NAU NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing, and Cyber Systems

Measures of parallel performance

CS450/CS550: Parallel Programming

Week 8

Parallel performance

- Parallel programs are developed to create a faster version of the corresponding sequential programs
- Speedup ψ is the ratio between sequential execution time τ_s and parallel execution time τ_p:

$$\psi = \frac{\tau_s}{\tau_p}$$

- The efficiency ε of a parallel program is a measure of a processor utilization (speedup divided by the number p of processors used):

$$\varepsilon = \frac{\psi}{p} = \frac{\tau_s}{p \cdot \tau_p}$$

Speedup model

Basic time characteristics for the problem of size on n:

- $\sigma(n)$ is the time for the necessary sequential computations
- $\varphi(n)$ is the time of the computation that can be executed in parallel
- $\kappa(n,p)$ is the time required for parallel overhead for p processors

In that case:

- sequential execution time $\tau_s = \sigma(n) + \varphi(n)$
- parallel execution time $\tau_p = \sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)$
- the speedup $\psi(n,p)$ achieved solving a problem of size n on p processors:

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

Efficiency model

The efficiency ε of a parallel program to solve the problem of size on n on p processors:

$$\varepsilon(n,p) = \frac{\psi(n,p)}{p}$$

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

$$\varepsilon(n,p) \le \frac{\sigma(n)+\varphi(n)}{p\cdot\sigma(n)+\varphi(n)+p\cdot\kappa(n,p)}$$

If $p = 1$, then $\kappa(n,p) = 0 \Longrightarrow \varepsilon(n,1) = \frac{\sigma(n)+\varphi(n)}{1\cdot\sigma(n)+\varphi(n)+1\cdot 0} = 1$

If $p > 1$, then $\kappa(n,p) > 0 \Longrightarrow \varepsilon(n,p) < \frac{\sigma(n)+\varphi(n)}{(p-1)\cdot\sigma(n)+\sigma(n)+\varphi(n)+p\cdot\kappa(n,p)} =$

$$= \frac{1}{1+\frac{(p-1)\cdot\sigma(n)+p\cdot\kappa(n,p)}{\sigma(n)+\varphi(n)}} < 1$$

Amdahl's Law

- Since $\kappa(n,p) \geq 0$

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

- Let f denote the sequential portion of the computation:

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

- In that case:

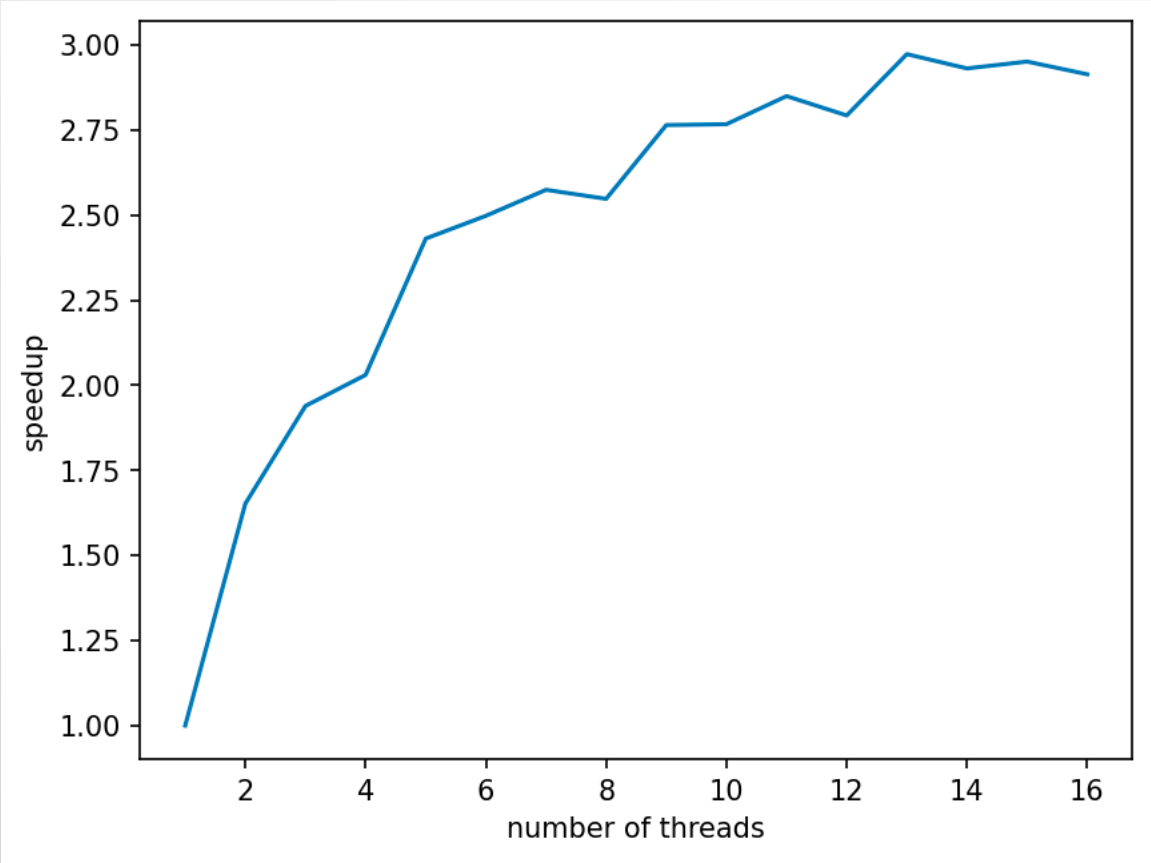
$$\sigma(n) + \varphi(n) = \frac{\sigma(n)}{f}, \varphi(n) = \frac{\sigma(n)}{f} - \sigma(n) = \sigma(n) \cdot \left(\frac{1}{f} - 1\right)$$

