Algoritmos Paralelos

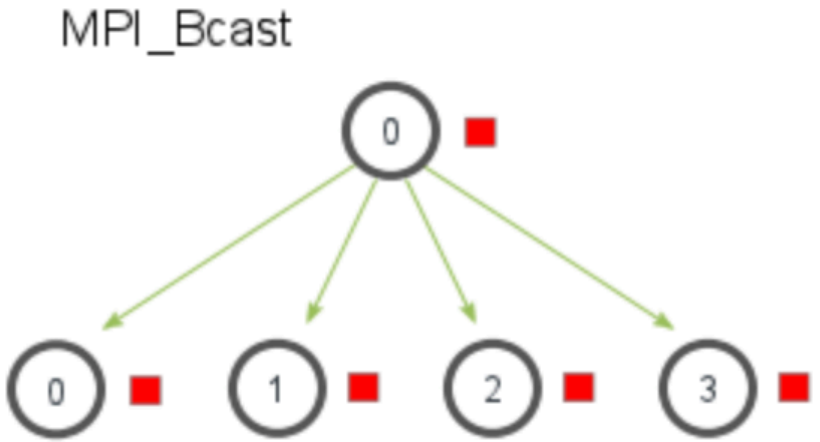
Laboratorio

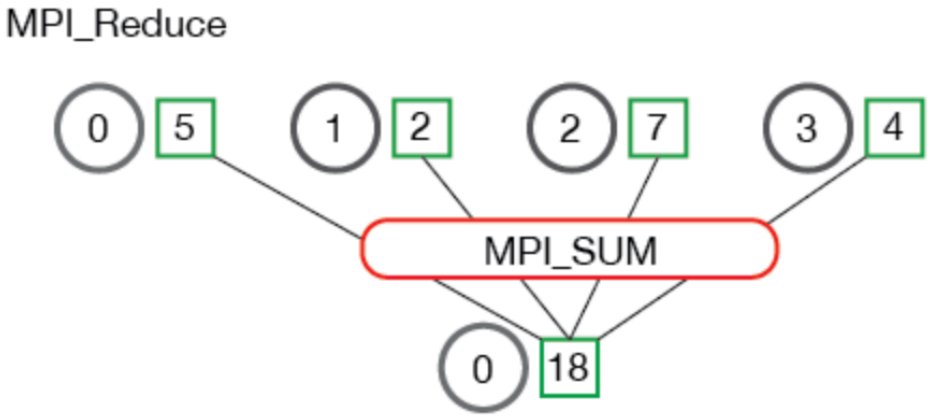
(04.09.15)

Prof. J.Fiestas

Ejemplo 1: Bcast y reducción

Elementos de cada proceso son copiados desde el root y se ejecuta una operación, cuyo resultado se escribe de nuevo en el root.





Ejemplo 1: Bcast y reducción

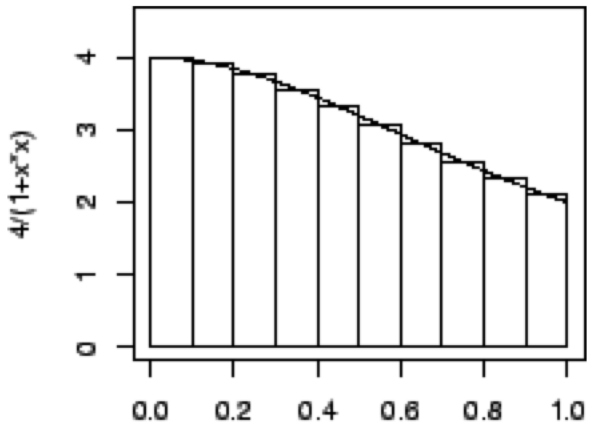
```
int main(int argc, char *argv[]) {
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &me);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
....
MPI_Bcast(&choice,1,MPI_INT,0,MPI_COMM_WORLD);
....
MPI_Reduce(&data, &res, DATA_SIZE, MPI_INT, MPI_SUM,
                    0, MPI_COMM_WORLD);
....
MPI_Finalize();
}
```

Ejercicio 4: Calculo de PI por integración

Calcular el valor de PI a través de la fórmula de aproximación.

