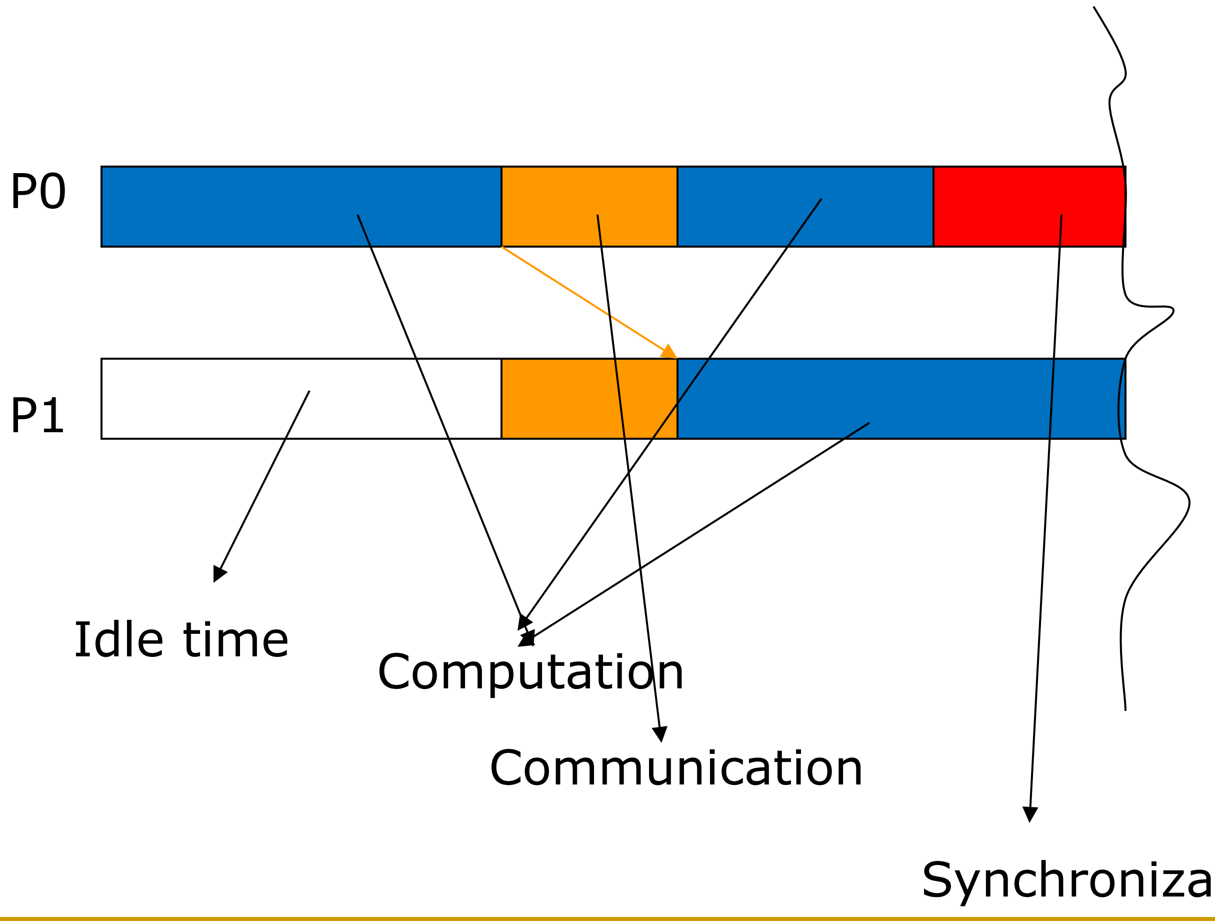
Parallelization Principles

Sathish Vadhiyar

Parallel Programming and Challenges

- Recall the advantages and motivation of parallelism
- But parallel programs incur overheads not seen in sequential programs
  - Communication delay
  - Idling
  - Synchronization

