SURFsara HPC Cloud Workshop

Design a Parallel Application

www.cloud.sara.nl → Tutorial-2014-06-11

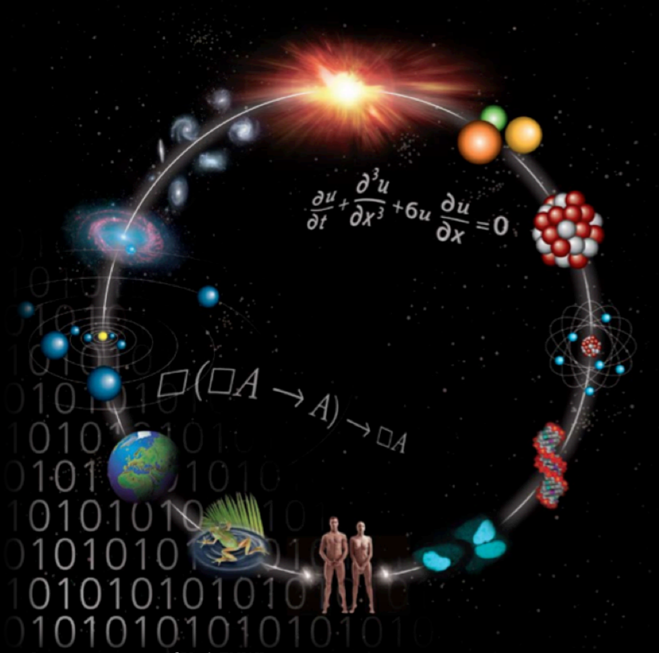
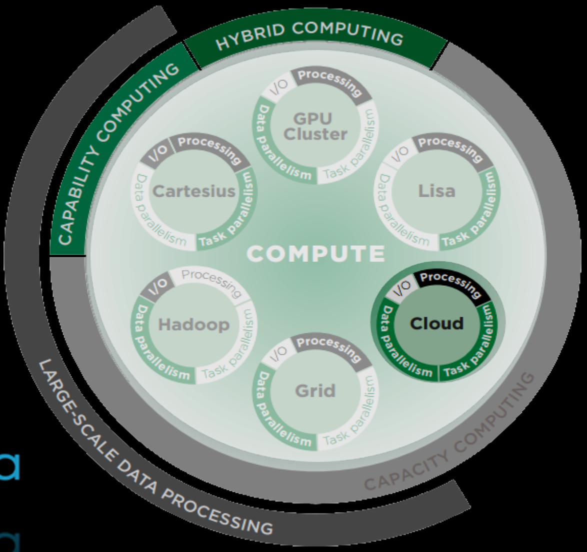

$\frac{\partial u}{\partial t}+\frac{\partial^3 u}{\partial x^3}+6u\frac{\partial u}{\partial x}=0$
$\Box(\Box A \to A) \to \Box A$
0101010101010

