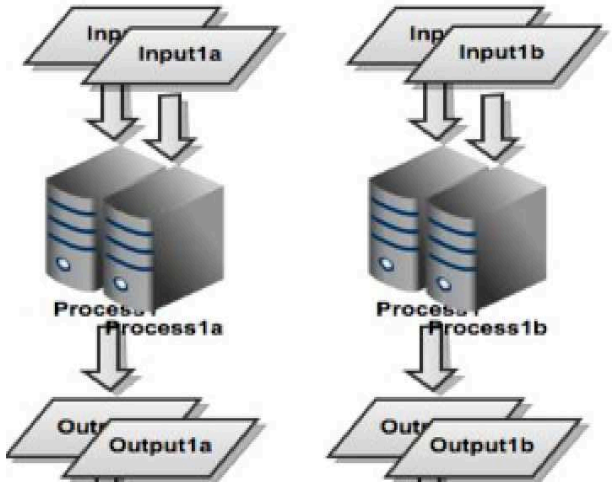
Introduction to Parallel and Distributed Computing

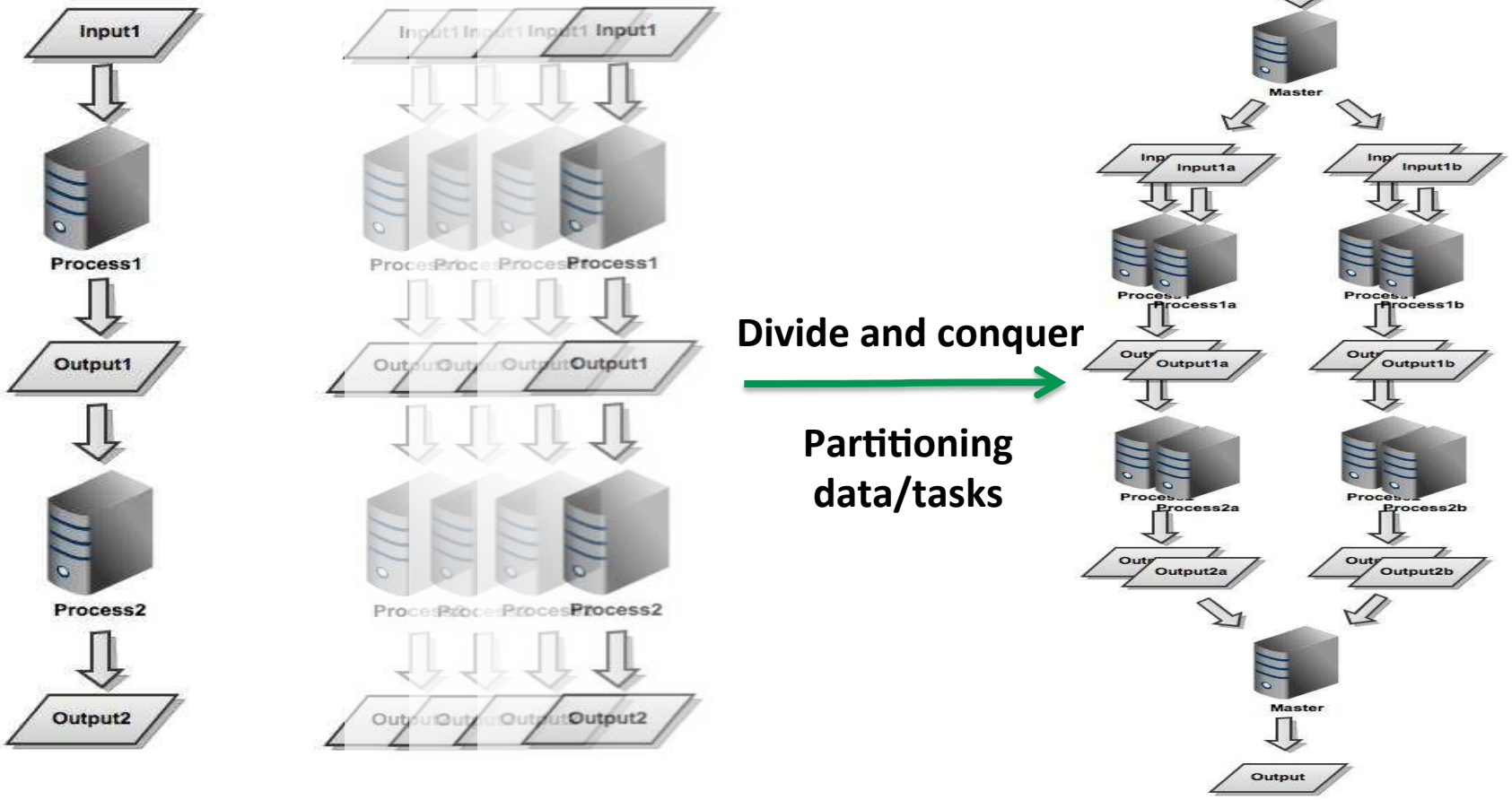
SURFsara – Online Grid Computing Course

Outline

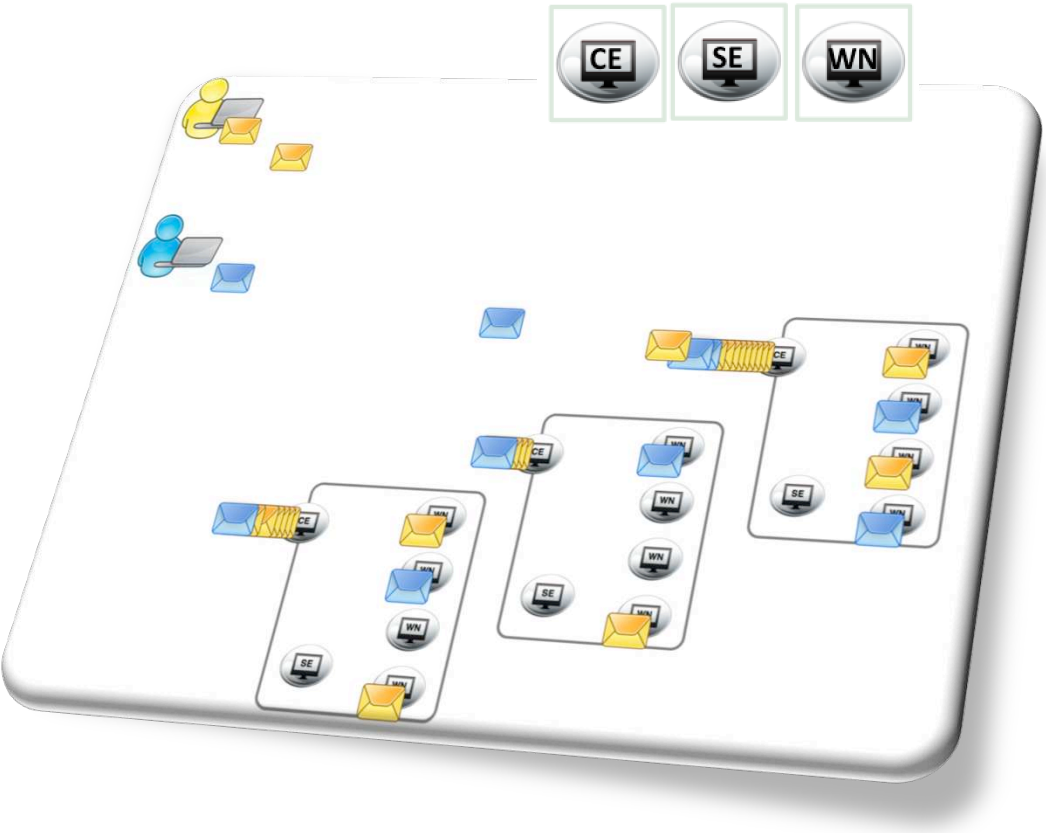
- **Think parallel**
- **Distributed vs. Parallel Computing**
  - Flynn's Taxonomy
  - Memory architectures
    - ✓ OpenMP & MPI
- **Designing Parallel applications**
  - Data parallelism
  - Functional parallelism
  - Amdahl's Law
  - CPU time vs. Wall clock



