Carleton UNIVERSITY

Canada's Capital University

RCS Workshop III

Introduction to Parallel Computing using Matlab

Overview

1. Parallel vs. sequential computing
2. Limitations of parallel computing
3. Types of parallel workers
4. Bottlenecks and overhead
5. Writing parallel code
6. MATLAB: Parallel Programming
7. MATLAB: Case Studies
8. MATLAB: Vectorization
9. MATLAB: Best Practices

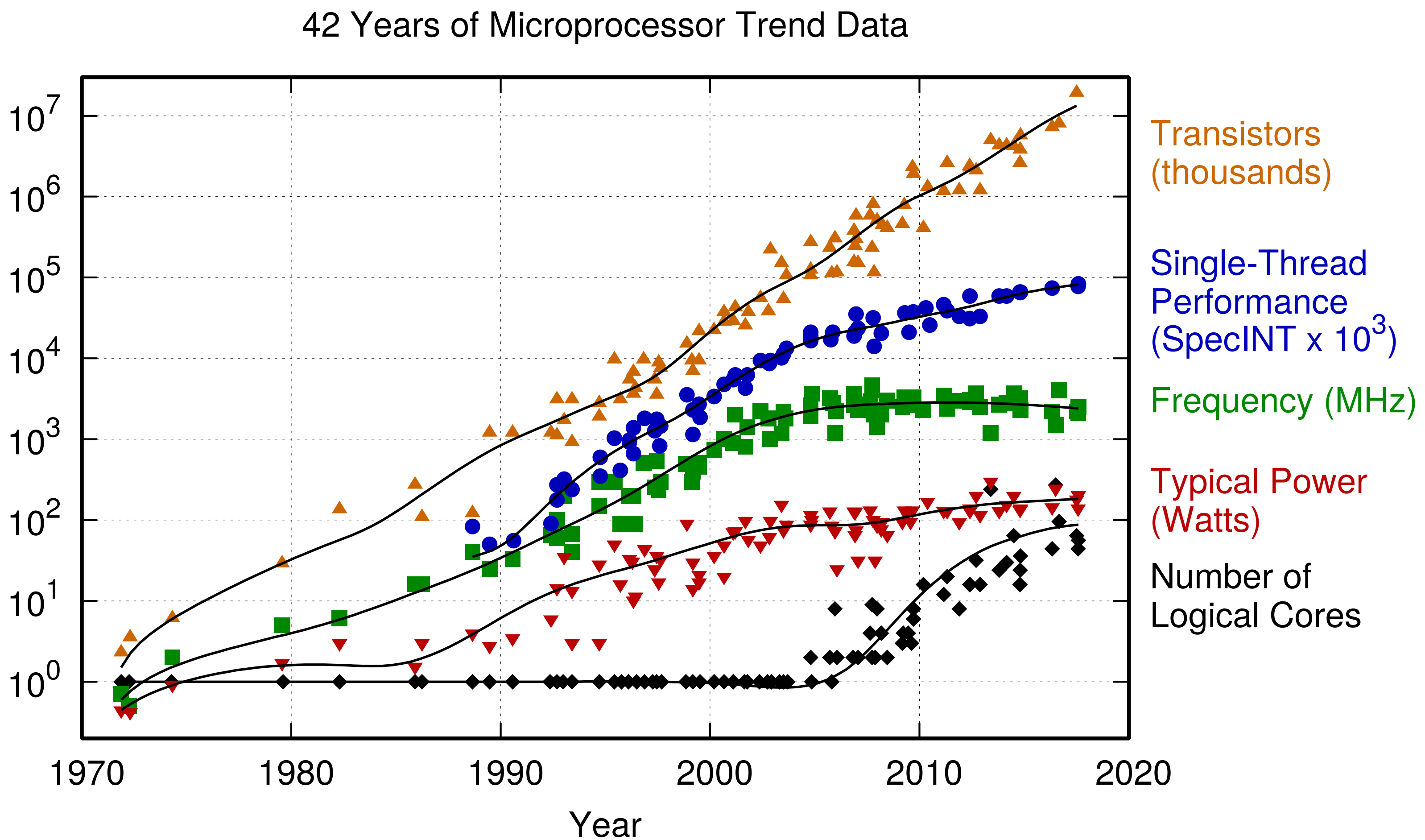
1. PARALLEL VS SEQUENTIAL COMPUTING

Carleton UNIVERSITY
Canada's Capital University

What is Parallel Computing?

