# MapReduce
## A programming model
## for processing large datasets

### Prof. Tahar Kechadi

School of Computer Science & Informatics

# Outline

- **MapReduce**, the model

- **Hadoop**, an open-source implementation

- **Pig**, a high-level abstraction layer

# What is MapReduce?

- A data-parallel programming model designed for high scalability and resiliency

- Pioneered by Google. Also designed Percolator – *incrementally processing updates*
  - *Implementation in C++*

- Popularised by the open-source Hadoop project
  - Used at Yahoo!, Facebook, Amazon, …

# Used / has been used for

- At Google
  - Index construction for Google Search, Article clustering for Google News, Statistical machine translation
- At Yahoo!
  - "Web map" powering Yahoo! Search, Spam detection for Yahoo! Mail
- At Facebook
  - Ad optimisation, Spam detection
- In research
  - Astronomical image analysis, Bioinformatics, Analysing Wikipedia conflicts, Natural language processing, Particle physics, Ocean climate simulation
- Among others…

# Parallel Programming

- Process of developing programs that express what computations should be executed in parallel.
- Parallel computing
  - use of two or more processors (computers), ***usually within a single system***, working simultaneously to solve a single problem
- Distributed computing
  - any computing that involves ***multiple computers remote from each other*** that each have a role in a computation problem or information processing

# Parallel vs. Distributed Computing

| Characteristic | Parallel | Distributed |
|---|---|---|
| Overall Goal | Speed | Convenience |
| Interactions | Frequent | Infrequent |
| Granularity | Fine | Coarse |
| Reliable | Assumed | Not Assumed |

# Parallel Programming Methodical Design
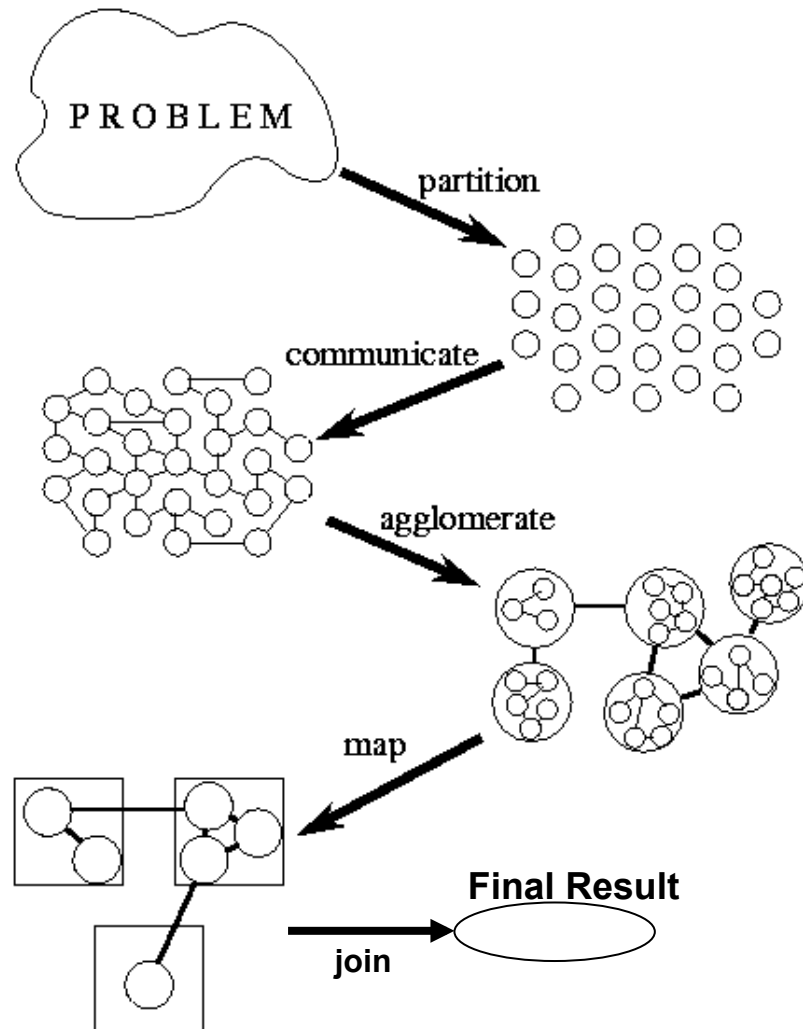


- **Partitioning**
  - Computation, data are decomposed into small tasks.
- **Communication**
  - Coordinate task execution
- **Agglomeration**
  - Tasks are combined to larger tasks
- **Map**
  - Assigned to a processor
- **Join**

# Parallel Programming Paradigms

● **Data parallelism**

- All tasks apply the same set of operations to different data

- Example:      for i ← 0 to 99 do
                    a[i] ← b[i] + c[i]
                 endfor


- Operations may be performed either synchronously or asynchronously

- Fine grained parallelism

- *Grain Size* is the average number of computations performed between communication or synchronization steps

# Parallel Programming Paradigms

- Task parallelism
  - Independent tasks apply different operations to different data elements
  - Example

    $$a \leftarrow 2$$
    $$b \leftarrow 3$$
    $$m \leftarrow (a + b) / 2$$
    $$s \leftarrow (a^2 + b^2) / 2$$
    $$v \leftarrow s - m^2$$

  - Concurrent execution of tasks, not statements
  - Problem is divided into different tasks
  - Tasks are divided between the processors
  - Coarse grained parallelism

# Traditional HPC systems

- CPU-intensive computations
  - Relatively small amount of data
  - Tightly-coupled applications
  - Highly concurrent I/O requirements
  - Complex message passing paradigms such as MPI, PVM…
  - Developers might need to spend some time designing for failure