Challenges

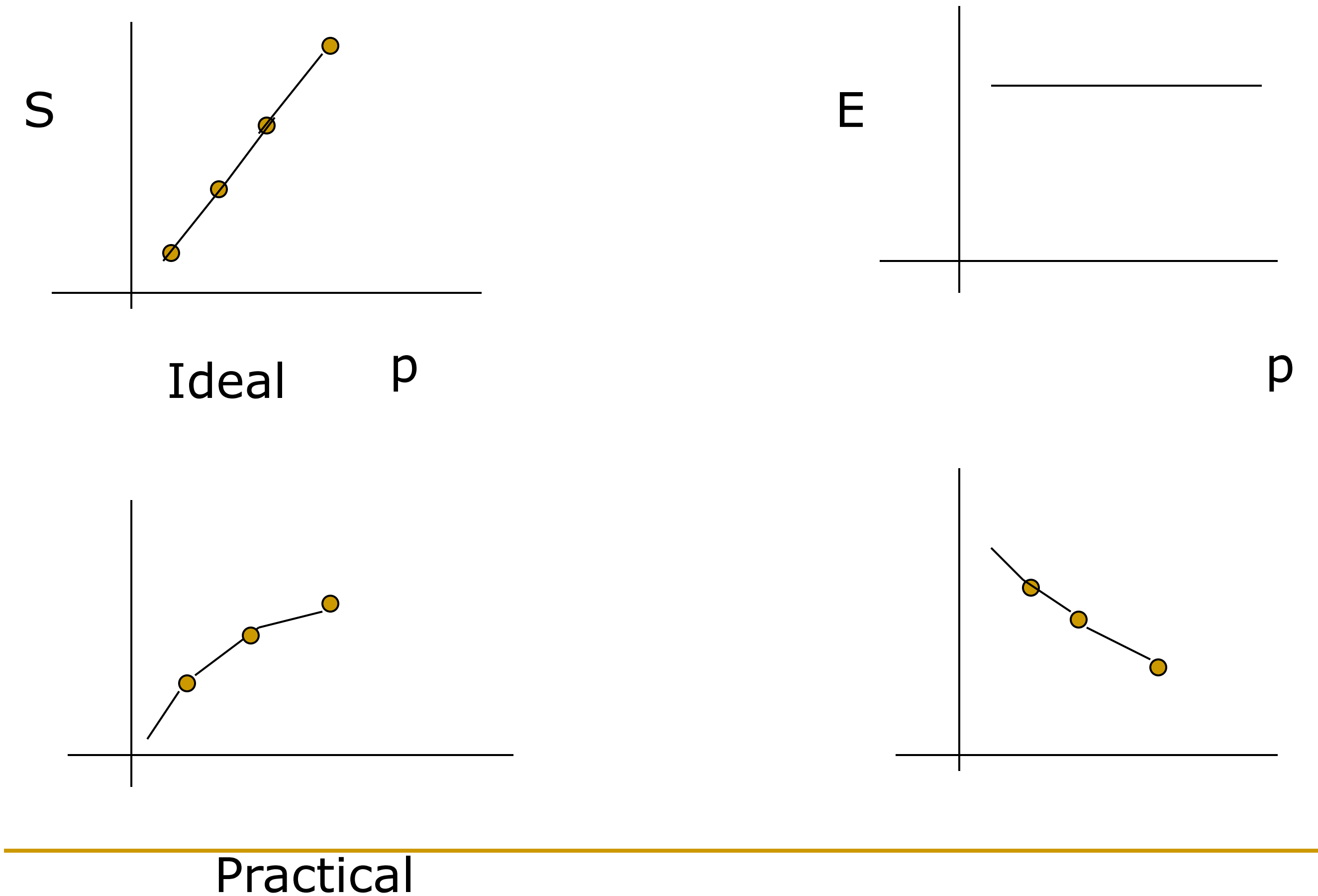


P0
P1
Idle time
Computation
Communication
Synchronization

How do we evaluate a parallel program?

- Execution time, T_p
- Speedup, S
  - $S(p, n) = T(1, n) / T(p, n)$
  - Usually, $S(p, n) < p$
  - Sometimes $S(p, n) > p$ (superlinear speedup)
- Efficiency, E
  - $E(p, n) = S(p, n)/p$
  - Usually, $E(p, n) < 1$
  - Sometimes, greater than 1
- Scalability – Limitations in parallel computing, relation to *n* and *p*.

Speedups and efficiency



Limitations on speedup – Amdahl's law

- Amdahl's law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- Overall speedup in terms of fractions of computation time with and without enhancement, % increase in enhancement.
- Places a limit on the speedup due to parallelism.
- $\text{Speedup} = \dfrac{1}{(f_s + (f_p/P))}$