- **Sequential**
  - One processor core
  - Step through instruction
- **Parallel**
  - Multiple cores
  - Independently executing instructions
- **Batch**

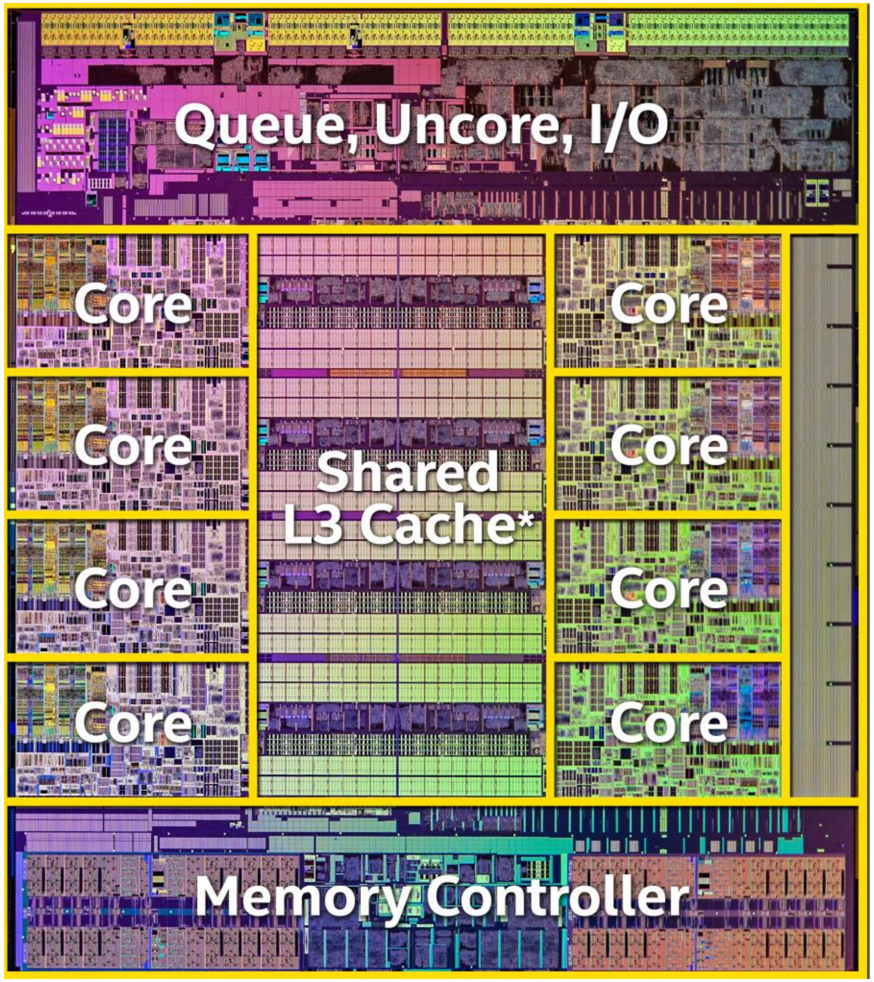
Why Parallel Computing?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Parallel Computing Hardware

- **Multicore CPUs**
  - 28+ cores per CPU
  - >7-8 billion transistors
- **Compute clusters and clouds**
  - Thousands of CPUs distributed over different compute nodes.



Parallel Computing Hardware (2)

- **Accelerator Cards**
  - Hundreds or thousands of "cores".
  - 10-20+ billion transistors
- **GPGPU**
  - AMD, Nvidia
  - CUDA, OpenCL, etc.
- **Intel's Xeon Phi**
  - Runs x86 code.

Parallel Computing Hardware (3)

2. LIMITATIONS OF PARALLEL SPEEDUP

How Fast is Parallel?

- **Suppose sequential code runs in 60 minutes on 1 processor:**
  - How fast with 2 processors?
  - How fast with huge number of processors?

What is Speedup?

- $T(1)$time on 1 processor (sequential).
- $T(p)$time on *p* processors (parallel).

