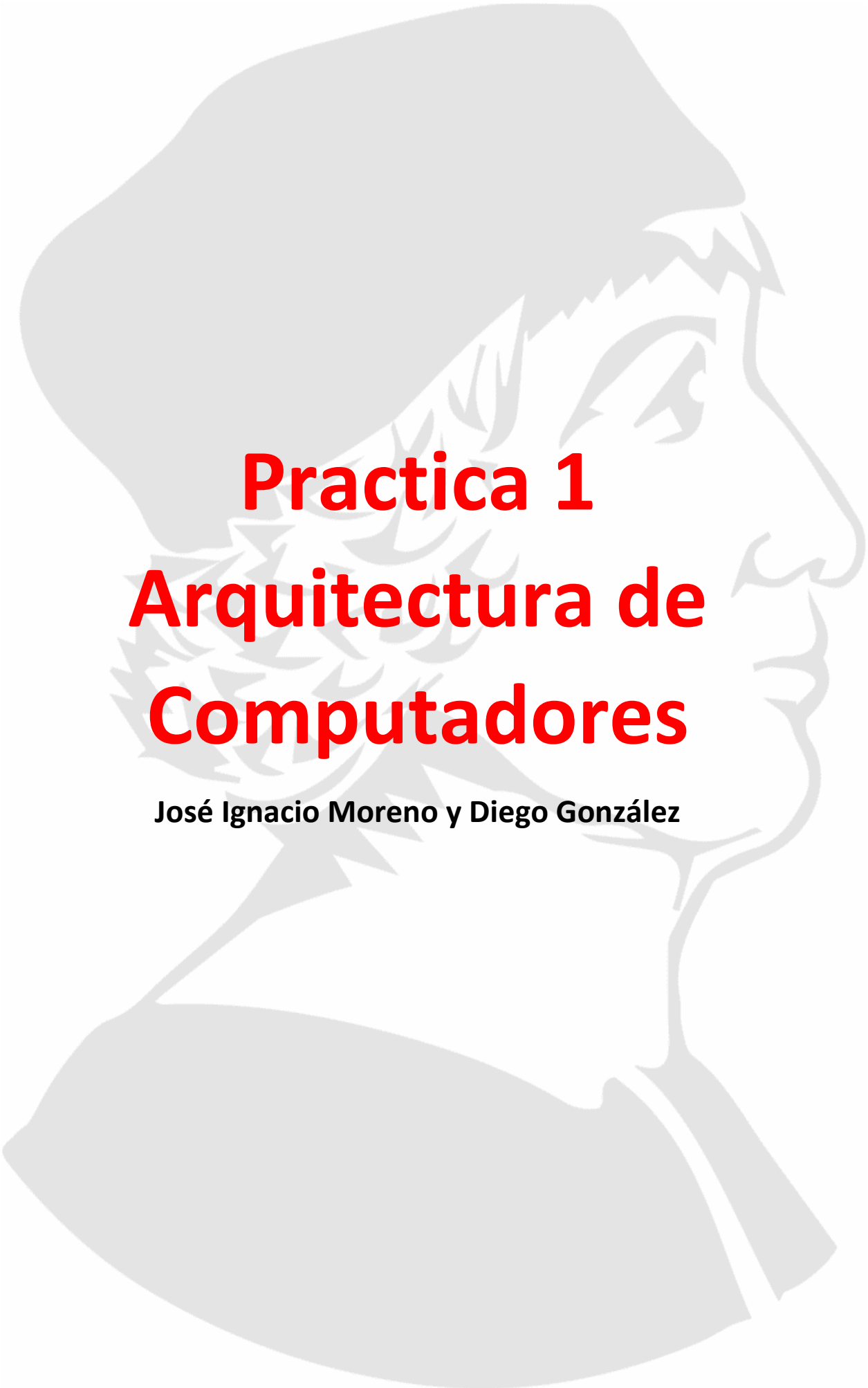


Practica 1
Arquitectura de Computadores

José Ignacio Moreno y Diego González

EJERCICIO 1:

En este ejercicio se pide mostrar por pantalla el proceso con el total de las tareas que le pertenecen. Además de eso, mostrar el tiempo de ejecución en cada caso.