Think parallel



Distributed vs. Parallel systems

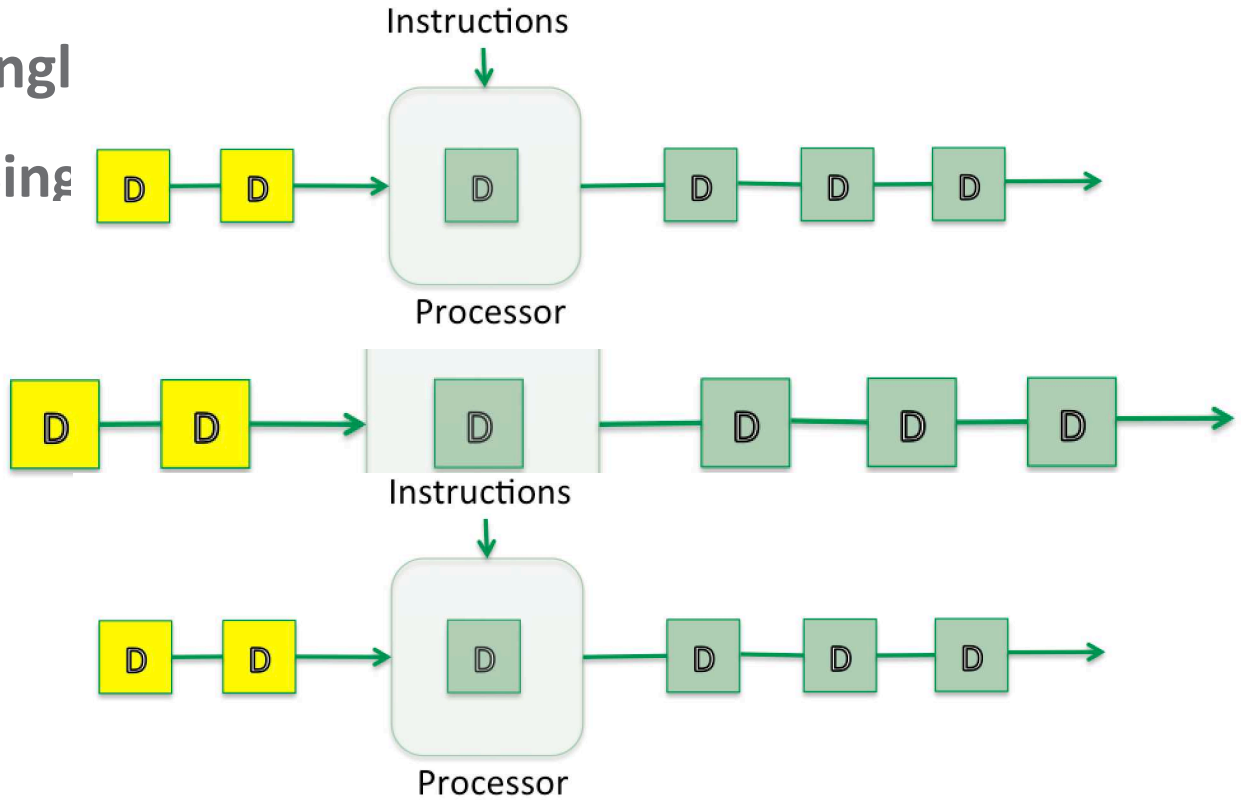


- **Distributed Computing**
  - Remote resources (CE, SE, WN)
  - Interconnected via network
  - RPC, CORBA, RMI
  - Milestones
    - ✓ Workstations (early 80's)
    - ✓ Clusters (90's)
    - ✓ Grid Computing (late 90's)
  - Loosely coupled
- **Parallel systems**
  - Shared memory
  - Fast data sharing
  - Tightly coupled

Flynn's taxonomy

Classify computers:

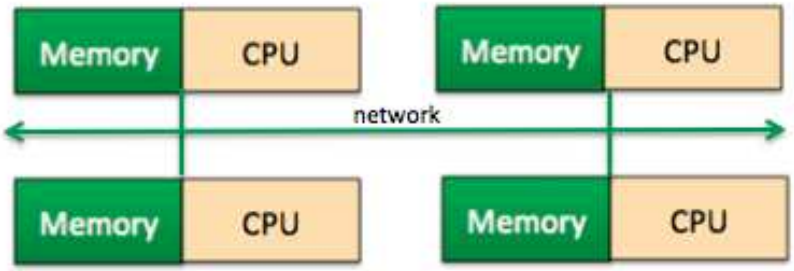
- SISD – Singl
- SIMD – Sing
- MISD – I
- MIMD –

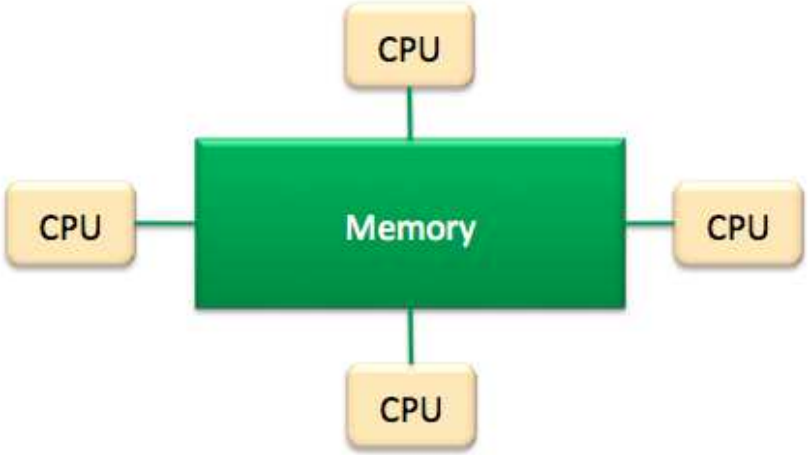


Memory architectures

Types of mapping:

- **Distributed memory systems**
  - Each CPU has its own memory
  - Message passing
  - eg. MPI:
    - ✓ Message Passing Interface
    - ✓ Communication overhead limits performance
- **Shared memory systems**
  - All CPUs access the same memory
  - Very fast
  - eg. OpenMP
    - ✓ Multicore programming
    - ✓ Fork/join threads

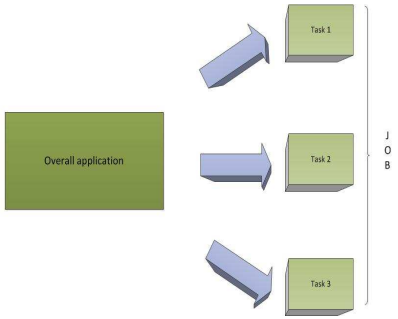


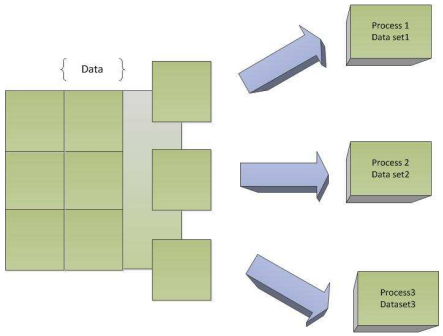


Designing parallel applications

- **Rule of thumb:**
  - place tasks that are able to execute concurrently on *different* processors: enhance concurrency
  - place tasks that communicate frequently on the *same* processor: increase locality
- **Steps to simplify code parallelization:**

1. ***Partitioning***: decompose the problem into smaller tasks.
2. ***Communication***: overhead is smaller for shared memory systems than for distributed systems
3. ***Agglomeration & Mapping***: make realistic decisions & map the tasks among processing units



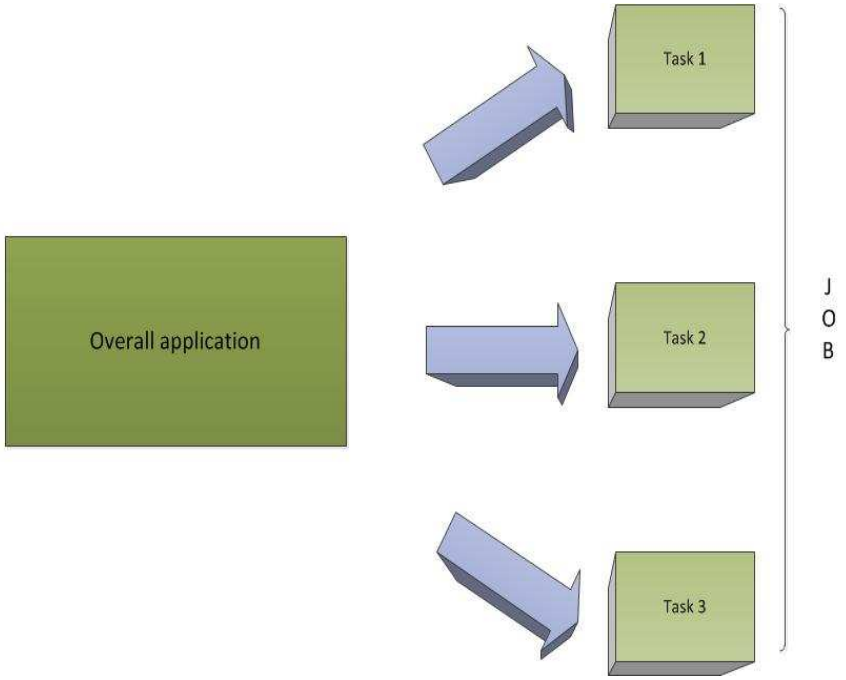


- **Granularity**
- **Optimization**
  - Check the time spent on message passing

Partitioning Tasks

Functional partitioning: a different task on the same (or different) data

Divide the application modules into small pieces (subtasks) and assign each to a separate processing element.



