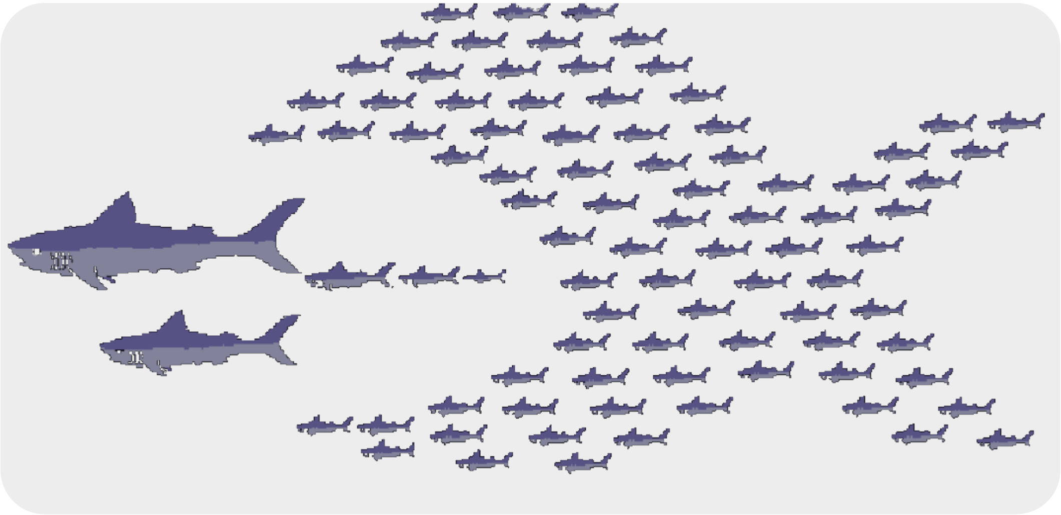
ICHEC
Irish Centre for High-End Computing

Parallel Programming Paradigms

Why Parallelisation?

Parallelism means doing multiple things at the same time

→ you can get more work done in the same time.



Serial vs parallel

- Serial (Sequential) Programming: Problem is broken into a discrete series of instructions. They are executed one after another.
- Parallel Programming: Problem is broken into discrete parts. Each part is broken into a series of instructions. Instructions are executed simultaneously on different CPUs.
- From Serial to Parallel Programming:
  - Can it be parallelized
  - Determine the hotspots and bottlenecks

Parallelisation Strategies

- Automatic Parallelisation
  - Compiler can analyze serial loops for potential parallel execution
- Using Parallel Libraries
  - A large collection of functions that can benefit applications and provide immediate performance.
- From-scratch Application Development
  - Compiler Directives
  - Message Passing

Types of Parallelism

- Data Parallelism: The same task run on different data in parallel

```
from i=0 to i=99
  c[i]=a[i]+b[i]
```