- Speedup is defined as:

$$S(p) = \frac{T(1)}{T(p)}$$

- Linear speedup is optimal*: $S(p) \leq p$

*there are exceptions…

Carleton UNIVERSITY
Canada's Capital University

Speedup Example

- **What if only Section 3 can be parallelized?**

Code Section	Sequential Time	Description
Section 1	4 minutes	Initialize
Section 2	5 minutes	Read input files
Section 3	60 minutes	Compute results
Section 4	5 minutes	Output results to file
Section 5	1 minute	Cleanup variables

Speedup Example (2)

- **What if only Section 3 can be parallelized?**

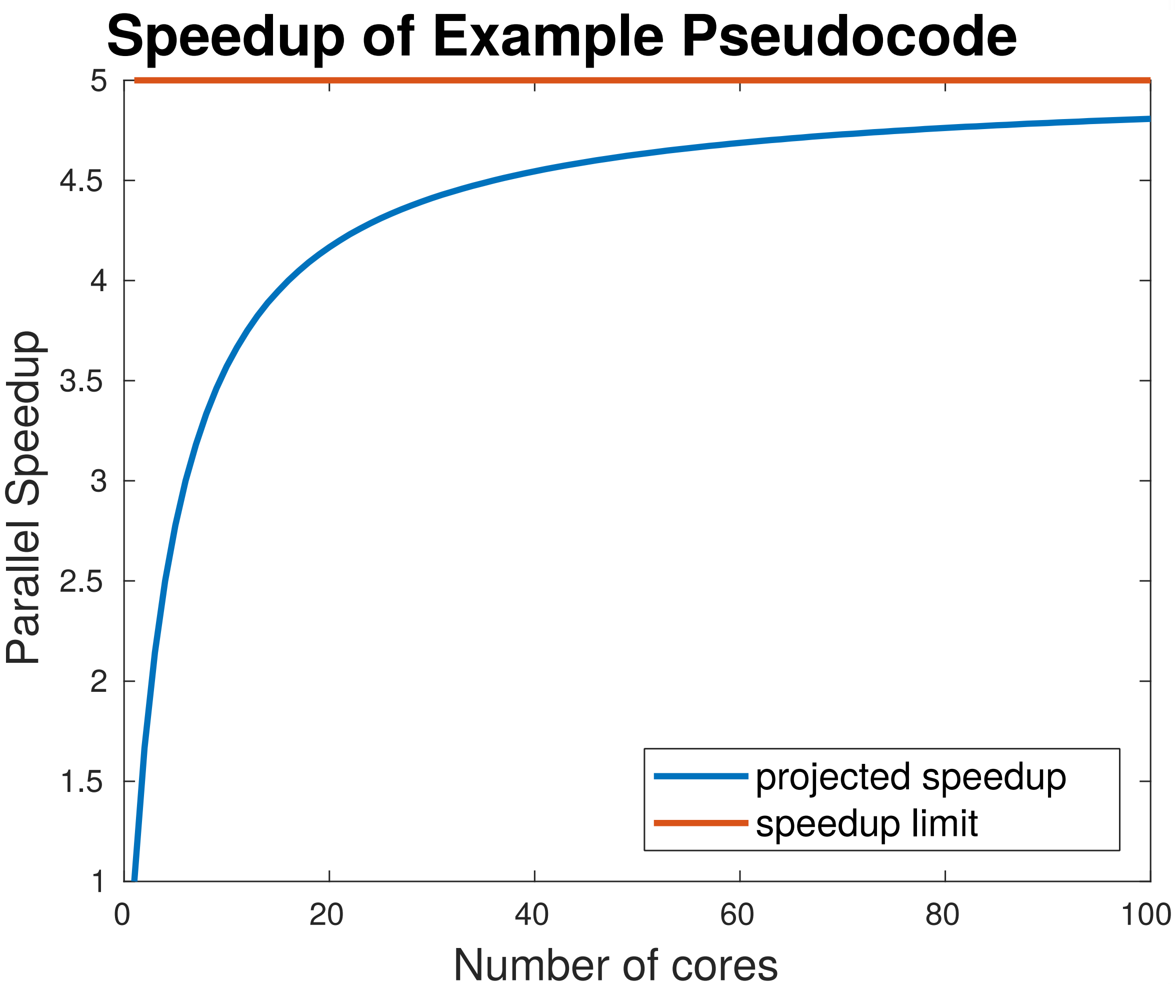
Code Section	Sequential Time	Description
Section 1	4 minutes	Initialize
Section 2	5 minutes	Read input files
Section 3	60 minutes	Compute results
Section 4	5 minutes	Output results to file
Section 5	1 minute	Cleanup variables