En el código hemos cogido la instrucción MPI_Init() que nos ayud a inicializar la primera llamada a nuestro proceso. Con MPI_Comm_size() y MPI_Comm_rank(), hemos cogido las variables del rango y del proceso correspondiente. Posteriormente, con un print hemos mostrado los resultados por pantalla. Aparte de eso, tenemos una función llamada MPI_Wtime(), que calcula el tiempo de ejecución de los procesos, haciendo uso de loa función MPI_Barrier(), y sin ella. Mostramos los resultados más abajo.

1. ¿Los procesos imprimen el hola mundo y el tiempo en orden? ¿A qué se debe esto?:

No, los procesos imprimen por pantalla los datos por orden de llegada, ninguna tiene más prioridad por lo que el primer proceso que se ejecute de 4 procesos, por ejemplo, el 1, 2, 3, 4, puede que el proceso 4 sea el primero que se ejecute, después el 2 y así con todos.

2. ¿Qué pasa con el tiempo de ejecución si eliminamos las barreras (MPI_Barrier())? ¿Y con el orden de ejecución? ¿Porqué?:

Lo que ocurre con el tiempo de ejecución es que es inferior que si utilizamos esas barreras ya que cuando lo tenemos bloquea al proceso hasta que todos los procesos pertenecientes al comunicador especificado lo ejecuten. Y el orden de ejecución sigue siendo el mismo puede ser cualquier orden y la explicación de la pregunta anterior.

Foto sin utilizar MPI_Barrier():

```
Hola mundo, soy el proceso 0 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 8 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 2 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 1 de un total de 10 y tarda 0.000002
Hola mundo, soy el proceso 3 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 5 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 9 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 6 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 7 de un total de 10 y tarda 0.000001
Hola mundo, soy el proceso 4 de un total de 10 y tarda 0.000001
```

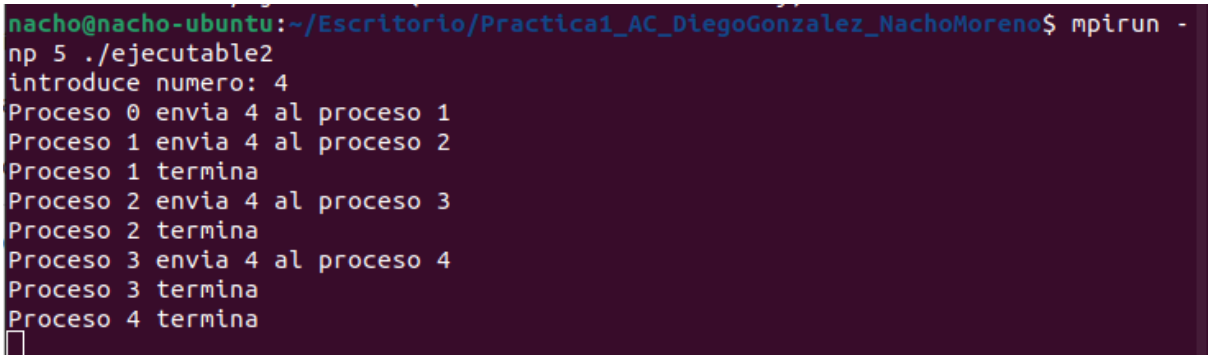
Foto utilizando MPI_Barrier():

```
n -np 10 ./ejecutable1
Hola mundo, soy el proceso 0 de un total de 10 y tarda 0.067618
Hola mundo, soy el proceso 1 de un total de 10 y tarda 0.062823
Hola mundo, soy el proceso 2 de un total de 10 y tarda 0.059988
Hola mundo, soy el proceso 4 de un total de 10 y tarda 0.099994
Hola mundo, soy el proceso 6 de un total de 10 y tarda 0.099957
Hola mundo, soy el proceso 7 de un total de 10 y tarda 0.099954
Hola mundo, soy el proceso 8 de un total de 10 y tarda 0.072071
Hola mundo, soy el proceso 9 de un total de 10 y tarda 0.075991
Hola mundo, soy el proceso 5 de un total de 10 y tarda 0.083973
```

EJERCICIO 2:

En este ejercicio se nos pedía implementar un programa usando MPI, donde un proceso X toma el dato del usuario y lo envía al siguiente nodo, que lo envía al siguiente y así hasta llegar al último.