Useful for:

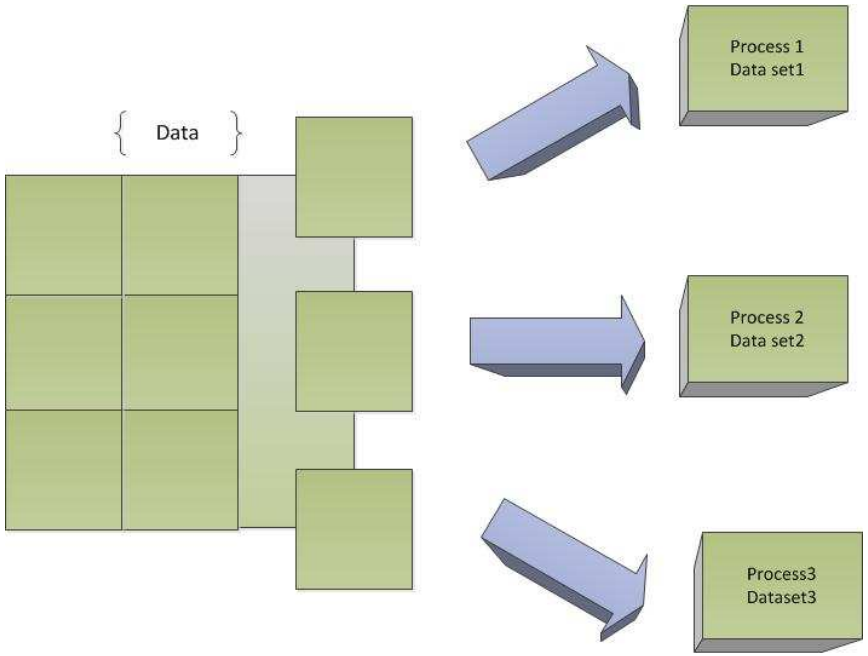
- reducing overall problem complexity

Drawbacks:

- dependencies between tasks, e.g. wait for intermediate results
- overlap of tasks may lead to shared data

Partitioning Data

- ***Data partitioning*** **: apply the same task on different data**
  - ***Load balancing:*** ensure that data blocks are roughly the same size to avoid threads waiting for a larger block of data to finish.



Useful for:

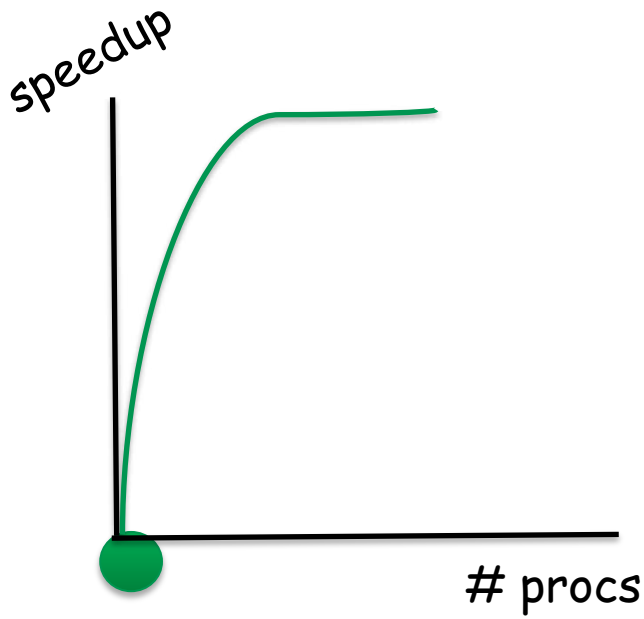
- large data sets
- independent data tasks

Drawbacks:

- The more tasks, the higher communication overhead!

Performance pick... Amdahl's law

- Parts of a program can be parallelized
- Other parts *must* execute sequentially
  - Amdahl's law:
    - ✓ T_p : parallel time
    - ✓ T_s : serial time
    - ✓ P : number of processors
    - ✓ $T_p = (1/S)\, T_s + (1 - 1/S)(\, T_s/P)$
    - ✓ Speedup = T_s / T_p



CPU vs. Wall clock time

- **CPU Hour (CH):**
  the time that the processor is actively working on a certain task.
- **Wall-clock time:**
  the real time taken by a computer to complete a job.

- **The less wall clock time:**
  - the higher the degree of parallelization
  - the more CPU time a program will use

For programs executed sequentially, the CPU time is slightly different to an hour of wall-clock time. For programs executed in parallel, the CPU time will be the sum of all the CPUs taking part in the process.