Basics of parallelism

Basic approaches to parallelism

Speedup and Efficiency

Material in this presentation from textbook:

Georg Hager and Gerhard Wellein, Introduction to High Performance Computing for Scientists and Engineers, Chapman & Hall/CRC Computational Science Series ISBN 978-1-4398-1192-4

Learning Objectives

- Basic approaches to parallelize problems
- Predict performance of parallel programs
- Understand barriers to higher performance

Outline

- Why parallelize?
- Parallelism
- Parallel scalability
  - Limitations
  - Scalability metrics
    - General speedup formula
    - Amdahl's Law
    - Gustafson-Barsis' Law
    - Karp-Flatt metric

Why parallelize

- Single core too slow for solving the problem in a “reasonable” time
  - “Reasonable” time: overnight, over lunch, duration of a PhD theses
- Memory requirements
  - Larger problem
  - More physics
  - More particles

Parallelism

- For multi-core or multi-node computers
- Data parallelism
  - Single Program Multiple Data (SPMD)
  - Same code is executed on all processors
  - Data is different on the nodes
- Functional parallelism
  - Splitting problem in separate subtasks
  - Multiple Program Multiple Data (MPMD)
  - Subtask could be SPMD
  - Difficult to load balance if subtasks have different performance properties

Examples Data Parallelism

P1	`do i=1,500`
	`  a(i)=c*b(i)`
	`enddo`
		`do i=1,1000`
		`  a(i)=c*b(i)`
		`enddo`

P2	`do i=501,1000`
	`  a(i)=c*b(i)`
	`enddo`	

Speedup Formula

$$\text{Speedup} = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

Execution Time Components

- Inherently sequential computations: $\mathrm{s(n)}$
- Potentially parallel computations: $\mathrm{p(n)}$
- Communication operations: $\mathrm{c(n,\ \ p)}$

- Speedup expression:

$$S \leq \frac{s+p}{s(n)+p/N+c}$$

Parallel Overhead

- Overhead because of
  - Startup time
  - Synchronizations
  - Communication
  - Overhead by libraries, compilers
  - Termination time

Efficiency

$$\text{Efficiency} = \frac{\text{Sequential execution time}}{\text{Processors} \times \text{Parallel execution time}}$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Processors}}$$

Amdahl's Law

- How much can we improve the time to solution
  - Fixed problem size

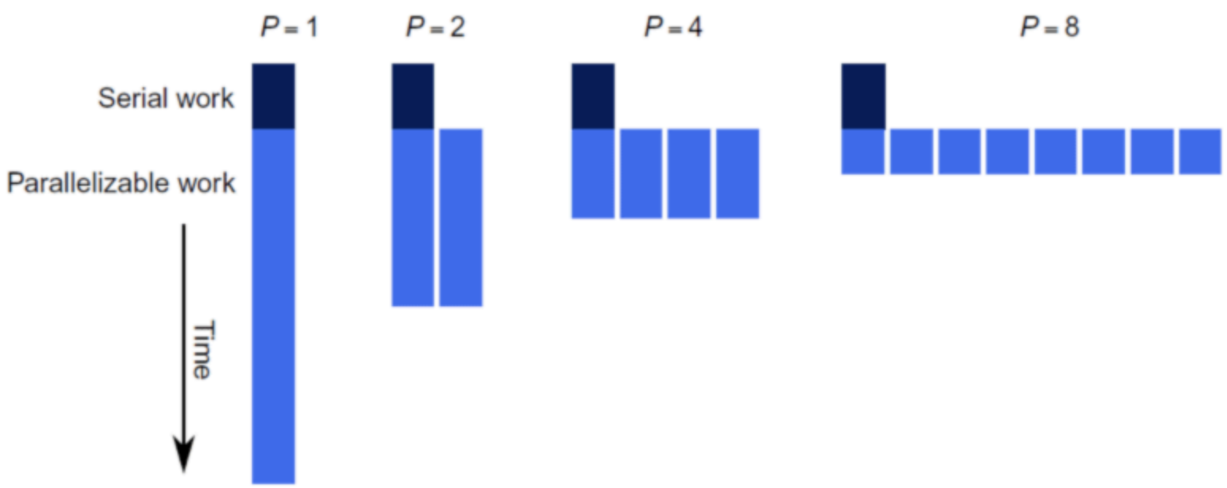
$$
\begin{aligned}
T_f^s &= s + p \\
T_f^p &= s + \frac{p}{N} \\
S &= \frac{1}{s + \frac{1-s}{N}}
\end{aligned}
$$