$$\psi(n,p) \leq \frac{\frac{\sigma(n)}{f}}{\sigma(n)+\frac{\sigma(n)\cdot\left(\frac{1}{f}-1\right)}{p}} \Rightarrow \psi(p) \leq \frac{\frac{1}{f}}{1+\frac{\frac{1}{f}-1}{p}} \Rightarrow \psi(p) \leq \frac{1}{f+\frac{1-f}{p}}$$

Amdahl's Law

- Amdahl's Law provides an upper bound on the speedup achievable by applying a certain number of processors to solve the problem in parallel
- If f is a fraction of operations that must be performed sequentially, then the **maximum** speedup ψ achievable by a parallel computer with p processors equals $\frac{1}{f + \frac{1-f}{p}}$
- Amdahl's Law determines speedup by taking a serial computation and predicting how quickly that computation could execute on multiple processors

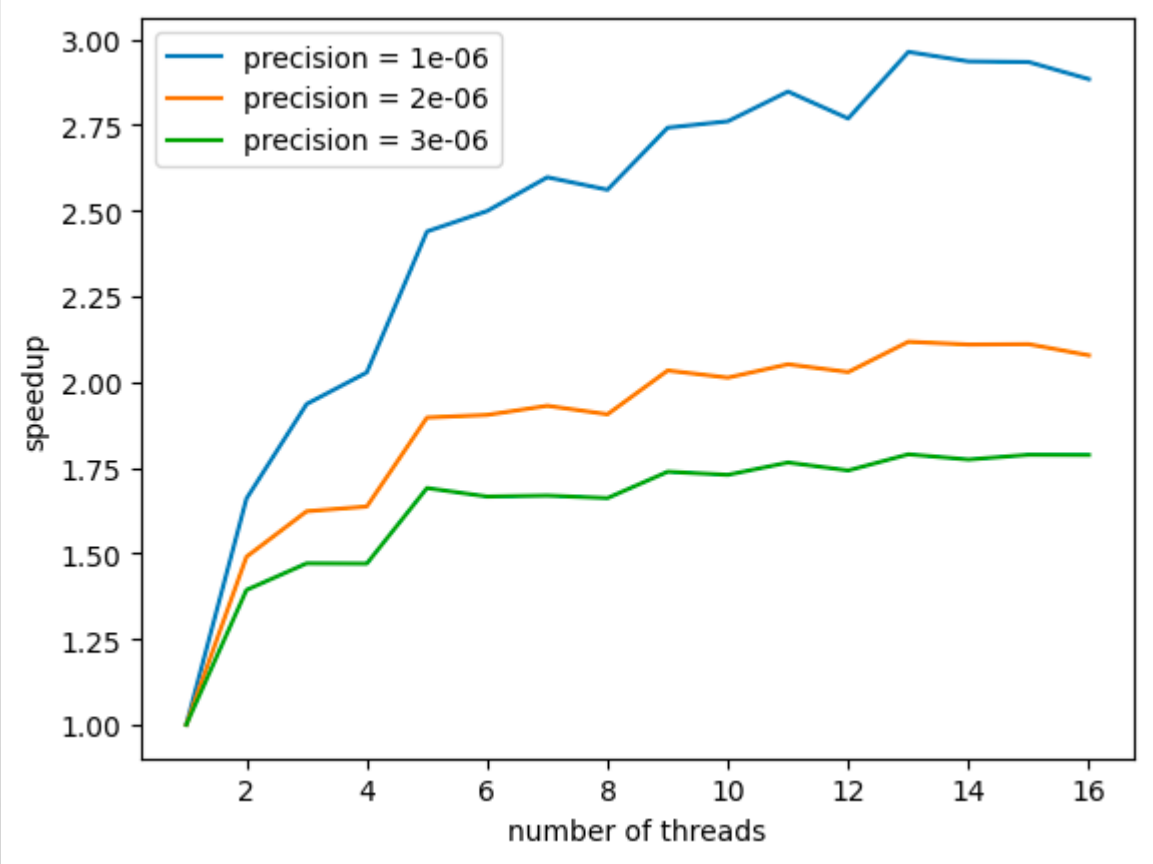
Amdahl's Law: numeric example



The Amdahl effect

- Increasing the size of the problem n increases the computation time faster than it increases the communication time
- For a fixed number of processors, speedup is an increasing function of the problem size

The Amdahl effect: numeric example



Gustafson-Barsis's law

- Let s denote the fraction of time spent in the parallel computation performing sequential operations:

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

- In this case: $\sigma(n) = \frac{\varphi(n)}{p} \cdot \frac{s}{1-s}$
- The program speedup may be evaluated as:

$$\psi(n,p) \le \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \frac{\varphi(n)}{p}} \Rightarrow \psi(n,p) \le \frac{\frac{\varphi(n)}{p} \cdot \frac{s}{1-s} + \varphi(n)}{\frac{\varphi(n)}{p} \cdot \frac{s}{1-s} + \frac{\varphi(n)}{p}} \Rightarrow$$