Image: courtesy of UvA FNWI


HYBRID COMPUTING

CAPABILITY COMPUTING

CAPACITY COMPUTING

LARGE-SCALE DATA PROCESSING

COMPUTE

GPU Cluster

Cartesius

Lisa

Hadoop

Cloud

Grid

VO

Processing

Data parallelism

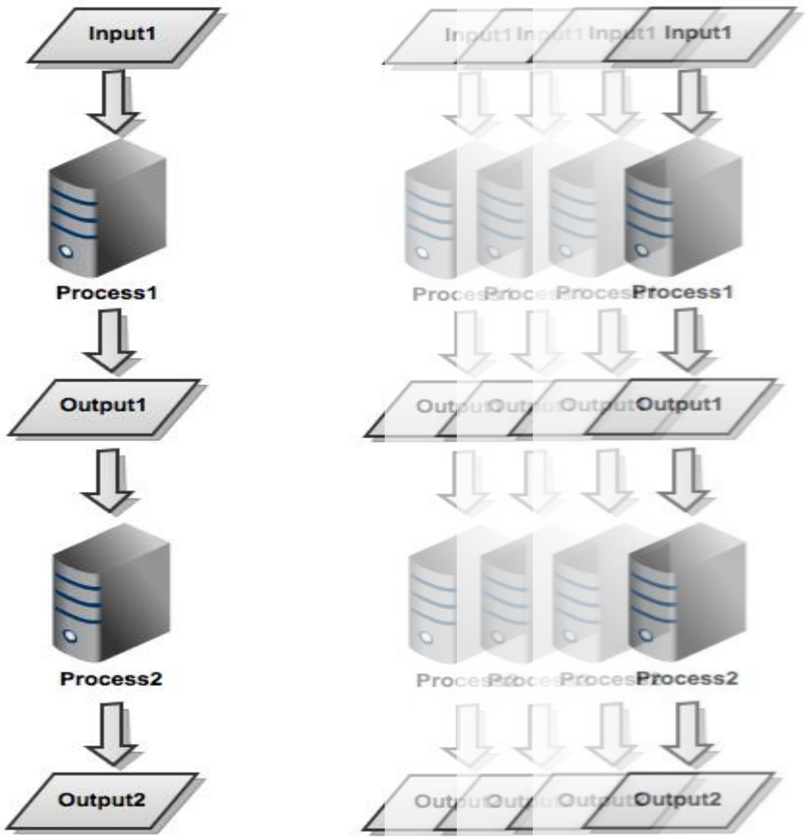
Task parallelism

UvA HPC and Big Data Course June 2014
Anatoli Danezi, Markus van Dijk
cloud-support@surfsara.nl

Agenda

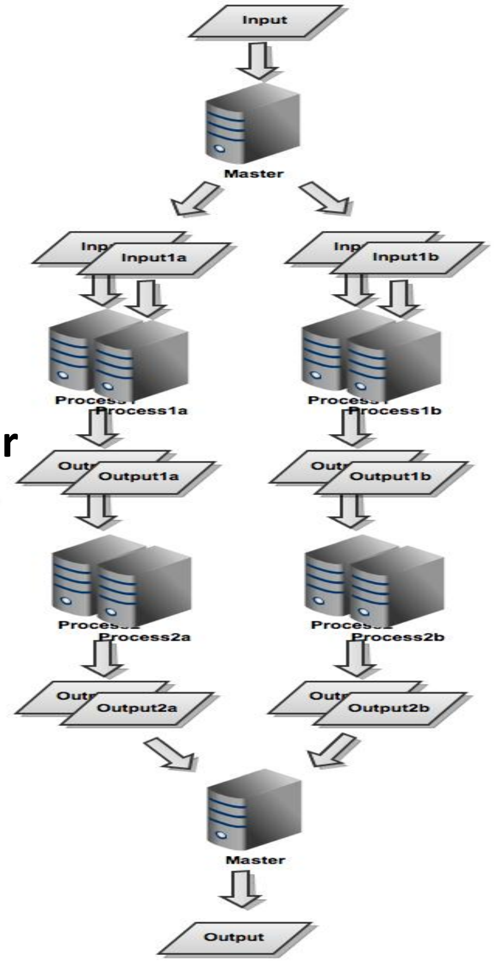
- ❑ **Think parallel**
- ❑ **Distributed vs. Parallel Computing**
  - Memory architectures
    - ✓ OpenMP & MPI
- ❑ **Designing Parallel applications**
  - Data & Functional parallelism
  - CPU time vs. Wall clock
  - Amdahl's Law
- ❑ **Hands-On: Tutorial 2014-06-11 MonkeySheet-Extras**
  - Calculation of an estimate of pi:
    - ✓ MPI
    - ✓ OpenMP (assignment)