- Programar en serie
- Utilizar MP_Bcast para distribuir el número de intervalos a integrar a todos los procesos

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \sim \frac{1}{n} \sum_{i=1}^{n} \frac{4}{1+\left(\frac{i-0.5}{n}\right)^2}.$$

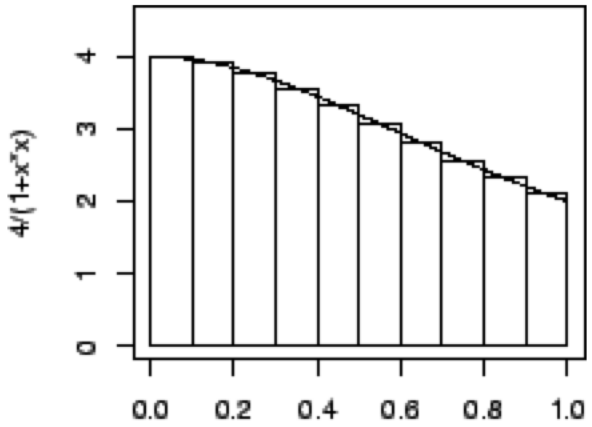


Ejercicio 4: Calculo de PI por integración

Calcular el valor de PI a través de la fórmula de aproximación.

- Asignar correctamente los límites de integración para cada proceso
- Usar MPI_Reduce para recolectar el cálculo de cada proceso y sumarlo en el resultado de Pi
- Calcular el error con respecto al valor de Pi de 3.14159265358979323846264 3

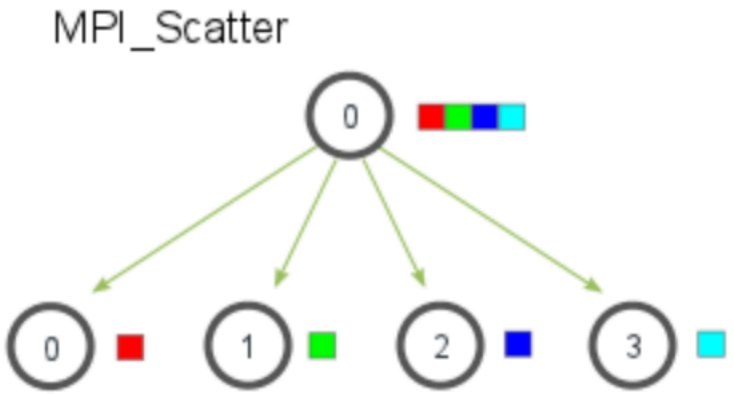
$$\pi = \int_0^1 \frac{4}{1+x^2} \mathrm{d}x \sim \frac{1}{n} \sum_{i=1}^{n} \frac{4}{1+\left(\frac{i-0.5}{n}\right)^2}.$$

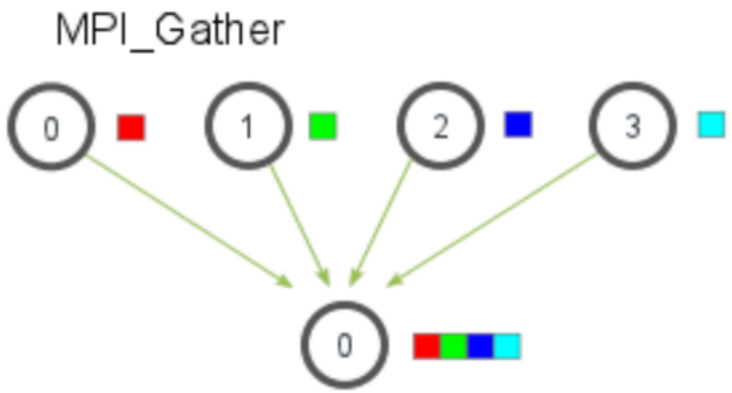


Ejercicio 5: Calculo del promedio

Se distribuye mensajes de una sola fuente a cada proceso en el grupo

Se recopila informacion de cada proceso a un solo destino.

