Programación de Sistemas Distribuidos

Curso 2013/2014

Práctica 2
Message Passing Interface (MPI)

1. Introducción

En los siguientes ejercicios, se pide utilizar el lenguaje C junto con una implementación del API MPI para crear distintos programas paralelos. En todos los programas paralelos debe suponer que un proceso actúa como coordinador; llamemos a los demás procesos trabajadores. Para ahorrar en tiempo de depuración del código C, donde sea posible se pueden utilizar tipos estáticos.

2. Multiplicación de matrices cuadrados

En este apartado se pide crear distintos programas que realicen la multiplicación de dos matrices cuadradas. Supongamos que, al inicio, solo uno de los procesos (el coordinador) tiene los datos de las dos matrices que se quiere multiplicar. Para simplificar los ejercicios, basta con que el coordinador empiece generando estas dos matrices, por ejemplo, rellenándolas con números enteros contiguos.

En todos los ejercicios, debe usar la función `MPI_Wtime()` para cronometrar el tiempo de ejecución del programa y debe probar lanzar su programa instanciando el parámetro del número de procesos con cada uno de los siguientes valores (utilizando las mismas matrices en cada caso, de preferencia bastante grandes):

- el número de cores físicos que tenga el ordenador; en Linux, se puede conocer este dato mediante el comando:
  ```
  grep "^core id" /proc/cpuinfo | sort -u | wc –l
  ```
- el número de cores lógicos que tenga el ordenador; en Linux, se puede conocer este dato mediante el comando:
  ```
  grep -c ^processor /proc/cpuinfo
  ```
- un número mayor que el número de cores lógicos.

y anotando los resultados.

A ser posible, debe ejecutar sus programas utilizando una versión *reciente* de OpenMPI para poder observar la asignación de procesos a cores mediante el parámetro del comando `mpiexec -report-bindings` (ver transparencias).

Ejercicio 1

Se pide implementar un programa secuencial.

Ejercicio 2

Se pide paralelizar el programa del ejercicio 1 utilizando las siguientes funciones MPI:

- `MPI_Bcast` para enviar uno de los operandos del coordinador a los trabajadores;
- `MPI_Send`, que ejecutará el coordinador, y `MPI_Recv`, que ejecutarán los trabajadores, para enviar un trozo distinto del otro operando del coordinador a cada trabajador;
- `MPI_Send`, que ejecutarán los trabajadores, y `MPI_Recv`, que ejecutará el coordinador, para enviar el resultado calculado por cada trabajador al coordinador.

Ejercicio 3

Se pide modificar el programa del ejercicio 2 para que utilice las siguientes funciones:

- `MPI_Bcast` para enviar uno de los operandos del coordinador a los trabajadores;
- `MPI_Isend`, que ejecutará el coordinador, y `MPI_Recv`, que ejecutarán los trabajadores, para enviar un trozo distinto del otro operando del coordinador a cada trabajador,
- `MPI_Send`, que ejecutarán los trabajadores, y `MPI_Irecv`, que ejecutará el coordinador, para enviar el resultado calculado por cada trabajador al coordinador.

Con el fin de obligar al proceso coordinador a esperar hasta que terminen los `MPI_Isend` antes de emprender su parte del trabajo y hasta que terminen los Irecv antes de emprender la impresión de los resultados, se puede utilizar `MPI_Waitall()` (habiendo declarado y usado previamente un array de `MPI_Request`).

Ejercicio 4

Se pide modificar el programa del ejercicio 3 para que utilice las siguientes funciones:

- `MPI_Bcast` para enviar uno de los operandos del coordinador a los trabajadores,
- `MPI_Scatter` para enviar un trozo distinto del otro operando del coordinador a cada proceso,
- `MPI_Gather` para recoger en el coordinador el resultado calculado por cada proceso.

Observe que solo se debe usar `MPI_Scatter` si el tamaño de los datos (en este caso, la matriz que se divide en trozos) es divisible por el número de procesos. Por tanto, si no es el caso, el programa debería imprimir un mensaje adecuado y terminar antes de llamar a `MPI_Bcast`.

Ejercicio 5 (opcional)

Se pide modificar el programa del ejercicio 4 para que utilice las siguientes funciones:

- `MPI_Bcast` para enviar uno de los operandos del coordinador a los trabajadores,
- `MPI_Scatterv` para enviar un trozo distinto del otro operando del coordinador a cada proceso,
- `MPI_Gatherv` para recoger en el coordinador el resultado calculado por cado proceso.

Nótese que con `MPI_Scatterv`, el tamaño de las matrices no es necesariamente divisible por el número de procesos.

3. Cálculo del producto interior de dos vectores

En este apartado se pide crear distintos programas que realicen el cálculo del producto interior de dos vectores (del mismo tamaño).

Ejercicio 6

Se pide implementar un programa secuencial.

Ejercicio 7

Se pide implementar un programa paralelo utilizando las siguientes funciones MPI:

- `MPI_Scatter` para enviar un trozo distinto de cada vector del coordinador a cada uno de los procesos
- `MPI_Reduce` para recoger en el coordinador el resultado calculado por cada proceso y calcular un resultado global a partir de estos resultados individuales.

Observe que solo se debe usar `MPI_Scatter` si el tamaño de los datos (en este caso, los vectores) es divisible por el número de procesos. Por tanto, si no es el caso, el programa debería imprimir un mensaje adecuado y terminar antes de llamar de a `MPI_Scatter`.

Ejercicio 8 (opcional)

Se pide implementar un programa paralelo utilizando las siguientes funciones MPI:

- `MPI_Scatterv` para enviar un trozo distinto de cada vector del coordinador a cada uno de los procesos
- `MPI_Reduce` para recoger en el coordinador el resultado calculado por cada proceso y calcular un resultado global a partir de estos resultados individuales.

Nótese que con `MPI_Scatterv`, el tamaño de los vectores no es necesariamente divisible por el número de procesos.

Entrega

El código de cada ejercicio se tiene que entregar por Campus Virtual en un solo archivo comprimido mediante zip antes de las 12 de la noche del 10 de febrero de 2014. El archivo ha de incluir una breve memoria explicando las soluciones implementadas y lo que ha observado de los tiempos de ejecución en los casos pedidos. La nota máxima que puede obtener un alumno que no entrega los ejercicios opcionales es 7 puntos.