Think parallel



Divide and conquer

Partitioning data/tasks

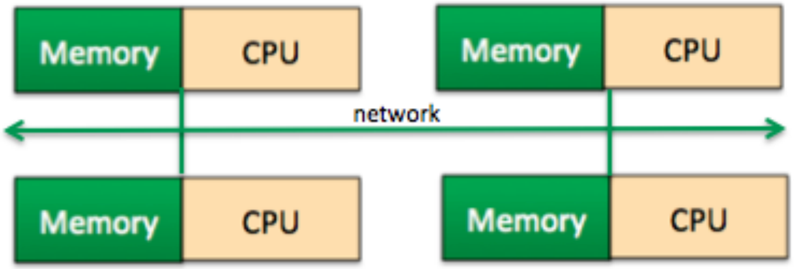


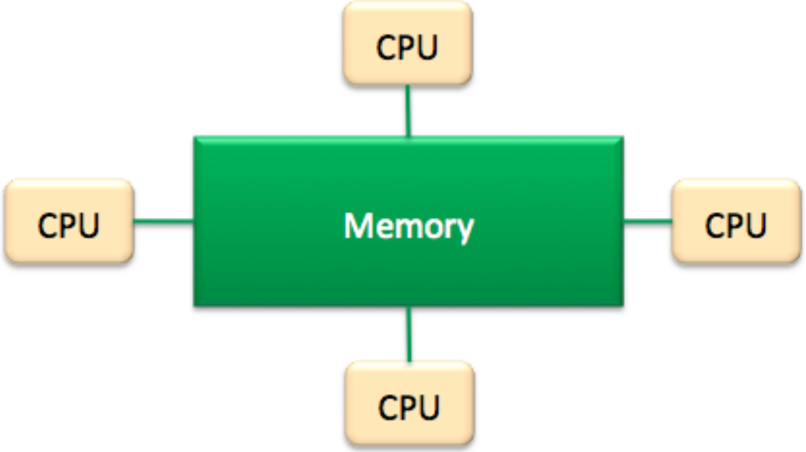
Distributed vs. Parallel systems

- ❑ **Distributed Computing**
  - Remote resources across multiple computers
  - Interconnected via network
  - Loosely coupled
- ❑ **Parallel systems**
  - Shared memory across multiple CPUs in a single computer
  - Fast data sharing
  - Tightly coupled

Memory architectures

- ❑ **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 (1/2)

- ❑ **Which Design model to choose?**
  - Multiple threads in one core
  - Multiple cores in a single machine
  - Multiple single core interconnected machines
  - Multiple multicore interconnected machines

- ❑ **Parallelization method:**
  - Partitioning Tasks
  - Partitioning Data

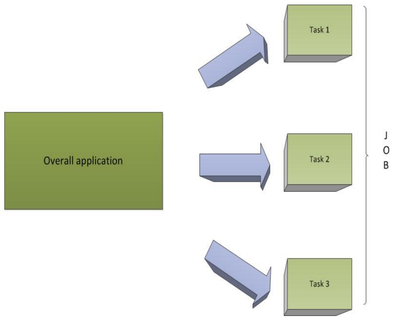
Designing parallel applications (2/2)

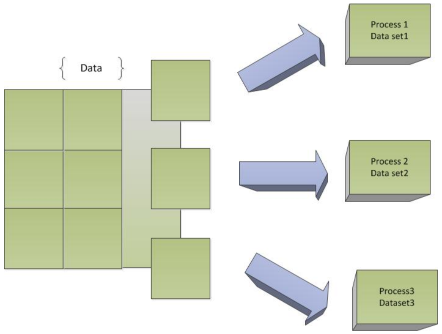
❑ **Rule of thumb:**

- 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**

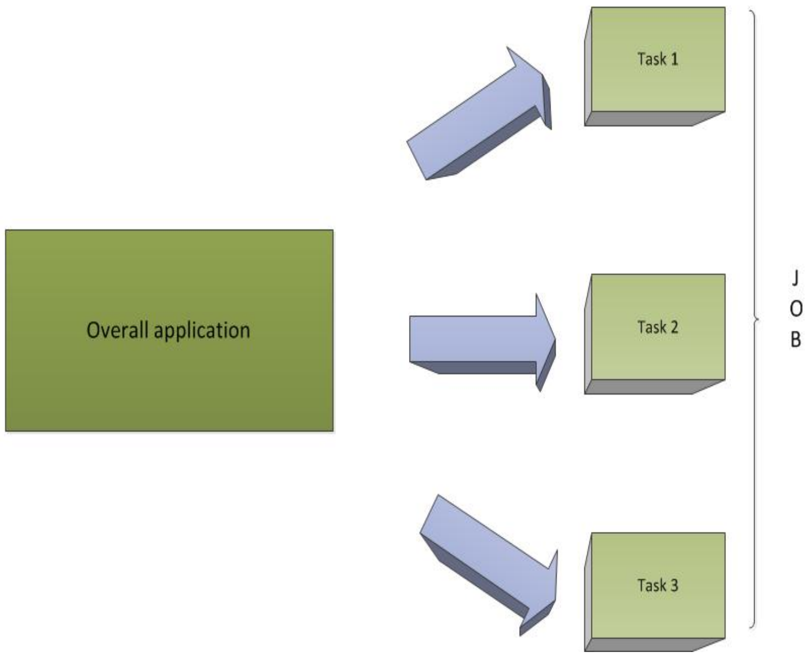
❑ **Optimization**

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:

