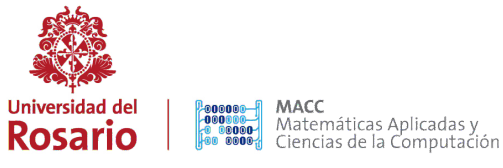
Message Passing

Carlos E. Alvarez[1].

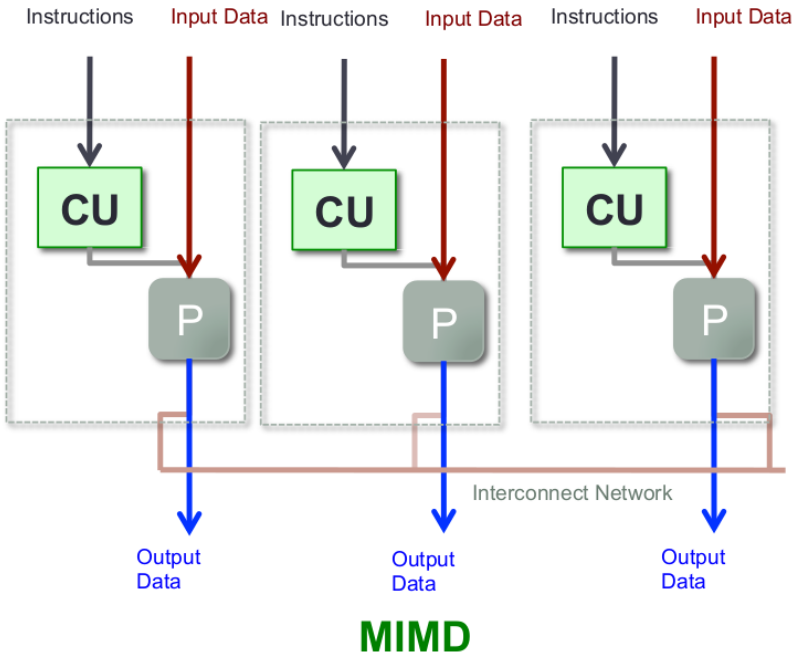
[1]Dep. de Matemáticas aplicadas y Ciencias de la Computación, Universidad del Rosario