$$T(1) = 4 + 5 + 60 + 5 + 1 = 75$$

$$T(p) = 15 + \frac{60}{p}$$

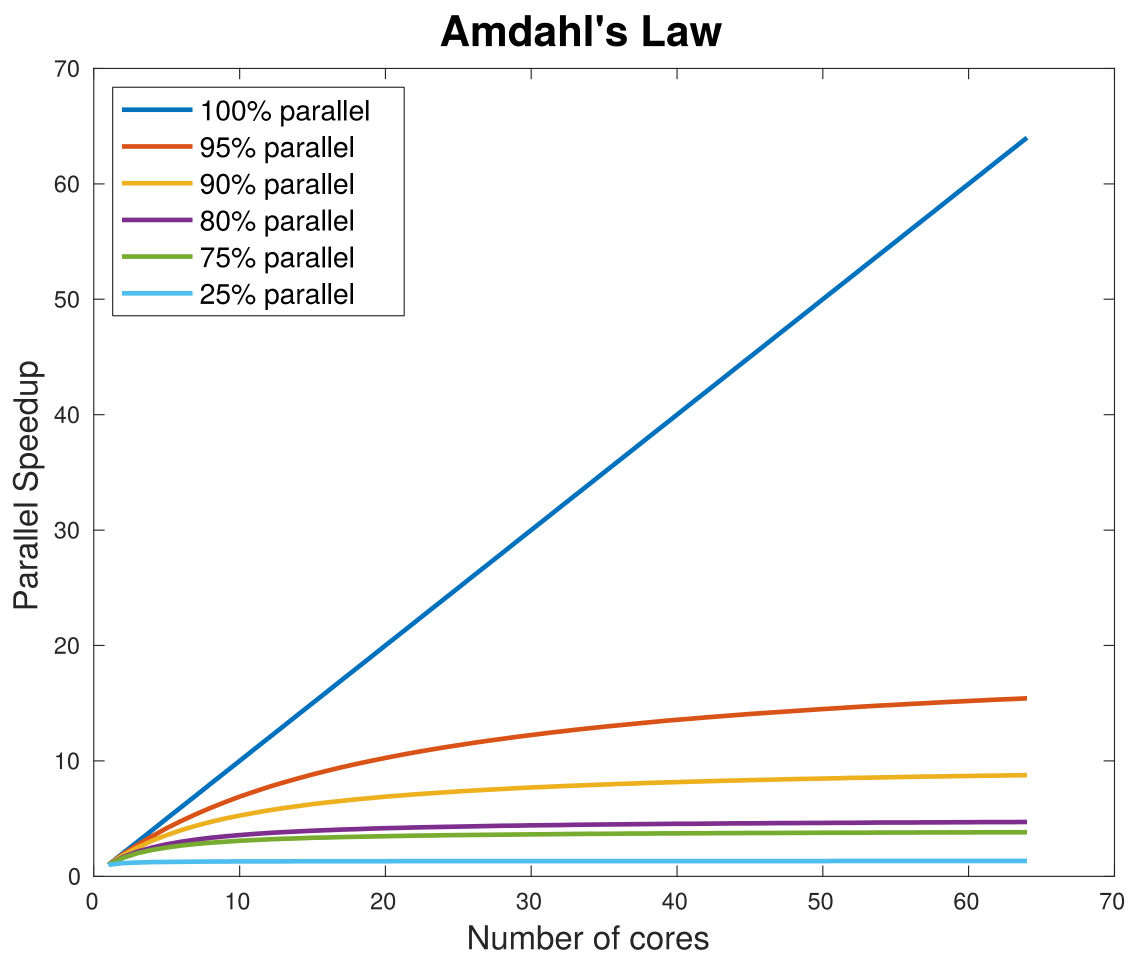
Speedup Example (3)

- $T(1) = 75$
- $T(p) = 15 + \frac{60}{p}$
- $S(p) = \frac{5}{1+\frac{4}{p}}$
- $S(4) = \frac{5}{1+\frac{4}{4}} = \frac{5}{2}$



Amdahl's Law

- $$S(p) = \frac{1}{(1-k)+\frac{k}{p}}$$
- Where *k* is the fraction of work that can be done in parallel.

