Programación Paralela
2021-0

Prof. José Fiestas

1). Haga el análisis de complejidad del algoritmo quicksort mostrado, en el cual cada uelemento de la lista es asignado a un proceso

```
procedure theoQSort(d,i,p)
  if(p=1)return;
  j := common random element from 0..p-1 for partition;
  v := d@j; //Pivot
  f := d ≤ v;
  j := Σ_{k=0}^{i} f@k; //Präfixsumme
  p' := j@(p-1);
  if(f)send d to PE j-1;


 else send d to PE p'+i-j;
 receive d;
 if(i<p'){
   join partition "left";
   theoQSort(d,i,p');
 }else{
   join partition "right";
   theoQSort(d,i-s,p-p');
 }
```

Ahora extienda el concepto para un $n >> 1$
Solución:
los procesos se separan en particiones. En una partición, el proceso j contiene el pivot, el cual es enviado a todos los demás procesos (**Broadcast**). Se utiliza luego una **suma de prefijos** para numerar elementos menores/mayores que el pivot. Los primeros p'-1 procesos contienen los p'-1 elementos menores que el pivot. Esta partición se vuelve a dividir en dos nuevas particiones, donde se aplica de nuevo el algoritmo en forma recursiva.
Es decir, se realizan un broadcast y una reducción en un tiempo O($T_{start}log(p)$), y repitiendo un proporcional a la profundidad del árbol de O(log(p)), se obtiene O($T_{start}log^2(p)$)
En caso de $n >> 1$, cada proceso p recibe n/p elementos, y estos, a su vez, son transportados entre procesos en un tiempo log(p). I.e.
O($\frac{n}{p}(logN + T_{byte}log(p)) + T_{start}log^2(p)$)
Note que esto implica un paradigma de memoria distribuída.

2) Dado el codigo secuencial

```
double calculo(int i,int j,int k)
{
  double a,b,c,d,x,y,z;
  a = T1(i);
  b = T2(j);
  c = T3(k);
  d = T4(a+b+c);
  x = T5(a/d);
  y = T6(b/d);
  z = T7(c/d);
  return x+y+z;
}
```

Dibuje el grafo de dependencias entre tareas