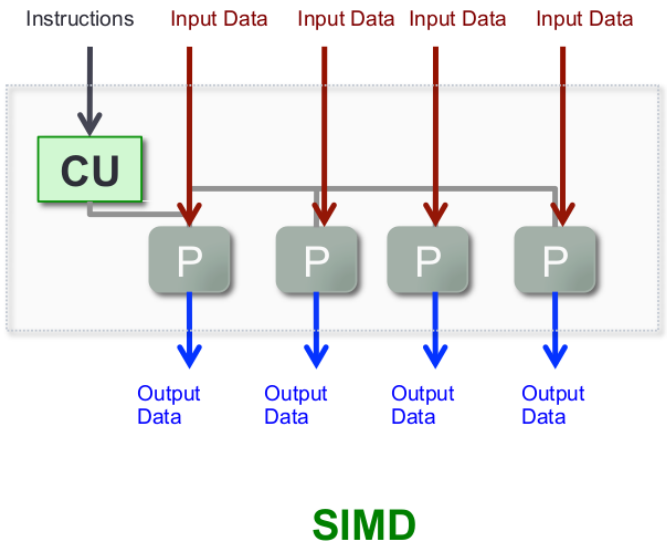
June 2019



Computational models

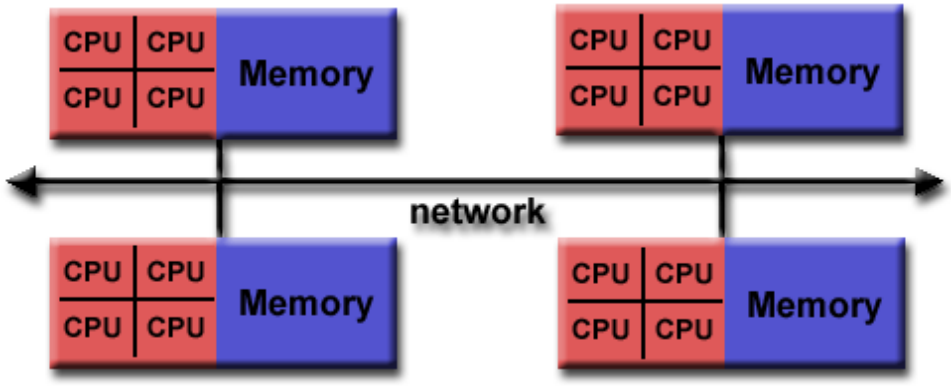
- Multiple Instructions Multiple Data
- Single Instruction Multiple Data





Communication models

- Shared memory
- Message passing



Several nodes connected by a network.

Process vs. thread

- **Process:** Provides the resources to execute a program (virtual address space, executable code, environment variables, etc.). Has a unique process identifier. Starts with a single thread
- **Thread:** Entity within a process that can be scheduled for execution. Share virtual address space and system resources with other threads of the same process. Has a unique thread identifier within the process

Message passing model

Single Instruction Multiple Data

Message passing model

Single Instruction Multiple Data

- Process: Instance of a program with its data

Message passing model

Single Instruction Multiple Data

- Process: Instance of a program with its data
- Many processes, one task

Message passing model

Single Instruction Multiple Data

- Process: Instance of a program with its data
- Many processes, one task
- Each process accesses its own data

Message passing model

Single Instruction Multiple Data

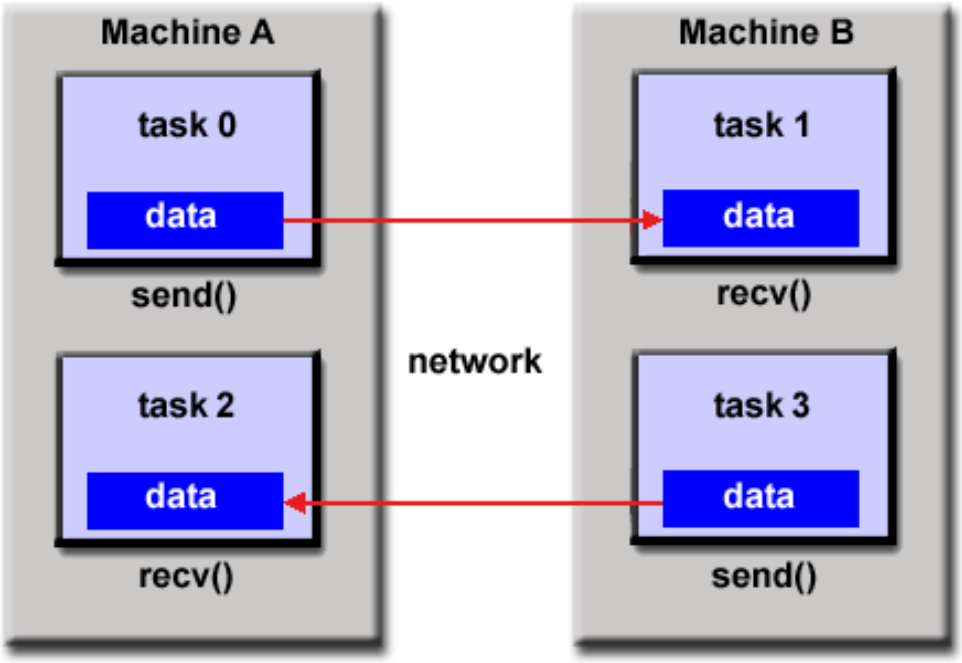
- Process: Instance of a program with its data
- Many processes, one task
- Each process accesses its own data
- Processes communicate sending and receiving messages

Message passing model

Single Instruction Multiple Data

- Process: Instance of a program with its data
- Many processes, one task
- Each process accesses its own data
- Processes communicate sending and receiving messages
- Usually run a single process per core

Message passing model



Message passing.