  - How much can the serial portion be if we want a speedup of 90 on 100 processors

$$90 = \frac{1}{s + \frac{1-s}{100}} \rightarrow s = 0.0001$$

Limitations of Amdahl's Law

- Ignores overhead
- Overestimates speedup achievable

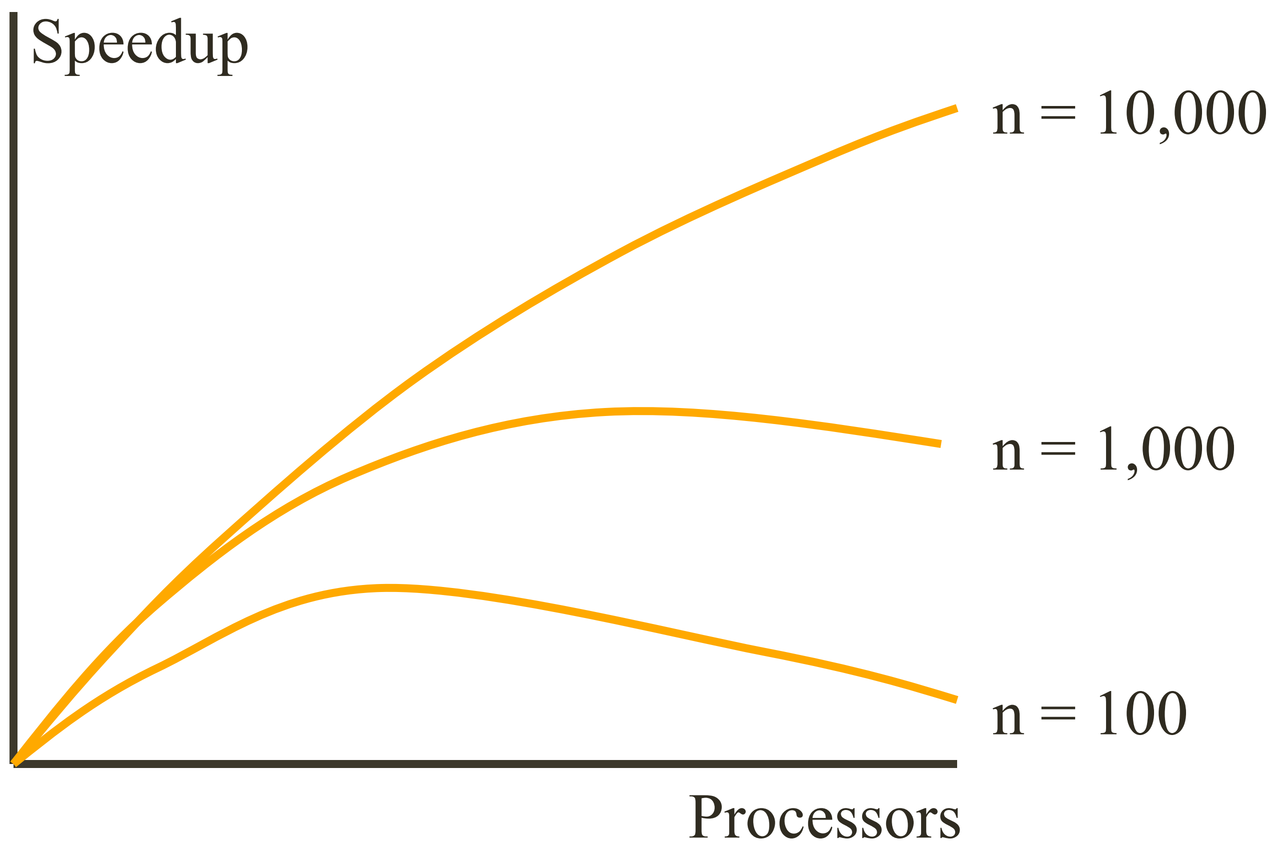


http://www.drdobbs.com/parallel/amdahls-law-vs-gustafson-barsis-law/240162980?pgno=2

Amdahl Effect

- Typically overhead has lower complexity than parallel work
- As *problem size* increases, parallel work dominates overhead
- As *n* increases, speedup increases

