speedup.

Example 2

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

p	2	3	4	5	6	7	8
$\Psi(n,p)$	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 4.71 on 8 processors?

Solution:

p	2	3	4	5	6	7	8
$\Psi(n,p)$	1.87	2.61	3.23	3.73	4.14	4.46	4.71
e	0.07	0.075	0.08	0.085	0.09	0.095	0.1

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

The Isoefficiency Metric

- **Parallel system**: A parallel program executing on a parallel computer
- **Scalability**: the scalability of a parallel system is a measure of its ability to increase performance as the number of processors increases.
- A scalable system should maintain efficiency as # of processors is increased.
- Isoefficiency: way to measure scalability.

- Let $T_0(n,p)$ be the total amount of time spent by all processes doing work not done by the sequential algorithm:

$$T_0(n,p) = (p-1)\sigma(n) + p\kappa(n,p)$$

  - $T_0(n,p)$ can also be interpreted as:
    $p \times$ (Parallel execution time) $-$ (Sequential execution time)
    - This is the total amount of overhead

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

$$\Rightarrow \Psi(n,p) \leq \frac{p(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+(p-1)\sigma(n)+p\kappa(n,p)}$$

$$\Rightarrow \Psi(n,p) \leq \frac{p(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+T_0(n,p)}$$

- Let $T(n,1)$ be the time of the sequential algorithm for solving the problem. $\varepsilon(n,p) \leq \frac{1}{1+\frac{T_0(n,p)}{\sigma(n)+\varphi(n)}} \leq \frac{1}{1+\frac{T_0(n,p)}{T(n,1)}}$

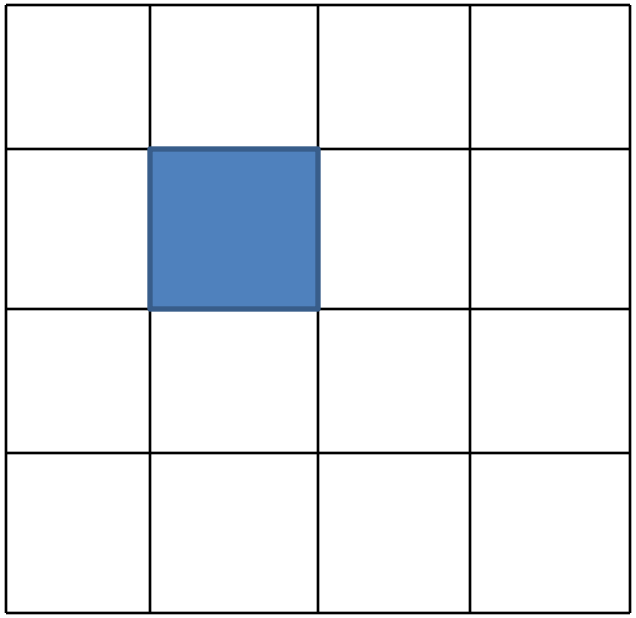
$$T(n,1) \geq \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)} T_0(n,p)$$

Isoefficiency Relation:

Suppose a parallel system has efficiency $\varepsilon(n,p)$. Define $\mathrm{C} = \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)}$.

In order to maintain the same level of efficiency as the number of processors increases, n must be increases so that the following inequality is satisfied: $T(n,1) \geq CT_0(n,p)$

Example: Explicit Finite Difference



- The problem is solved on a $n \times n$ grid.
- p processors are used.
- Each processor is responsible for a subgrid of size $\left(\frac{n}{\sqrt{p}}\right) \times \left(\frac{n}{\sqrt{p}}\right)$.
- During each time step, every processor sends boundary values to its four neighbors; the time needed for communication is $O\left(\frac{n}{\sqrt{p}}\right)$.

Find the isoefficiency.

Solution: The time complexity of the serial algorithm for solving this problem is

$$T(n,1) = O(n^2).$$

$$T_0(n,p) = O\left(\frac{n}{\sqrt{p}}\right)p.$$

The isoefficiency relation is

$$n^2 \geq Cp(\frac{n}{\sqrt{p}}) \Longrightarrow n \geq C\sqrt{p}$$

Reference

- M.J. Quinn. Parallel Programming in C with MPI and OpenMP