Messages

Parts of a message:

Messages

Parts of a message:

- ID of sender (core)

Messages

Parts of a message:

- ID of sender (core)
- ID of receiver (core)

Messages

Parts of a message:

- ID of sender (core)
- ID of receiver (core)
- Type of the data

Messages

Parts of a message:

- ID of sender (core)
- ID of receiver (core)
- Type of the data
- Number of data items

Messages

Parts of a message:

- ID of sender (core)
- ID of receiver (core)
- Type of the data
- Number of data items
- The data

Messages

Parts of a message:

- ID of sender (core)
- ID of receiver (core)
- Type of the data
- Number of data items
- The data
- A message tag (identifier)

Point to point communication

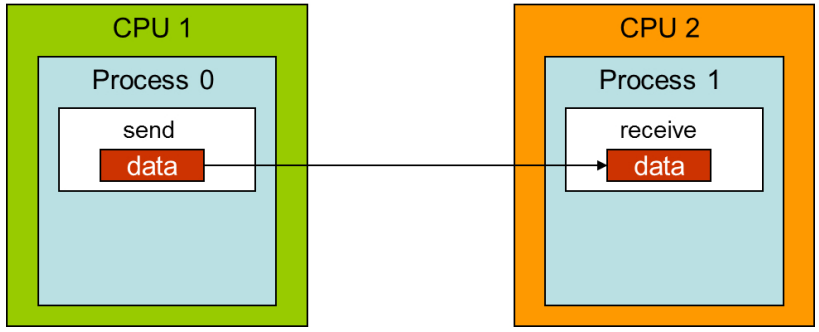
- Simplest form of communication

Point to point communication

- Simplest form of communication
- Relies on matching send and receive

Point to point communication

- Simplest form of communication
- Relies on matching send and receive
- Involves two processes
  - Sender
  - Receiver



Simple send-receive.

Collective communications

Communication between groups of processes.

MACC
Matemáticas Aplicadas y
Ciencias de la Computación

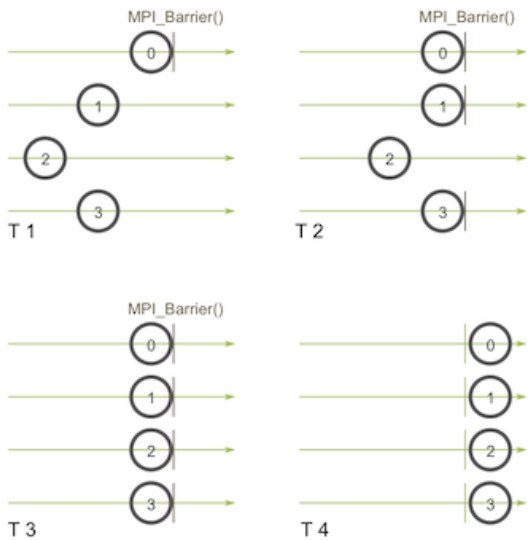
Collective communications

Communication between groups of processes.

- Barrier
- Broadcast
- Scatter
- Gather
- Reduction

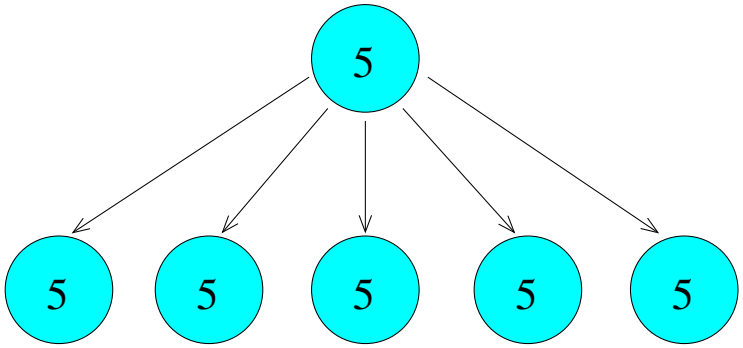
Barrier

- Execution continues once all processes get to the barrier
- Global synchronization



