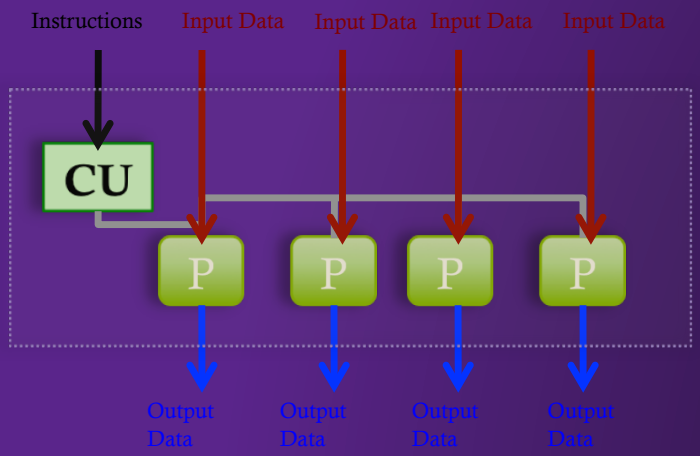
Introducing MPI to Distributed Memory Programming

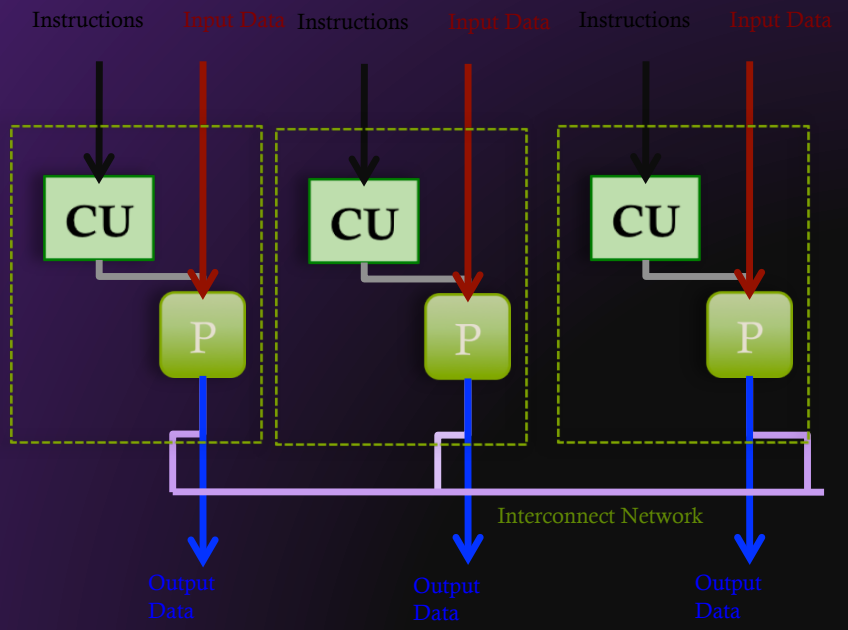
Carlos Jaime Barrios Hernández, PhD.

Remember Special Features of Architecture

- Remember "concurrency": it exploits better the resources (shared) within a computer.
- Exploit SIMD and MIMD Architectures