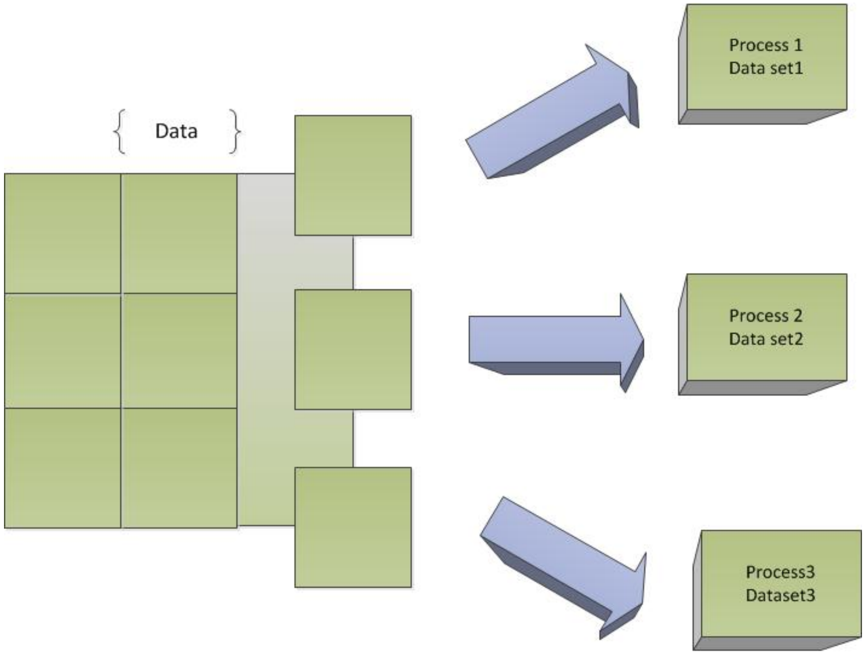
- reducing overall problem complexity

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:

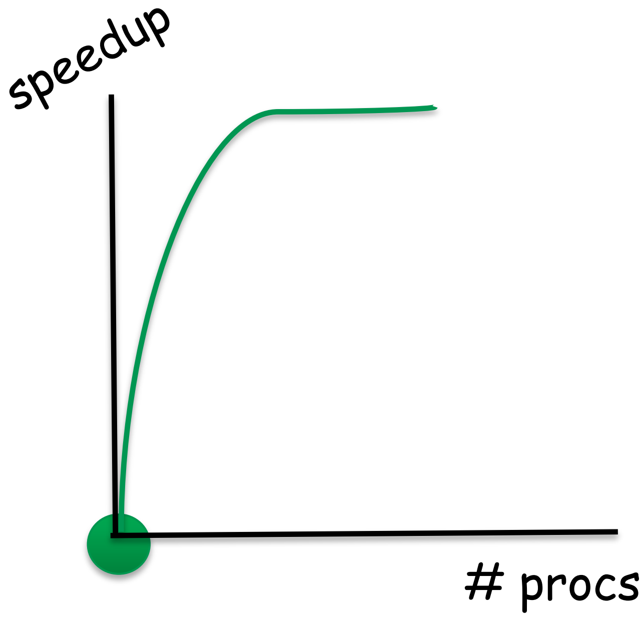
- large data sets
- independent data tasks

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:



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, CPU time is close to wall-clock time. For programs executed in parallel, CPU time is the sum of all the CPUs taking part in the process.

Hands on - Calculation of an estimate of pi

- ❑ **MPI (Tutorial 2014-06-11 MonkeySheet-Extras)**
  - Start a virtual cluster with 1 CPU VMs and a Linux distribution
  - Install Open MPI
  - Observe the performance on:
    - ✓ the master machine only
    - ✓ the virtual cluster (multiple VMs)
- ❑ **OpenMP (assignment)**
  - Start a VM with 2 CPUs and a Linux distribution
  - Scale up to more CPUs
  - Observe the performance for:
    - ✓ serial calculation
    - ✓ OpenMP optimized implementations

Hands On

Support portal: https://www.cloud.sara.nl

Search: *Tutorial 2014-06-11 MonkeySheet-Extras*

Self-service portal: https://ui.cloud.sara.nl

UvA HPC and Big Data Course June 2014
Anatoli Danezi, Markus van Dijk
cloud-support@surfsara.nl