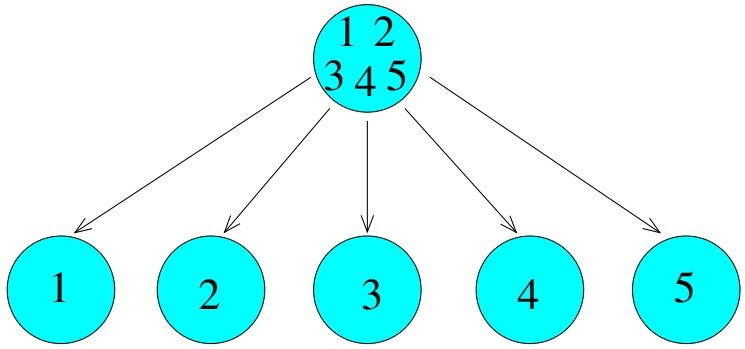
Broadcast

- One to all communication
- Copy of the same data



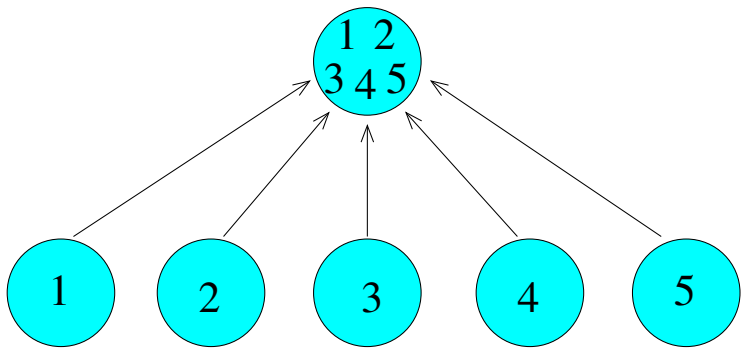
Scatter

- One to all communication
- The data is distributed in different pieces



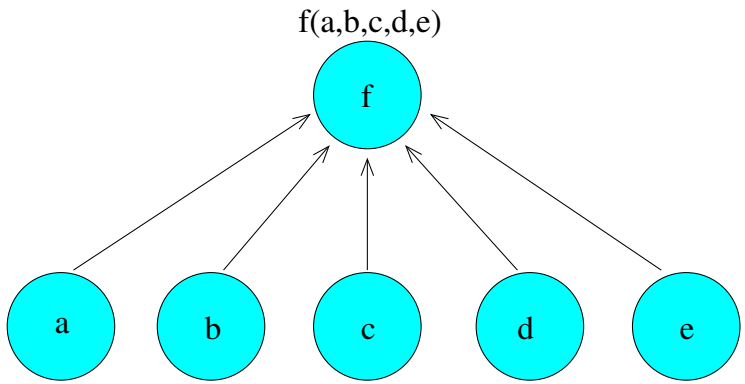
Gather

- All to one communication
- The data is assembled from the different pieces



Reduction

- All to one communication
- Compute a function of the incoming data
- Global sum, product, max, min, etc.



Group and context

- **Group:** Subset of processes that communicate with one another. Process ranks are interpreted relative to the group