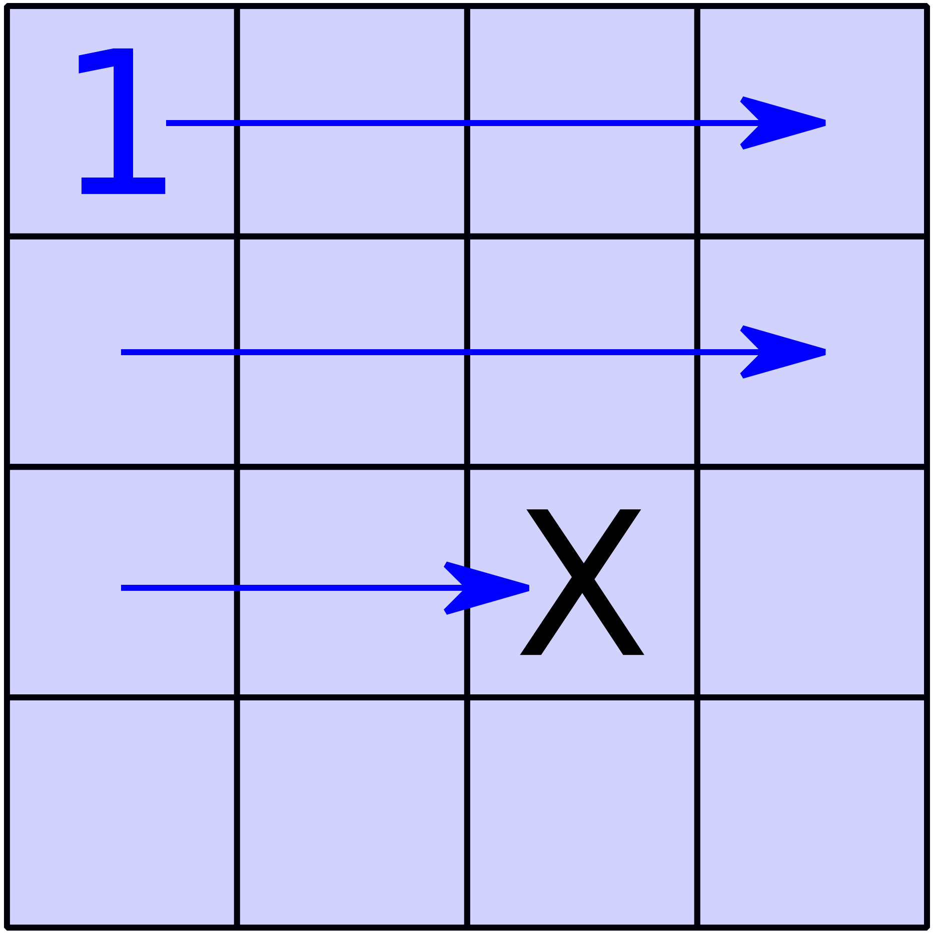


Superlinear Speedup

- Some problems seem to break Amdahl's law where $S(p) > p$
- There are several possible reasons:
  - Parallel version is using a different algorithm.
  - Sequential code is not optimal (comparing a "good" parallel implementation vs. a "bad" sequential one).
  - Search-type problems where more "searchers" yields answer quicker in some scenarios (example to follow).
  - Hardware is being used more efficiently (e.g. more RAM available, cache, more disks, etc).
- Compare apples to apples!