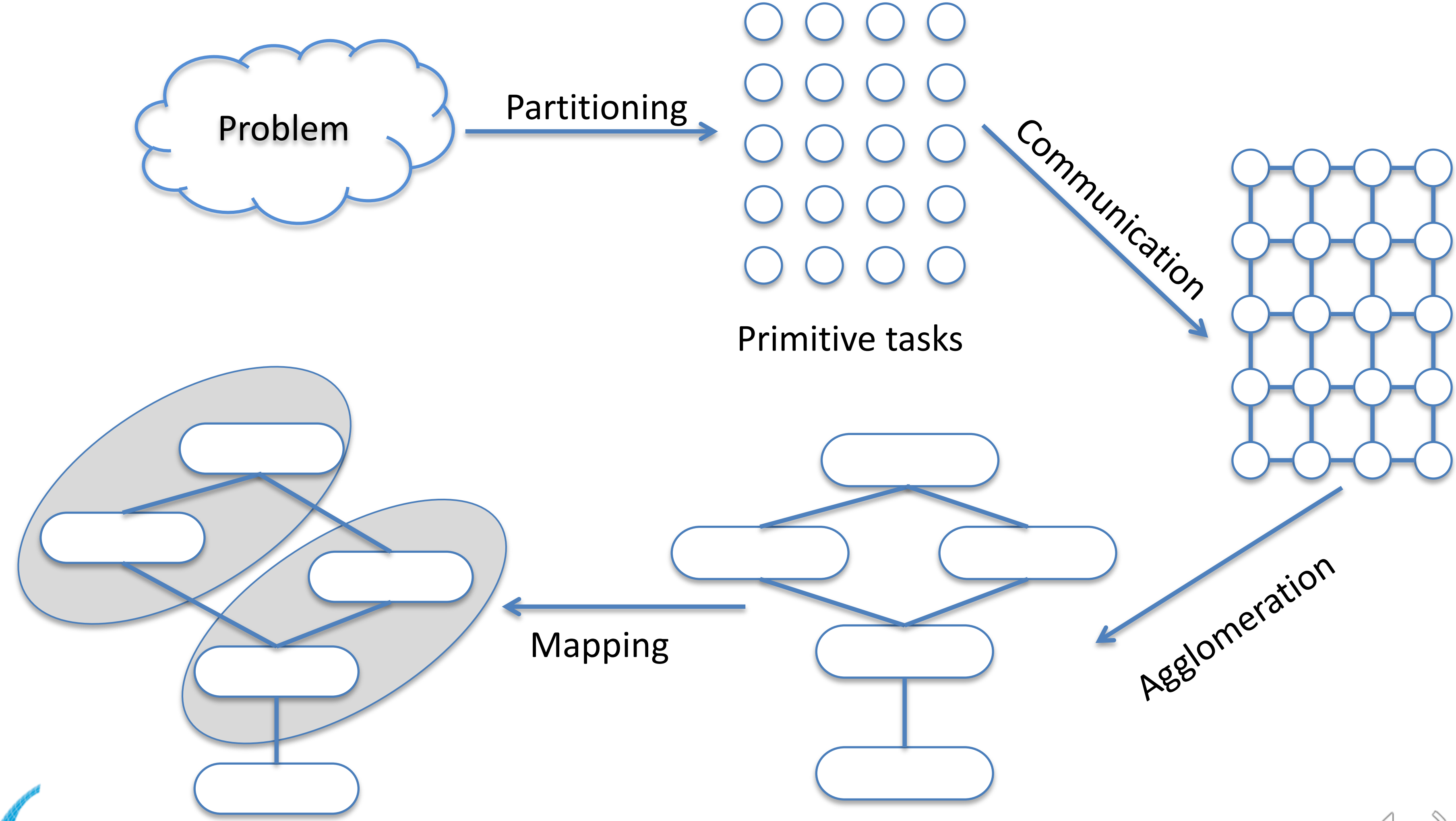
- Functional (Task) Parallelism: Different tasks running on the same data

```
Calculate a and b
  x=(a+b)/2
  y=min(a, b)
```

- Pipelining: A set of data processing elements connected in series. The output of one is the input of the next

```
a[0]=temp
from i=1 to i=n
  a[i]=a[i-1]+temp
```

Foster's Design Methodology



Partitioning

- Dividing computation and data into pieces
- Domain decomposition
  - Divide data into pieces
  - Determine how to associate computations with the data
- Functional decomposition
  - Divide computation into pieces
  - Determine how to associate data with the computations