Group and context

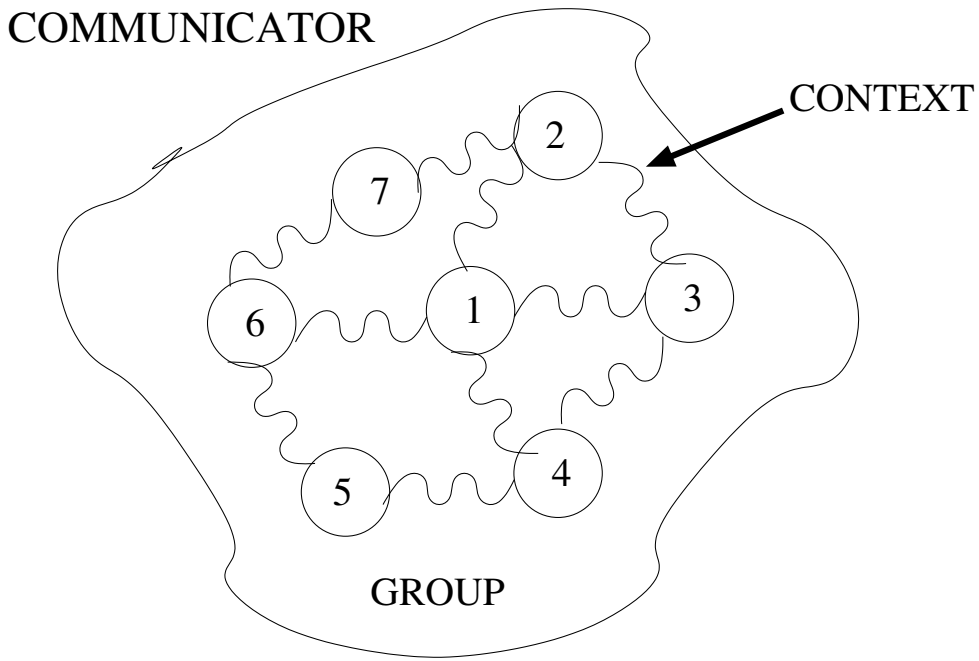
- **Group:** Subset of processes that communicate with one another. Process ranks are interpreted relative to the group
- **Context:** Partitions the communication space (analog to frequency in radio communications)

Group and context

- **Group:** Subset of processes that communicate with one another. Process ranks are interpreted relative to the group
- **Context:** Partitions the communication space (analog to frequency in radio communications)
- **Communicator:** Object that envelops a group and a context and specifies the scope of a communication operation

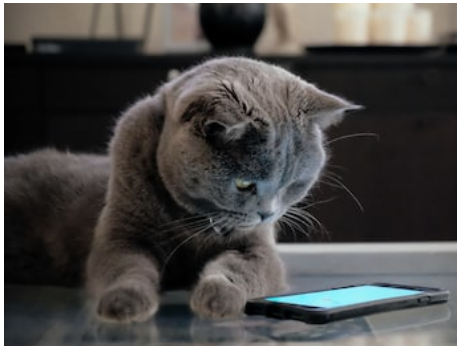
Communicator

Is the central object for communication in MPI.



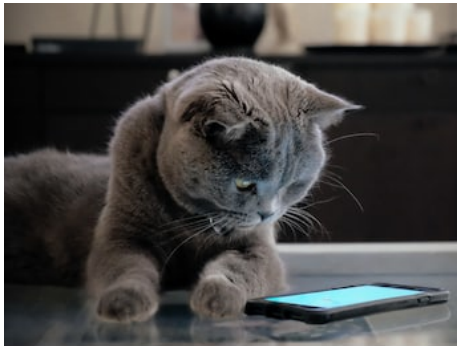
Synchronous vs. asynchronous communication

Synchronous (blocking):
The process waits for the message to be received



Synchronous vs. asynchronous communication

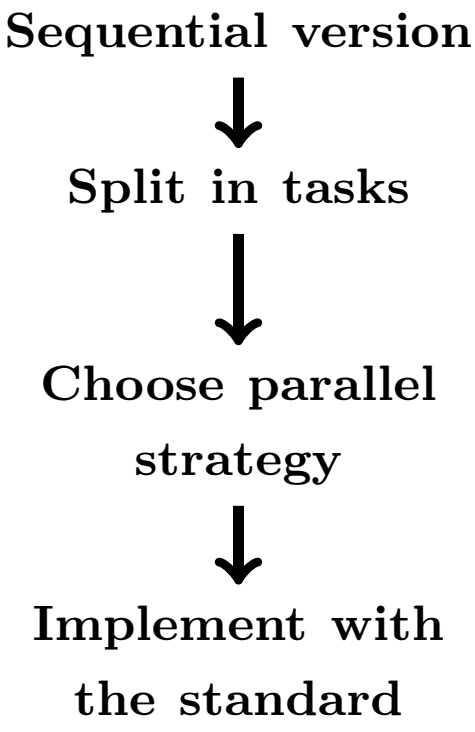
Synchronous (blocking):
The process waits for the message to be received