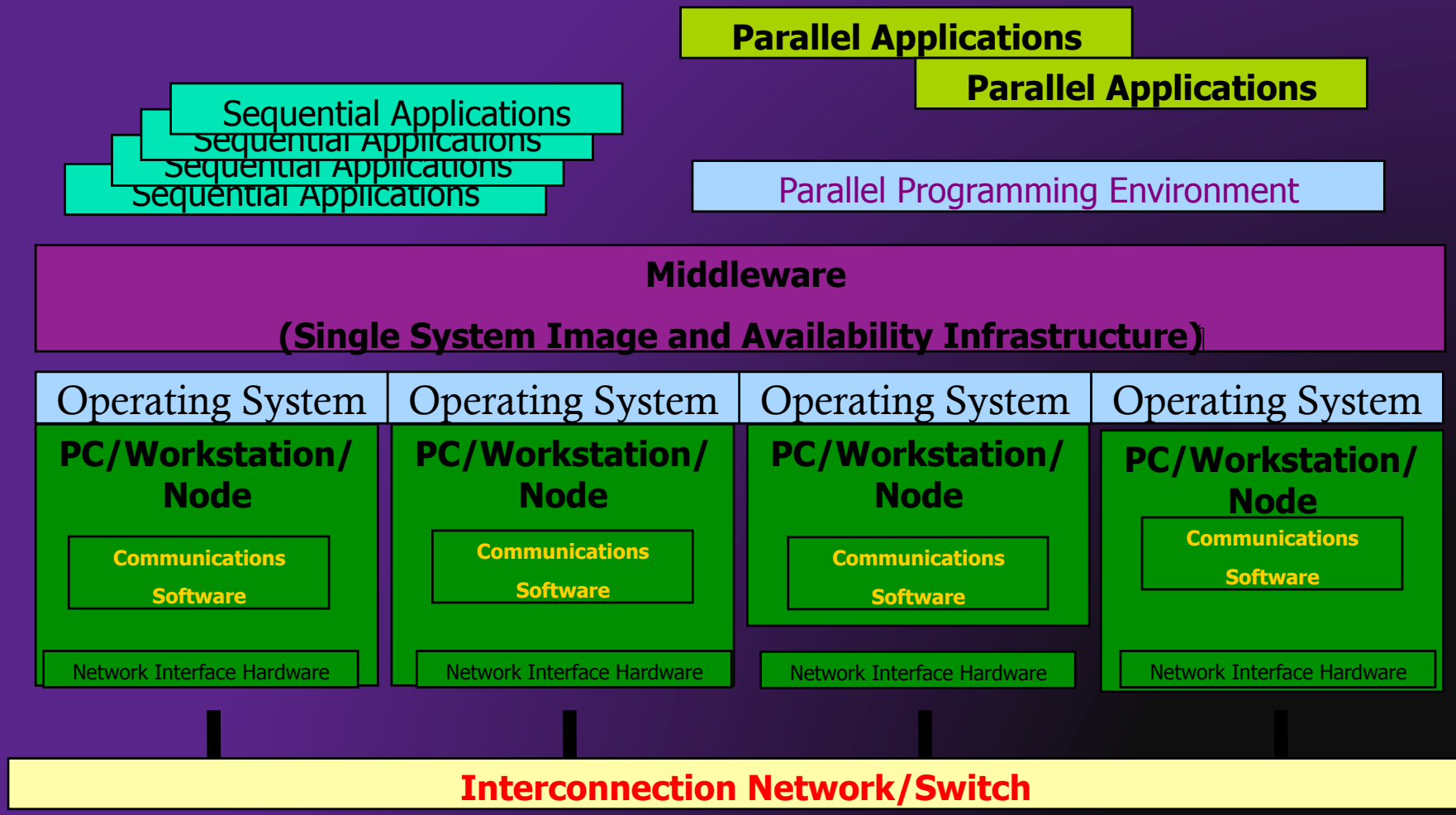


SIMD



MIMD

Cluster Computing Architecture



MPI – Message Passing Interface

- **MPI in a nutshell**
  - It is a library specification
  - Works natively with C and Fortran
  - Not a specific implementation or product
  - Scalable
    - Must handle multiple machines
  - Portable
    - Sockets API change from one OS to another
    - Handles Big-endian/little-endian architectures
  - Efficient
    - Optimized communication algorithms
    - Allow communication and computation overlap