$$\psi(p) \le \frac{\frac{s}{1-s} + p}{\frac{s}{1-s} + 1} \Rightarrow \psi(p) \le \frac{s + p \cdot (1-s)}{s + 1 - s} \Rightarrow \psi(p) \le s + p \cdot (1-s)$$

Gustafson-Barsis's Law

- Given a parallel program that solves some problem using p processors, let s denote the fraction of total execution time spent in serial code. Then, the maximum speedup ψ achievable by this program is $(1 - s) \cdot p + s$
- Gustafson-Barsis's Law determines how much faster the parallel computation is than the same computation executing on a single processor
- The speedup predicted by Gustafson-Barsis's Law is referred to as **scaled speedup** because it allows the problem size to be an increasing function of the number of processors

The Karp-Flatt metric

- Parallel execution time:

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

$$\tau_p(n,1) = \sigma(n) + \varphi(n)$$

- Experimentally determined serial fraction of the parallel computation:

$$e = \frac{\sigma(n) + \kappa(n,p)}{\tau_p(n,1)} \Longrightarrow \sigma(n) + \kappa(n,p) = e \cdot \tau_p(n,1)$$

$$1 - e = \frac{\varphi(n)}{\tau_p(n,1)} \Longrightarrow \varphi(n) = (1-e) \cdot \tau_p(n,1)$$

- Parallel execution time in this case:

$$\tau_p(n,p) = e \cdot \tau_p(n,1) + \frac{\varphi(n)}{p} \Longrightarrow \tau_p(n,p) = e \cdot \tau_p(n,1) + \frac{1-e}{p} \cdot \tau_p(n,1)$$

The Karp-Flatt metric

- The program speedup:

$$\psi = \frac{\tau_p(n,1)}{\tau_p(n,p)} \Longrightarrow \tau_p(n,1) = \psi \cdot \tau_p(n,p)$$

- Parallel execution time in this case:

$$\tau_p(n,p) = e \cdot \psi \cdot \tau_p(n,p) + \frac{1-e}{p} \cdot \psi \cdot \tau_p(n,p)$$

- Then we obtain:

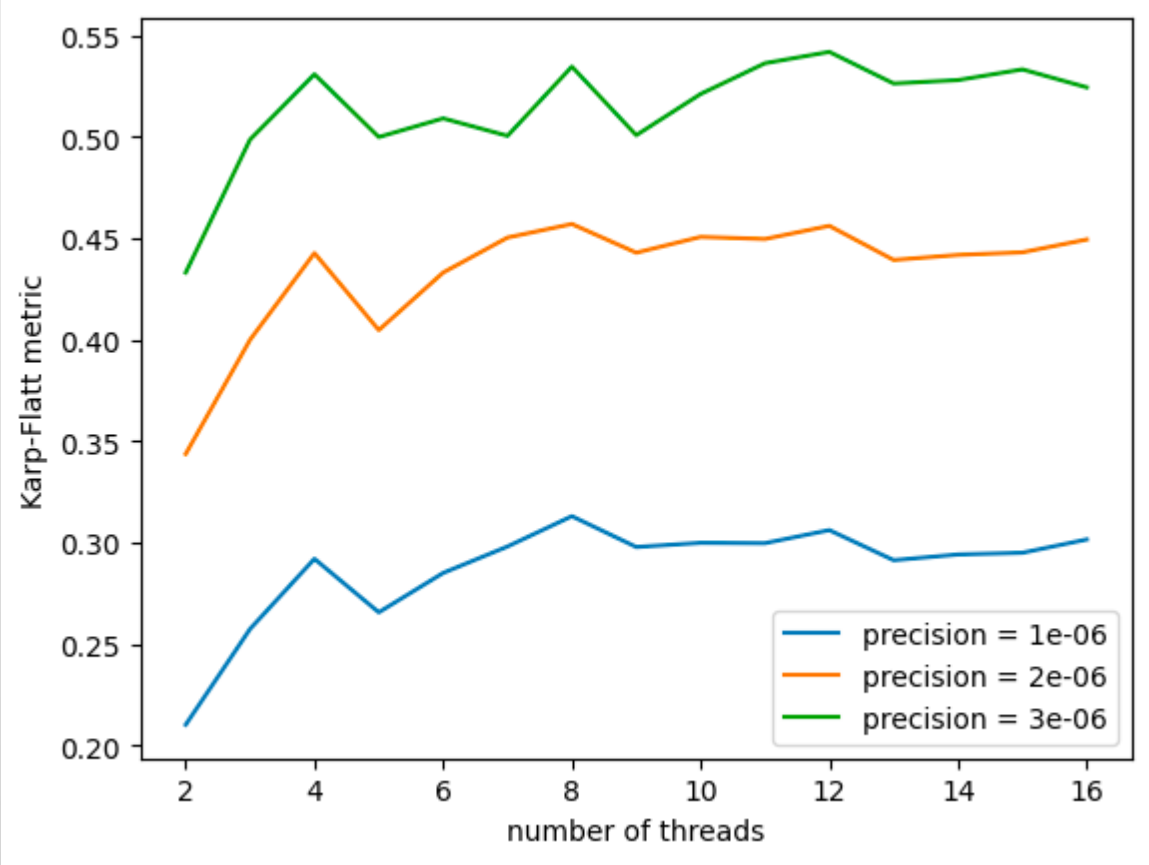
$$1 = e \cdot \psi + \frac{1-e}{p} \cdot \psi$$

$$e = \frac{\frac{1}{\psi} - \frac{1}{p}}{1 - \frac{1}{p}}$$

The Karp-Flatt metric

- Given a parallel computation exhibiting speedup ψ on p processors ($p > 1$), the experimentally determined serial fraction e is defined to be $e = \frac{1/\psi - 1/p}{1 - 1/p}$
- The experimentally determined serial fraction e is a valuable metric, because:
  - it considers parallel overhead $\kappa(n, p)$ ignored by Amdahl's Law and Gustafson-Barsis's Law
  - it can be used to detect other sources of inefficiency not considered in the presented speedup model

The Karp-Flatt metric: numeric example



Conclusions

- Amdah's Law helps to decide whether a program merits parallelization
- Gustafson-Barsis's Law allows the evaluation of the performance of a parallel program
- The Karp-Flatt metric shows whether the principal barrier to speedup is the amount of sequential code or parallel overhead

Recommended literature

- Quinn, Michael J. (2004) Parallel programming in C with MPI and OpenMP. 1st ed. McGraw Hill