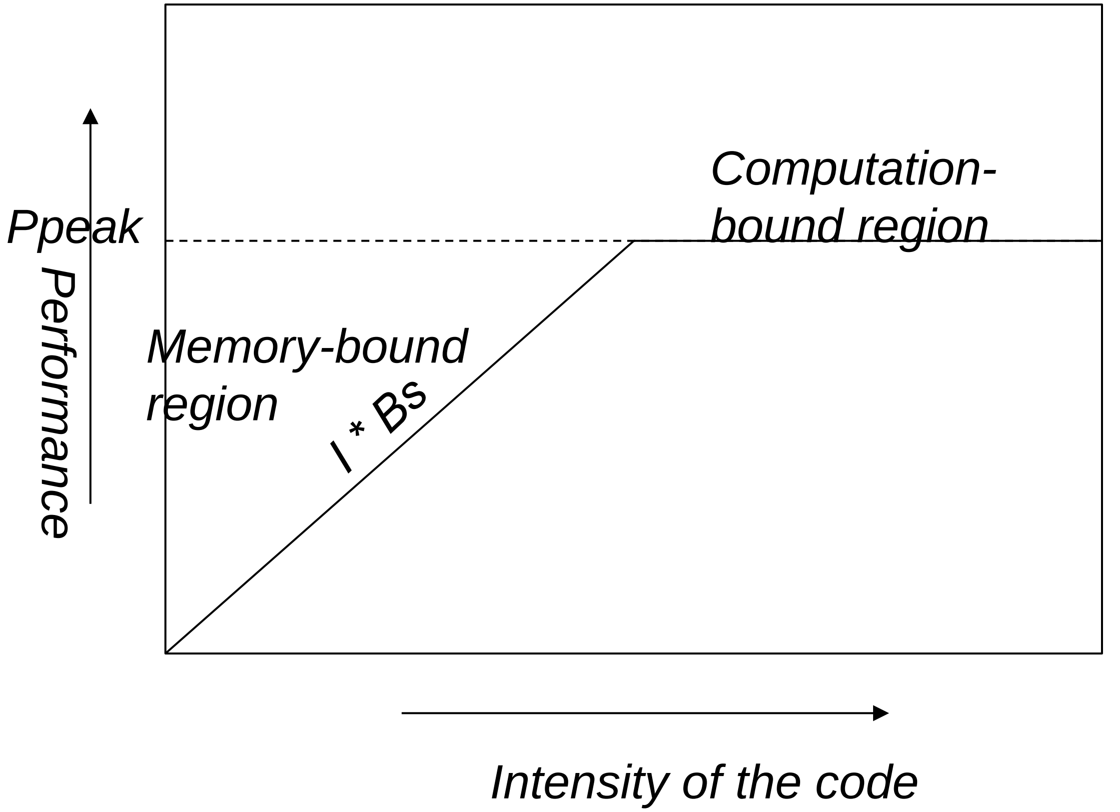
Gustafson's Law

- Increase problem size proportionally so as to keep the overall time constant
- The scaling keeping the problem size constant (Amdahl's law) is called **strong scaling**
- The scaling due to increasing problem size is called **weak scaling**

Roofline performance model

- Gives a bound on the performance of an application on a particular architecture
- Helps to categorize the code's performance as memory-bound or performance-bound
- Depends on three parameters
  - Peak performance of a machine, P_{peak} (FLOP/s)
  - Memory bandwidth of the architecture, B_s (Bytes/s)
  - Computation intensity of the code, I (FLOP/Byte)
- Time taken given by $I \times B_s$

Roofline performance model graph

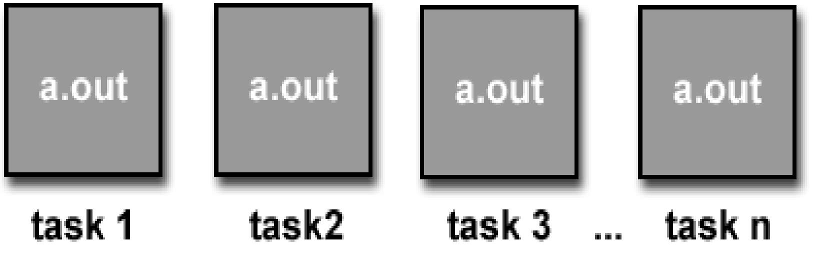


- Refer youtube video for excellent details: https://youtu.be/IrkNZG8MJ64?si=nMb8XnV02NLuxwQj

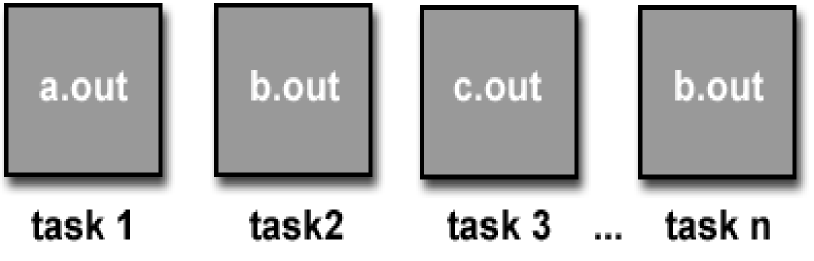
PARALLEL PROGRAMMING CLASSIFICATION AND STEPS

Parallel Program Models

- Single Program Multiple Data (SPMD)



- Multiple Program Multiple Data (MPMD)