MPI – Message Passing Interface

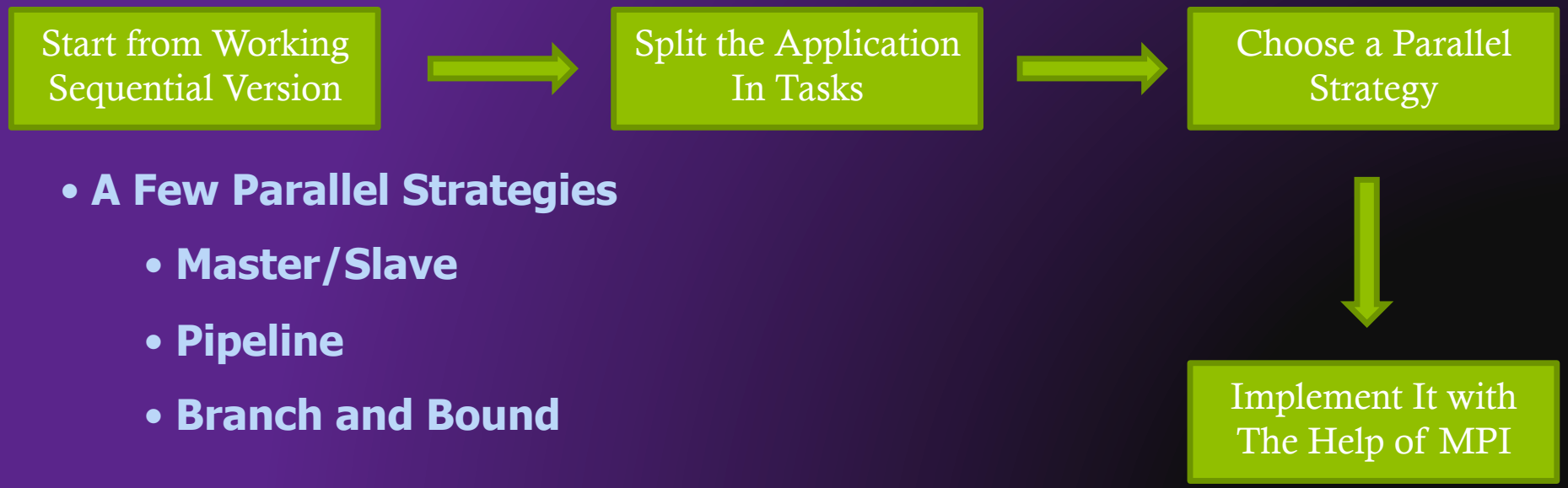
- **MPI References**
  - Books
    - Using MPI: Portable Parallel Programming with the Message Passing Interface, by Gropp, Lusk, and Skejellum, MIT Press, 1994.
    - MPI: The Complete Reference, by Snir, Otto, Huss-Lederman, Walker, and Dongarra, MIT Press, 1996.
    - Parallel Programming with MPI, by Peter Pacheco, Morgan Kaufmann, 1997.
  - The standard:
    - at http://www.mpi-forum.org

MPI Programming

- **MPI**
  - Use of a single program, on multiple data
  - What does it do?
    - way of identifying process
    - Independent of low-level API
    - Optimized communication
    - Allow communication and computation overlap
  - What does it do not?
    - gain performance of application for free
    - application must be adapted

MPI Programming

- **Possible Programming Workflow**



- **A Few Parallel Strategies**
  - **Master/Slave**
  - **Pipeline**
  - **Branch and Bound**