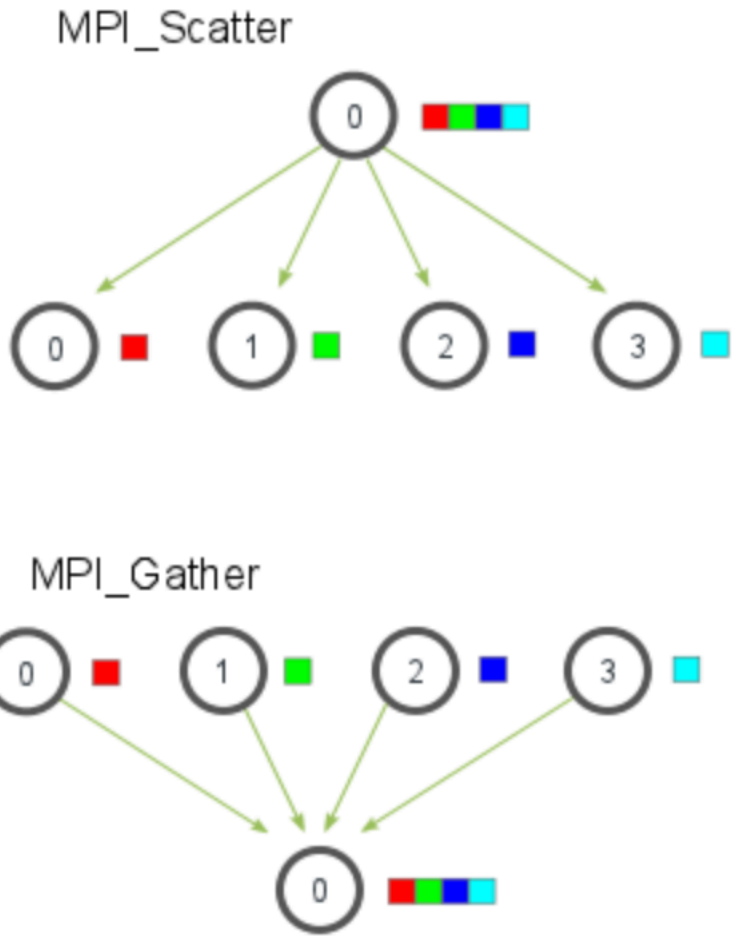




Ejercicio 5: Calculo del promedio

Calcular el promedio de números en un array. Para ello:

- Generar un array con números random
- Asignar a cada proceso un número equivalente de números (**MPI_Scatter**)
- Cada proceso calcula el promedio de su muestra
- Se agrupan los promedios (**MPI_Gather**) en el nodo principal, y este calcula el promedio global



Ejemplo 2: Send/Recv

```
typedef struct _MPI_Status {
  int count;
  int cancelled;
  int MPI_SOURCE;
  int MPI_TAG;
  int MPI_ERROR;
} MPI_Status, *PMPI_Status;
```

MPI_ANY_SOURCE
MPI_ANY_TAG

e.g.

```
MPI_Status stat;
MPI_Recv(&buf,1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD,&stat);
```

Ejemplo 2: Send/Recv (1 a 1)

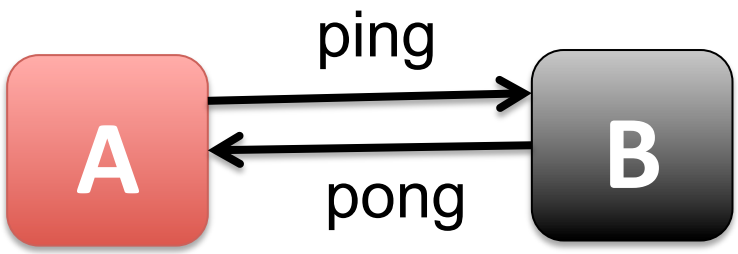
```
int main(int argc, char *argv[]) {
MPI_Init(&argc,&argv);
....
 if (world_rank == 0) {
    // If we are rank 0, set the number to -1 and send it to process 1
    number = -1;
    MPI_Send(&number, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
  }
  else if (world_rank == 1) {
    MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
...
  }
MPI_Finalize();
}
```

Ejemplo 3: Send/Recv (1 a np)

```
int main(int argc, char *argv[]) {
MPI_Init(&argc,&argv);
....
if (myid==0) {
      buf=1;
      for (i=1; i<numprocs; i++) {
         MPI_Send(&buf,1,MPI_INT,i,0,MPI_COMM_WORLD);
      }
...
   }
   else {
      MPI_Recv(&buf,1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD,&stat);
}....
MPI_Finalize();
```

Ejercicio 6: Ping-pong

Envio y recibo de mensajes entre dos procesos



Programe el algoritmo de Ping-Pong entre dos procesos tal que proceso A y B intercambien un array de floats.
Variar la dimension del array de 1 a 2^{18}, duplicando la dimensión en cada iteración.
Mida los tiempos de ejecución utilizando MPI_Time()

Ejercicio 7: Algoritmo del anillo

- Cada proceso envia un mensaje al vecino posterior
- Cada proceso recibe un mensaje del vecino anterior
- El nodo principal (0) recibe mensajes solo del último proceso
- El mensaje consistirá en una suma de usuarios = 1000 + 100*rank, donde rank es el entero identificador de cada proceso
- Utilizar np =7