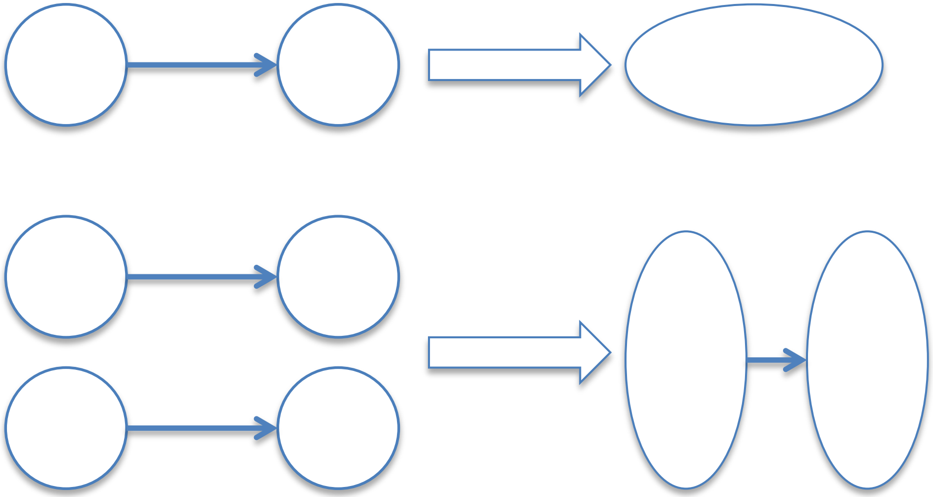
Communication

- Determine values passed among tasks
- Local communication
  - Task needs values from a small number of other tasks
  - Create channels illustrating data flow
- Global communication
  - Significant number of tasks contribute data to perform a computation
  - Don’t create channels for them early in design
- Minimize communication overhead, lower latency, increased bandwidth.

Agglomeration

- Grouping tasks into larger tasks
- Goals
  - Improve performance
  - Maintain scalability

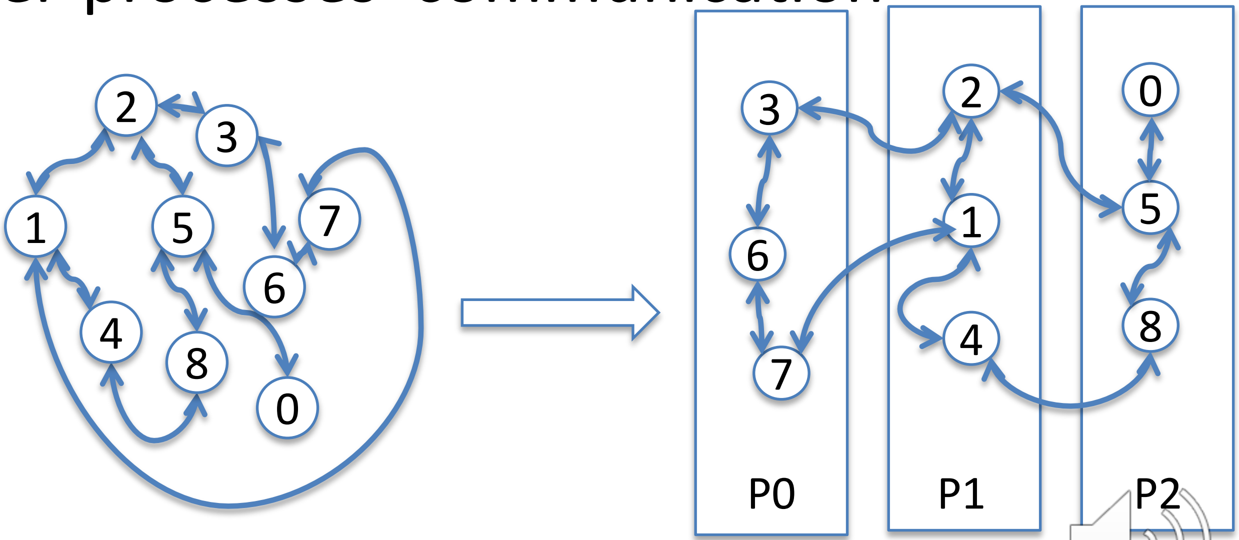
  of program
  - Simplify programming



Eliminate communication between primitive tasks agglomerated into consolidated task. Combine groups of sending and receiving tasks.

Mapping

- Process of assigning tasks to processes
- Goals of mapping
  - Maximize processor utilization
  - Minimize inter-processes communication


2
3
1
5
7
6
4
8
0
3
6
7
2
1
4
0
5
8
P0
P1
P2