Parallel Strategies

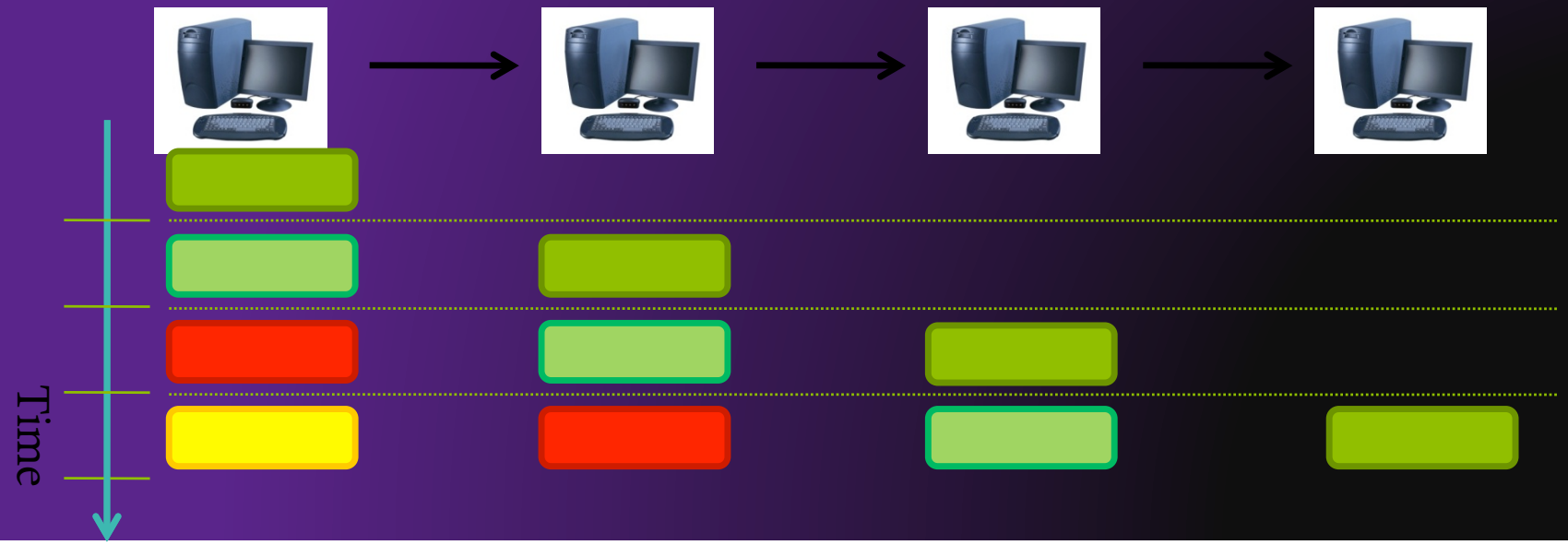
- Master/Slave
  - Master is one process that centrilizes all tasks
  - Slaves starve for work

Parallel Strategies

- Master/Slave
  - Master is often the bottleneck
  - Scalability is limited due to centralization
  - Possible to use replication to improve performance
  - It is adatable to heterogenous platforms

Parallel Strategies

- Pipeline
  - Each process plays a specific role, pipeline stages
  - Data follows in a single direction
  - Parallelism is achieved when the pipeline is full