Superlinear Example

$$T(1) = 11$$

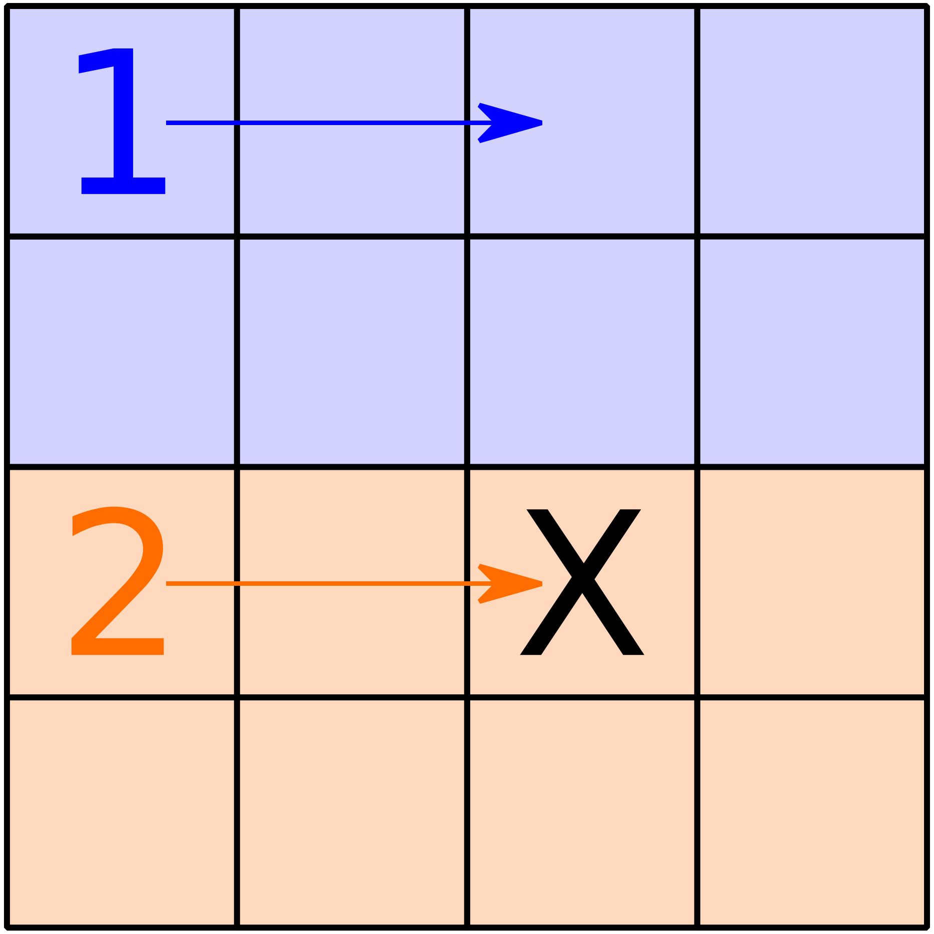


Superlinear Example (2)