Courtesy: http://www.llnl.gov/computing/tutorials/parallel_comp/

Programming Paradigms

- Shared memory model – Threads, OpenMP, CUDA
- Message passing model – MPI

Data Parallelism and Domain Decomposition

- Given data divided across the processing entitites
- Each process owns and computes a portion of the data – owner-computes rule
- Multi-dimensional domain in simulations divided into subdomains equal to processing entities
- This is called **domain decomposition**

Domain decomposition and Process Grids

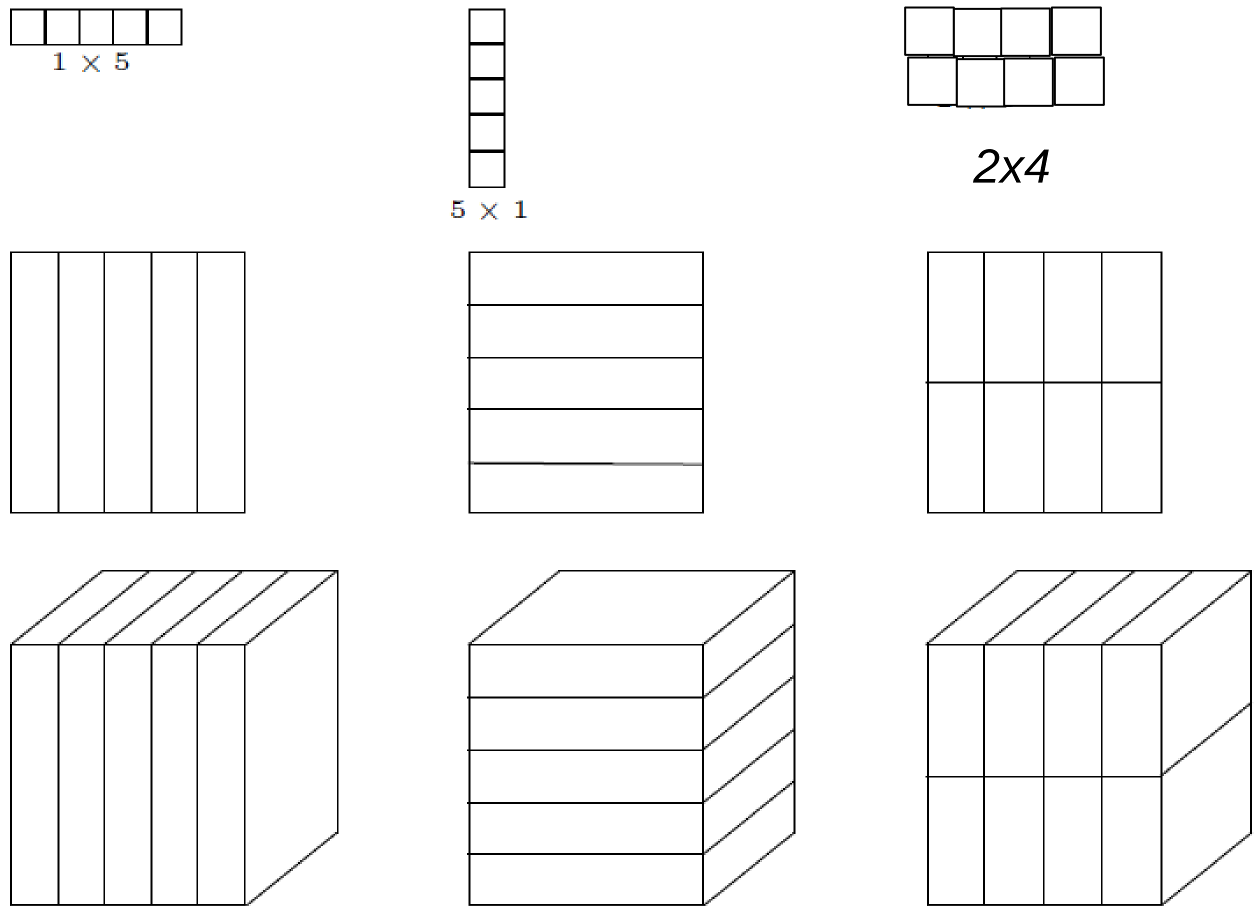
- Process grid used to specify domain decomposition
- The given P processes arranged in multi-dimensions forming a **process grid**
- The domain of the problem divided into process grid

Illustrations

Process grid

2-D domain decomposed using the process grid

3-D domain decomposed using the process grid



Data Distributions

- For dividing the data in a dimension using the processes in a dimension, **data distribution** schemes are followed
- Common data distributions:
  - Block: for regular computations
  - Block-cyclic: when there is load imbalance across space

b_2 ↔, b_1 ↕

0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
3	4	5	3	4	5	3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
3	4	5	3	4	5	3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
3	4	5	3	4	5	3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
3	4	5	3	4	5	3	4	5	3	4	5	3	4	5