Illustration of Amdahl Effect



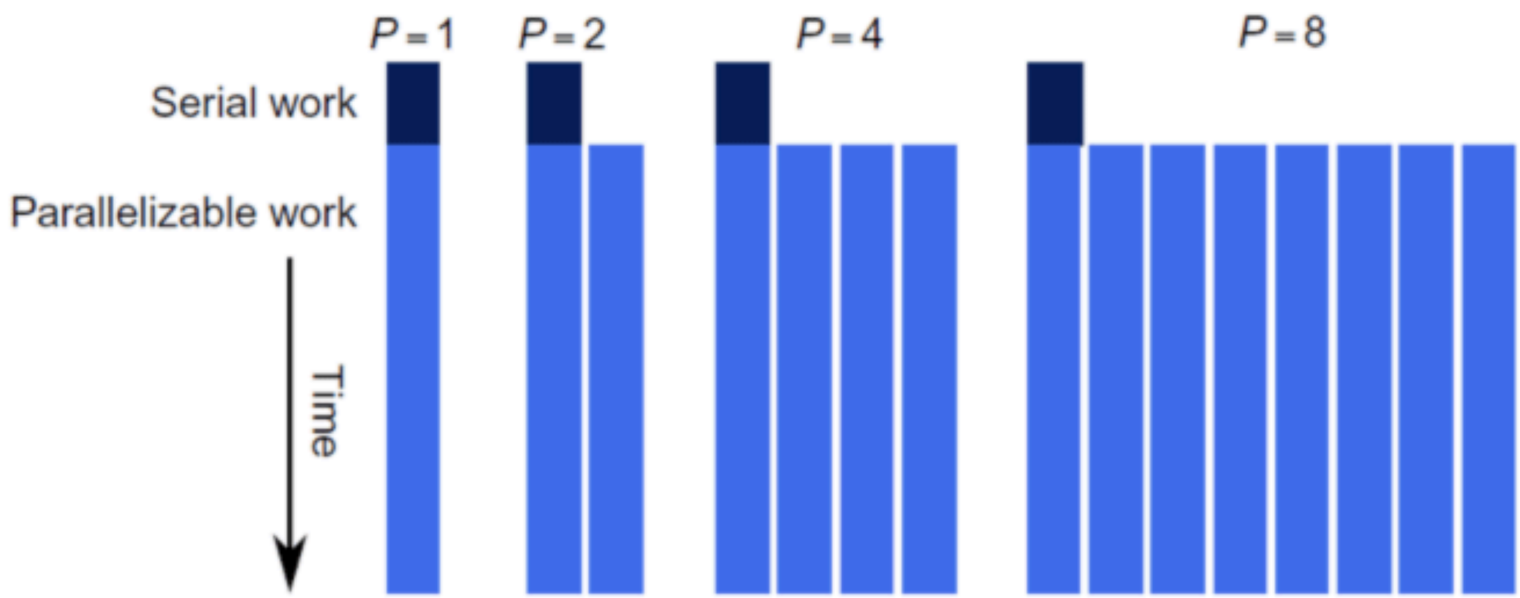
Review of Amdahl's Law

- Treats problem size as a constant
- Shows how execution time decreases as number of processors increases
- Strong scaling
  - Problem size is fixed
  - Number of processor increases

Another Perspective

- We often use faster computers to solve larger problem instances
- Let’s treat time as a constant and allow problem size to increase with number of processors
- “…speedup should be measured by scaling the problem to the number of processors, not by fixing the problem size” – John Gustafson

Gustafson-Barsis's Law



$$SS(N) = \frac{s+p*N}{s+p} = N + (1-N)s$$

http://www.drdobbs.com/parallel/amdahls-law-vs-gustafson-barsis-law/240162980?pgno=2

Gustafson-Barsis's Law

- Begin with parallel execution time
- Estimate sequential execution time to solve same problem
- Problem size is an increasing function of N
- Predicts **scaled speedup**
- Weak Scaling

Example 1

- An application running on 10 processors spends 3% of its time in serial code. What is the scaled speedup of the application?

$$S = 10 + (1 - 10)(0.03) = 10 - 0.27 = 9.73$$

…except 9 do not have to execute serial code

Execution on 1 CPU takes 10 times as long…

Example 2

- What is the maximum fraction of a program's parallel execution time that can be spent in serial code if it is to achieve a scaled speedup of 7 on 8 processors?

$$90 = 100 + (1 - 100)s \Rightarrow s \approx 0.1$$

The Karp-Flatt Metric

- Amdahl's Law and Gustafson-Barsis' Law ignore overhead
- They can overestimate speedup or scaled speedup
- Karp and Flatt proposed another metric

Experimentally Determined Serial Fraction

$$e = \frac{s+c}{s+p}$$

$$\frac{\text{Inherently serial component of parallel computation + processor communication and synchronization overhead}}{\text{Single processor execution time}}$$

$$e = \frac{1/S - 1/p}{1 - 1/p}$$

Experimentally Determined Serial Fraction

- 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

Example 1

p	2	3	4	5	6	7	8
ψ	1.8	2.5	3.1	3.6	4.0	4.4	4.7

What is the primary reason for speedup of only 4.7 on 8 CPUs?

e	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Since *e* is constant, large serial fraction is the primary reason.

Example 2

p	2	3	4	5	6	7	8
ψ	1.9	2.6	3.2	3.7	4.1	4.5	4.7

What is the primary reason for speedup of only 4.7 on 8 CPUs?

e	0.070	0.075	0.080	0.085	0.090	0.095	0.100

Since *e* is steadily increasing, overhead is the primary reason.

Pop Quiz

p	4	8	12
ψ	3.9	6.5	?

- Is this program likely to achieve a speedup of 10 on 12 processors?

Summary (1/3)

- Performance terms
  - Speedup
  - Efficiency
- Model of speedup
  - Serial component
  - Parallel component
  - Communication component

Summary (2/3)

- What prevents linear speedup?
  - Serial operations
  - Communication operations
  - Process start-up
  - Imbalanced workloads
  - Architectural limitations

Summary (3/3)

- Analyzing parallel performance
  - Amdahl's Law
  - Gustafson-Barsis' Law
  - Karp-Flatt metric