$$T(1) = 11$$

$$T(2) = 3$$

$$S(2) = \frac{11}{3} = 3.7$$

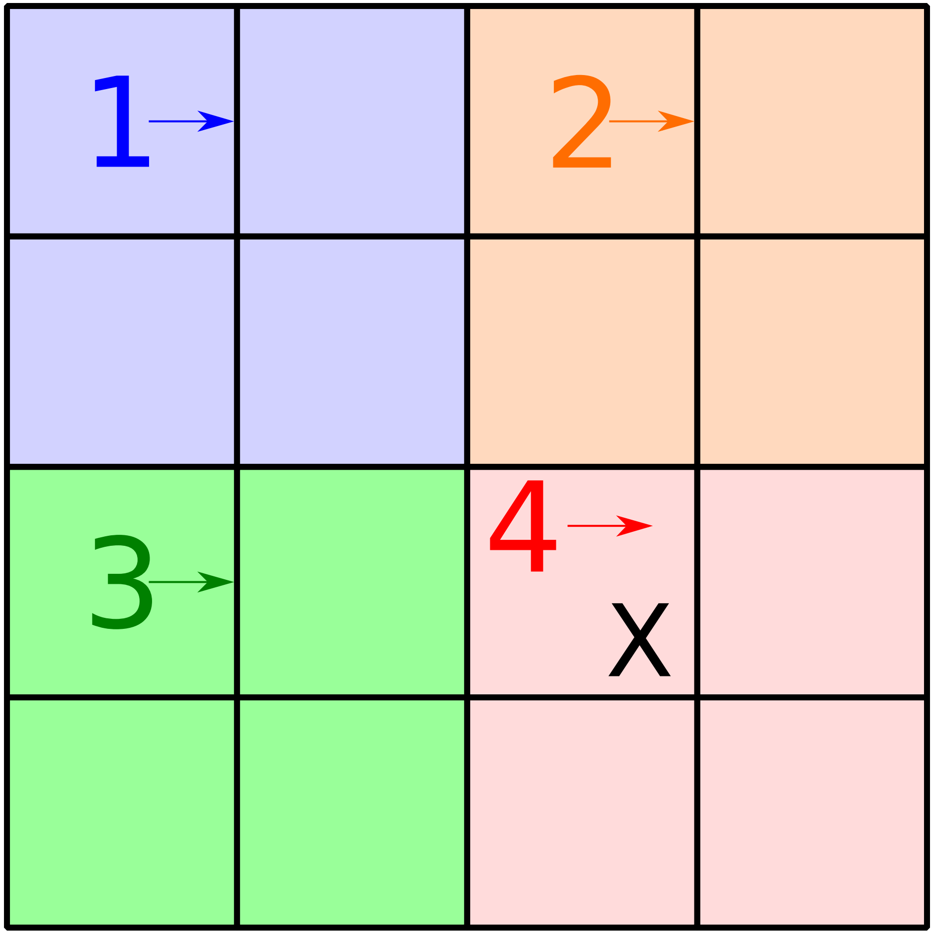


Superlinear Example (3)