De nuevo lo primero que hacemos es inicializar y posteriormente en size y rank, cogemos las variables de nproc y rango. También hemos creado una variable llamada n para que podemos saber que número introducimos por teclado. Myrank nos ayuda a saber cual es el id del proceso, y la de procesos el número del procesos que tenemos. En el primer if que nos encontramos en el código, nos dice que si el rango es igual a 0, es decir, que si estamos en el primer proceso, introduzcamos el numero para así poder enviárselo al siguiente nodo. Una vez este proceso ha terminado, terminan todos los procesos y una vez que el último recibe ya no envía y posteriormente se termina.



EJERCICIO 3:

Para este ejercicio, teníamos que modificar la implementación del anterior ejercicio para que el dato que introducimos dé tantas vueltas como este indicado en el anillo.

Hemos reutilizado parte del código del ejercicio 2. De nuevo en el if, en caso de que el rango sea 0, significa que nos encontramos en el proceso primero, y si esto se cumple, pedirá el programa al usuario que introduzca el número de vueltas del anillo. Una vez hecho esto, el dato introducido lo enviará al proceso inicial y así poder comenzar el ciclo del anillo.

Dicho lo anterior, pasamos al bucle do-while, que se hará una y otra vez siempre y cuando el numero sea mayor que 0. En caso contrario, saldrá del bucle. Para contar el número del vueltas que llevamos, lo hacemos con el número que hemos introducido por el teclado, y le vamos restando 1. Además de eso, podemos ver que el rango nos coge el id del proceso y seguirá así hasta que se acabe.

```
nacho@nacho-ubuntu:~/Escritorio/Practica1_AC_DiegoGonzalez_NachoMoreno$ mpirun -np 5 ./ejecutable3
Introduce el numero de vueltas al anillo: 3
Proceso 0 envia 3 al proceso 1
Proceso 1 ha recibido 3
Proceso 1 envia 3 al proceso 2
Proceso 2 ha recibido 3
Proceso 2 envia 3 al proceso 3
Proceso 3 ha recibido 3
Proceso 3 envia 3 al proceso 4
Proceso 4 ha recibido 3
Proceso 4 envia 3 al proceso 0
Proceso 0 ha recibido 3
Proceso 0 descuenta una vuelta
Proceso 0 envia 2 al proceso 1
Proceso 1 ha recibido 2
Proceso 1 envia 2 al proceso 2
Proceso 2 ha recibido 2
Proceso 2 envia 2 al proceso 3
Proceso 3 ha recibido 2
Proceso 3 envia 2 al proceso 4
Proceso 4 ha recibido 2
Proceso 4 envia 2 al proceso 0
Proceso 0 ha recibido 2
Proceso 0 descuenta una vuelta
Proceso 0 envia 1 al proceso 1
Proceso 1 ha recibido 1
Proceso 1 envia 1 al proceso 2
Proceso 2 ha recibido 1
Proceso 2 envia 1 al proceso 3
Proceso 3 ha recibido 1
Proceso 3 envia 1 al proceso 4
Proceso 4 ha recibido 1
Proceso 4 envia 1 al proceso 0
Proceso 0 ha recibido 1
Proceso 0 descuenta una vuelta
Proceso 0 envia 0 al proceso 1
Proceso 0 termina
Proceso 1 ha recibido 0
Proceso 1 envia 0 al proceso 2
Proceso 1 termina
Proceso 2 ha recibido 0
Proceso 2 envia 0 al proceso 3
Proceso 2 termina
Proceso 3 ha recibido 0
Proceso 3 envia 0 al proceso 4
Proceso 3 termina
Proceso 4 ha recibido 0
Proceso 4 envia 0 al proceso 0
Proceso 4 termina
```

1- ¿Qué desventaja se aprecia en este tipo de comunicaciones punto a punto a medida que aumentan el número de procesos requeridos? Razonar la respuesta:

La desventaja que se aprecian en las comunicaciones punto a punto con muchos procesos es que reducen su rendimiento, además de que no son escalables, es decir, puede perder calidad de la información, no maneja el crecimiento continuo del trabajo de manera fluida.

2. ¿Cómo podría mejorar el sistema y su implementación? Razonar la respuesta.

Una forma es usar procesos en el anillo. En todo caso, se dedicará a dar todas las vueltas necesarias hasta completar el número de vueltas proporcionado por teclado.