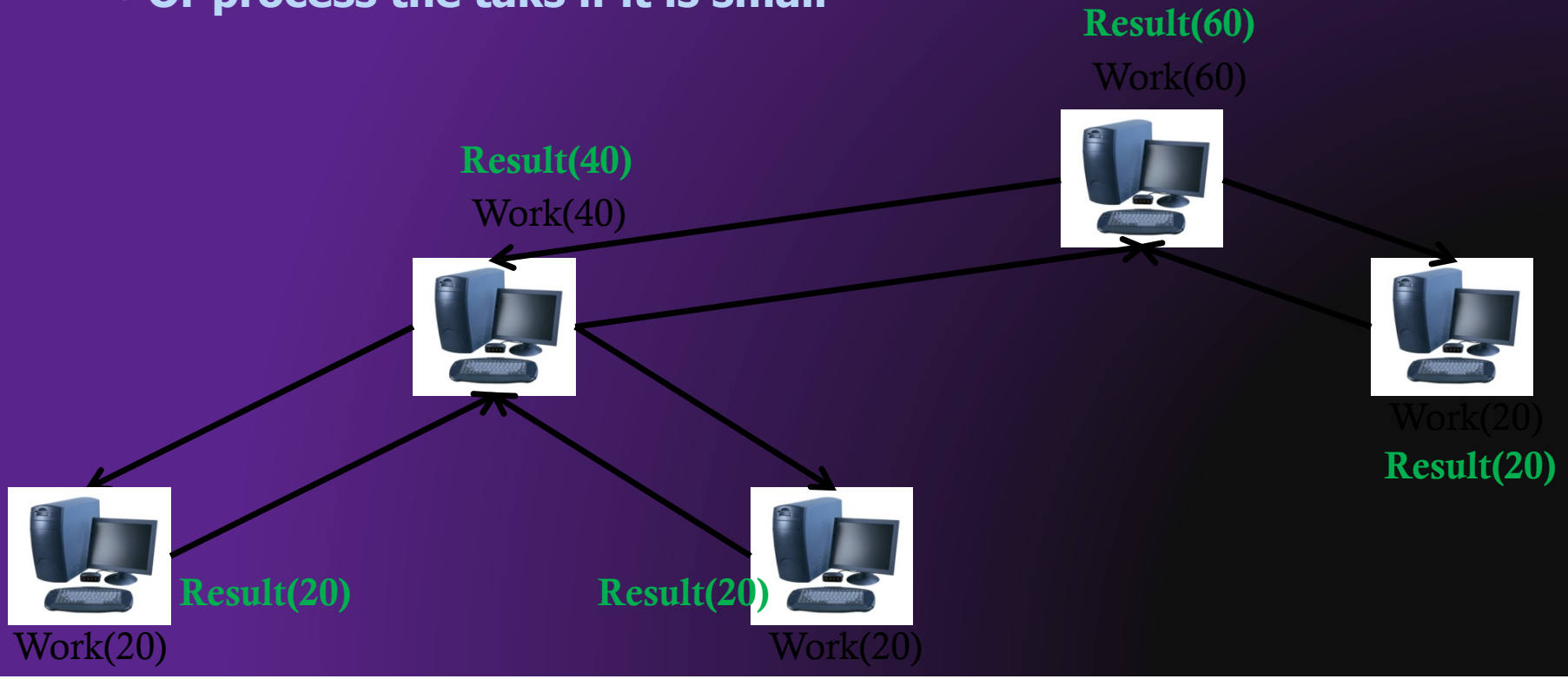


Parallel Strategies

- Pipeline
  - Scalabillity is limited by the number of stages
  - Synchronization may lead to bubbles
    - Slow sender
    - Fast receiver
  - Difficult to use on heterogenous platforms

Parallel Strategies

- Divide and Conquer
  - Recursevely partion task on roughly equal sized tasks
  - Or process the taks if it is small



Parallel Strategies

- Divide and Conquer
  - More scalable
  - Possible to use replicated branches
  - In practice is difficult to split tasks
  - Suitable for branch and bound algorithms

MPI Programming

- **Installing**
  - Some common MPI implementations, all free:
    - OpenMPI

      http://www.open-mpi.org/

    - MPICH-2 http://www.mcs.anl.gov/research/projects/mpich2/

    - LAM/MPI

      http://www.lam-mpi.org/

MPI Programming

- **Installing**
  - I'm using MPICH-2
  - Installed in Ubuntu 10.04 Lucid Lynx with

```
$ sudo apt-get install mpich2
```

  - Should work for most Debian based distributions
  - Must create a local configuration file

```
$ echo "MPD_SECRET_WORD=ChangeMe" > ~/.mpd.conf
```

MPI Programming

- Test program

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char **argv){

/* Initialize MPI */
MPI_Init(&argc, &argv);

printf("Test Program\n");

/* Finalize MPI */
return MPI_Finalize();
}
```

MPI Programming

- **Compiling**

  – Compiled with gcc, but a mpicc script is provided to invoke gcc with specific MPI options enabled

```
$ mpicc mpi_program.c -o my_mpi_executable
```

  – Executed with a specital script

```
$ mpirun -np 1 my_mpi_executable
$ mpirun -np 2 my_mpi_executable
$ mpirun -np 3 my_mpi_executable
```

MPI Programming

- **Running**
  - Compiled with gcc, but a mpicc script is provided to invoke gcc with specific mpi functions

```
$ mpicc mpi_program.c -o my_mpi_executable
```

  - For a complete list of parameters try

```
$ man mpicc
```

  - Executed with a specital scrip

```
$ mpirun -np 2 my_mpi_executable
```

MPI Programming

- **How many processing units are available?**

```
int MPI_Comm_size(MPI_Comm comm, int *psize)
```

  - **comm**
    - Group of process to communicate
    - For grouping all process use **MPI_COMM_WORLD**
  - **psize**
    - Passed as reference will return the total amoung of proccess in this communicator

MPI Programming

- Exercise 1 – Hello World
  - Create program that prints hello world and the total number of available process on the screen
  - Use –np with a variable number to verify that your program is working

MPI Programming

- **Assigning Process Roles**

