

Puzle, Parte 1

Mikel Dalmau

4 de Abril de 2016

1 Comunicaciones colectivas

2 Ejercicios, primera parte del puzle.

2.1 P1.1

Hay que repartir un vector de N elementos entre npr procesos. Completa el programa serie *P11-distribute0.c*, para que genere el tamaño de cada trozo del vector y el desplazamiento desde el origen del vector al comienzo de cada trozo, en estos dos casos:

- los posibles restos se añaden al último trozo
- los posibles restos se añaden uno a uno a diferentes trozos

2.1.1 a.

Inicialmente calculo $Nloc$ y $remainder$.

```
//Compute Nloc and remainder
Nloc = floor(((double)N)/((double)npr));
remainder = N - Nloc*npr;
```

Este caso es sencillo y se resuelve con el siguiente bucle y las asignaciones finales.

```
//We
for(i=0; i<npr-1; i++){
    size[i] = Nloc;
    shift[i] = i*Nloc;
}
//Finally we charge the last process with the remainder
size[npr-1] = Nloc + remainder;
shift[npr-1] = (npr-1)*Nloc;
```

2.1.2 b.

En este segundo caso he comenzado distribuyendo la carga del resto entre los primeros procesadores, luego, las cargas distintas entre procesos no permiten el cálculo de $shift$ usado anteriormente, por lo que tomo las referencias de tamaño y $shift$ calculadas en la anterior iteración para calcular el $shift$, sabemos donde empezamos porque sabemos dónde termina el anterior.

```
//Value assignment to first process
size2[0] = Nloc;
shift2[0] = 0;
```



```
//Distribution of the remainder among the first processes
i = 0;
while(remainder){
    size2[i] += 1;
    remainder -= 1;
    i++;
}

//Distribute the rest of the vector among the processes
for(i=1; i<npr; i++){
    size2[i] += Nloc;
    shift2[i] = shift2[(i-1)] + size2[(i-1)];
}
```

2.2 P1.2

El programa *P12-inteser.c* calcula el valor de una integral mediante el conocido método de sumar las áreas de n trapecios bajo la curva que representa una función. A mayor valor de n , más preciso el resultado.

Completa el programa MPI *P12-inteser.c* para realizar esa misma función entre P procesos, utilizando funciones de comunicación colectiva. Compara el resultado con el de la versión serie.

Bibliografía

- [1] Pacheco P.: *An Introduction to Parallel Programming*. Morgan Kaufmann, 2011. Capítulo 3, apartado 4.