$$T(1) = 11$$

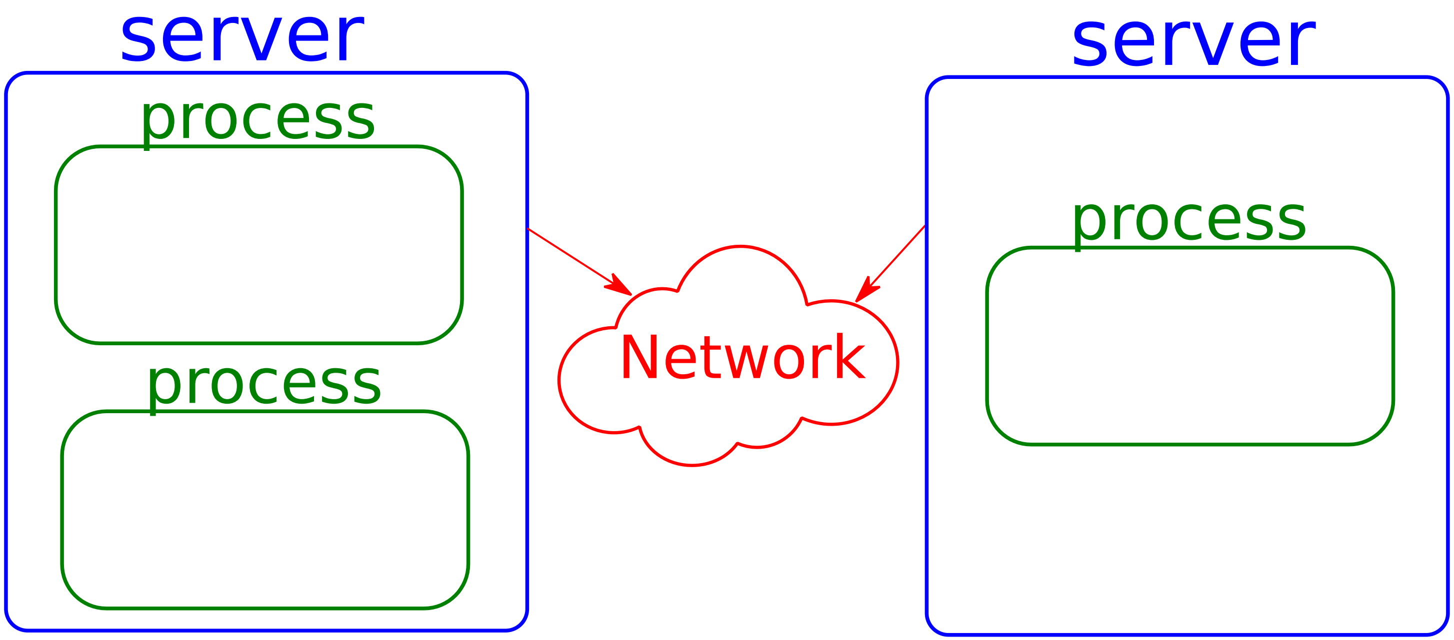
$$T(4) = 1$$

$$S(4) = \frac{11}{1} = 11$$

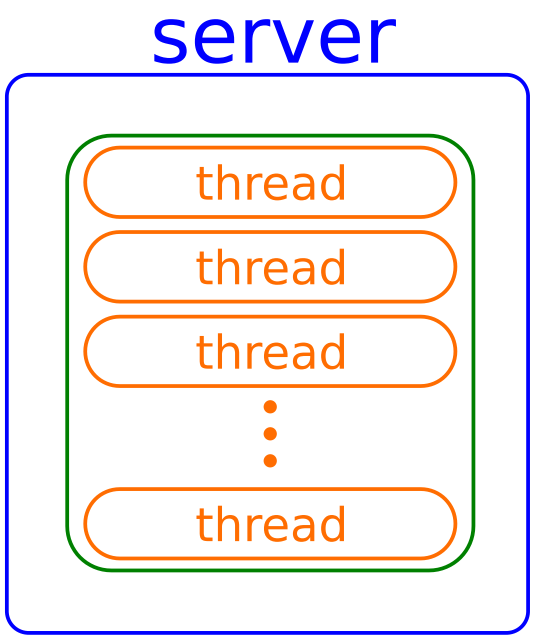


3. TYPES OF PARALLEL WORKERS

Process



Thread



4. BOTTLENECKS AND OVERHEAD

Bottlenecks and Overhead

- **Communication Overhead**
  - Communication vs. Computation
- **Hardware Bottlenecks**
  - RAM
  - Disk
  - Network

5. WRITING PARALLEL CODE

Writing Parallel Code

1. Working code
2. Reproducible code (correctness)
3. Profile code
4. Break dependencies
5. Convert to parallel
6. Measure improvement

Parallel Libraries & Tools

- **Two common ways to write parallel code already mentioned:**
  - OpenMP (multithreading)
  - MPI (distributed computing)

Parallel Libraries & Tools (2)

- **There are several others:**
  - Julia (https://julialang.org/)
  - OpenCL
  - CUDA (specific for Nvidia GPU)
- **Several tools and software support parallel:**
  - Matlab
  - Domain-specific tools (Tensorflow, Caffe, etc).

6. MATLAB: PARALLEL PROGRAMMING

Temporary Variables

```
N = 100;
b = pi;
S = zeros(N);
R = 0;
parfor i = 1:N
    T = rand(1,N);
    S(i,:) = b * T;
    R = R + max(T);
end
```

temporary

Loop Variables

```
N = 100;
b = pi;
S = zeros(N);
R = 0;
parfor i = 1:N
    T = rand(1,N);
    S(i,:) = b * T;
    R = R + max(T);
end
```

loop

Sliced Variables

sliced

```
N = 100;
b = pi;
S = zeros(N);
R = 0;
parfor i = 1:N
    T = rand(1,N);
    S(i,:) = b * T;
    R = R + max(T);
end
```

Broadcast Variables

```
N = 100;
b = pi;
S = zeros(N);
R = 0;
parfor i = 1:N
    T = rand(1,N);
    S(i,:) = b * T;
    R = R + max(T);
end
```

broadcast

Reduction Variables

```
N = 100;
b = pi;
S = zeros(N);
R = 0;
parfor i = 1:N
    T = rand(1,N);
    S(i,:) = b * T;
    R = R + max(T);
end
```

reduction

Transparency

- **Cannot manipulate workspace**
- **Commands not allowed:**
  - save()
  - load()
  - clear()
  - eval()
  - Etc…

7. MATLAB CASE STUDIES

8. MATLAB VECTORIZATION

9. MATLAB BEST PRACTICES

MATLAB: Best Practices

- Preallocation
- Loop Constants
- Vectorization
- Parallelization
- MATLAB Mex Compilation
- Column-Major Memory Access