```
int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

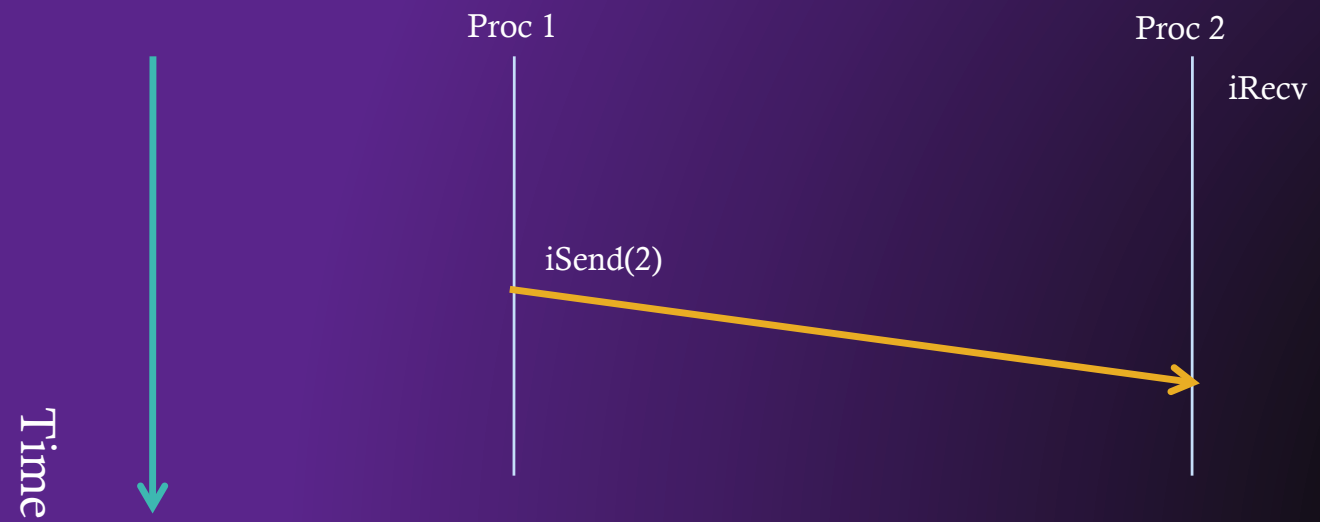
  - **comm**
    - Group of process to communicate
    - To group all available process use `MPI_COMM_WORLD`
  - **rank**
    - Passed as reference will return the unique ID of the calling process in this communicator

MPI Programming

- Exercise 2 – Who am I
  - If I am process 0
    - Prints: “hello world”
  - else
    - Prints: “I’m process <ID>”
    - Replacing <ID> by the process rank

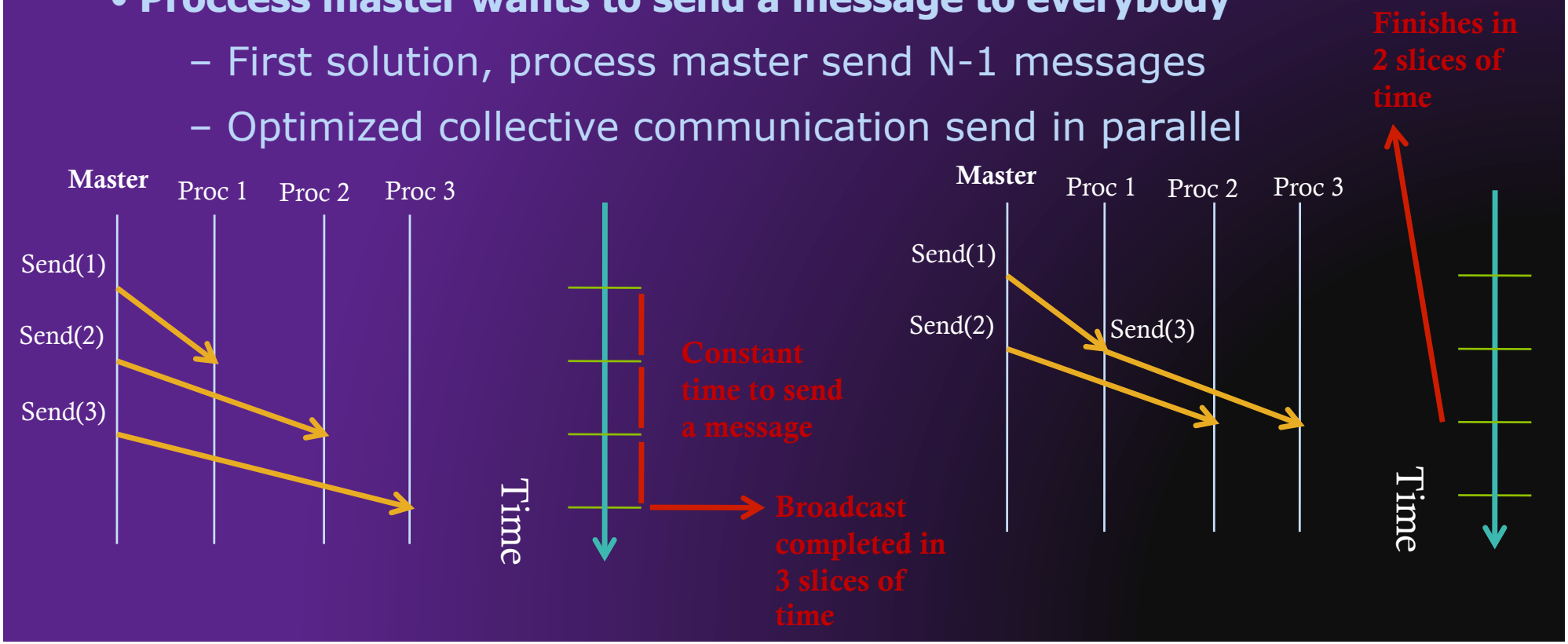
MPI One-to-one Communication

- **Assynchronous/Non-Blocking**
  - Process signs it is waiting for a message
  - Continue working meanwhile



MPI Collective Communication

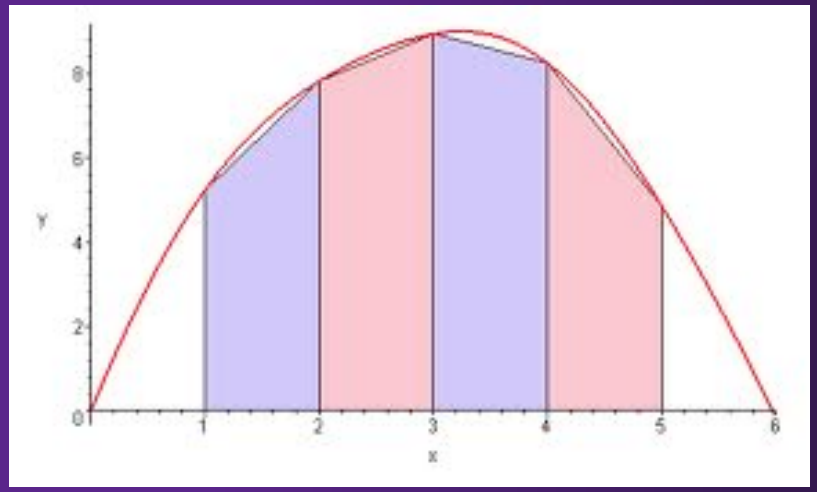
- **Proccess master wants to send a message to everybody**
  - First solution, process master send N-1 messages
  - Optimized collective communication send in parallel



Ejercicio de Clase

- Teniendo en cuenta la forma trapezoidal para integrar la formula :

$$\pi = \int_{o}^{1} \frac{4}{(1+x^2)} dx$$



Para Observar y Ejecutar

- http://people.sc.fsu.edu/~jburkardt/cpp_src/mpi/mpi.html
- http://www.slac.stanford.edu/comp/unix/farm/mpi.html
- http://www.mcs.anl.gov/research/projects/mpi/usingmpi/examples/main.htm