Asynchronous (non-blocking):
The process continues without waiting

Programming with message passing

One possibility:



Programming with message passing

Some parallel strategies:

Programming with message passing

Some parallel strategies:

- **Master/Slave:** One process has unidirectional control over one or more processes

Programming with message passing

Some parallel strategies:

- **Master/Slave:** One process has unidirectional control over one or more processes
- **Pipeline:** Chain of data processing units that can be executed in parallel (On process takes the output of another as input)

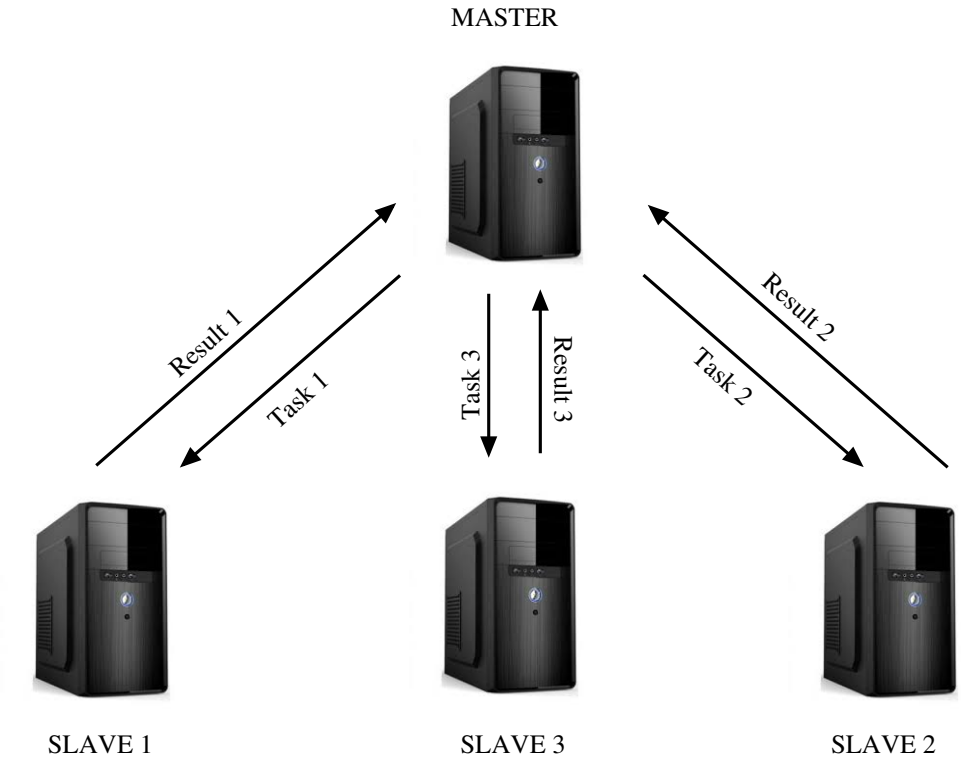
Programming with message passing

Some parallel strategies:

- **Master/Slave:** One process has unidirectional control over one or more processes
- **Pipeline:** Chain of data processing units that can be executed in parallel (On process takes the output of another as input)
- **Divide and Conquer:** A process takes a part of a problem and might spawn another process to further partition the problem, creating a tree-like structure

Programming with message passing

Master/Slave:



Programming with message passing

Pipeline:

Step 1	Step 2	Step 3	Step 4
Step 1	Step 2	Step 3	
Step 1	Step 2		
Step 1			

Programming with message passing

Divide and Conquer: