Examen Multiprocesadores

Primavera 2005

Preguntas/Problemas (sin apuntes 40')

Responder razonadamente las siguientes cuestiones:

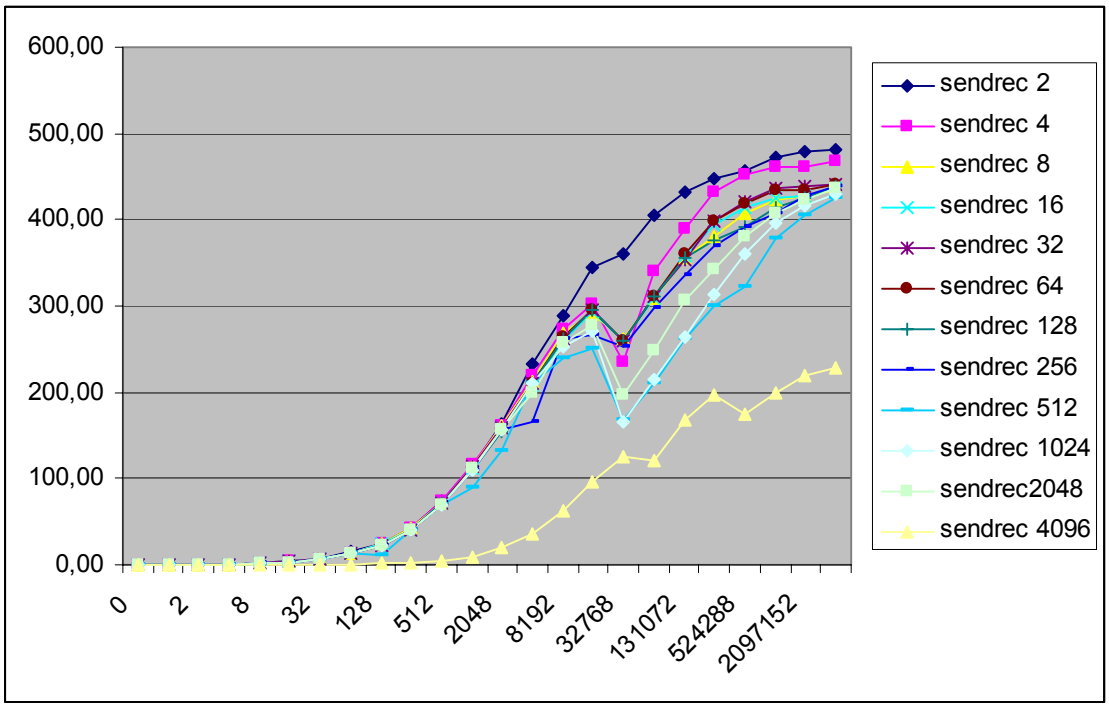
1. Describir la funcionalidad de las llamadas MPI:
   - MPI_AllReduce
   - MPI_Alltoall
2. Cual es el diámetro de una malla 3D de k3 nodos? Y de un toro del mismo numero de procesadores?
3. En un sistema de coherencia basado en directorio para qué sirve las transacciones NACK?

Problema: MareNostrum

MareNostrum tiene 2400 nodos SMP de dos procesadores y una red óptica con enlaces de ancho de banda nominal de 250 MB/s para paso de mensajes entre nodos. Hemos ejecutado un benchmark en el que varios procesos se envían mensajes de un cierto tamaño. El patrón de comunicaciones es un shift circular. En cada fase un proceso envía un mensaje al siguiente y recibe uno del anterior (los dos del mismo tamaño). La asignación de procesos a nodos es cíclica (primer proceso al primer nodo, segundo al segundo, …. proceso 2400 al último, 2401 al primero,….) . La siguiente curva presenta los anchos de banda medidos (bytes totales que entran y salen de un nodo dividido por el tiempo). El eje de abscisas es el tamaño del mensaje. Cada curva representa los valores obtenidos para un cierto número de procesadores (según la leyenda).

1. Comentar las curvas obtenidas. Identificar regiones que merezcan una explicación especial e indicar posibles causas del comportamiento observado. Si se os dieran los valores numéricos, indicar qué tipos de cálculos, modelos,… podrían hacerse para validar alguna de las hipótesis anteriores.

2. Escribir la estructura del programa de prueba



Problema: Sort

Tenemos el siguiente algoritmo de ordenación por el método de la burbuja.

```
convergence =FALSE;
i=0;
while (!convergence && i<N)  {
  convergence = TRUE;
   for (j=N-1;j>i;j--) {
      if (a[j]<a[j-1]) {
         tmp= a[j-1];
         a[j-1]=a[j];
         a[j]=tmp;
         convergence=FALSE;
      }
   }
   i++;
}
```

1. Describir el funcionamiento (no estructura del código) de una paralelización MPI que conserve el mismo orden de operaciones que el algoritmo original.
2. Comentar si es posible que el algoritmo tenga desbalanceo de carga en sus distintas fases. Puede depender de los valores iniciales en a? Suponer que N >>#procesos
3. Escribir el código MPI.

4. Se realizó la siguiente paralelización del mismo algoritmo usando OpenMP.

```
convergence =FALSE;
i=0;
while (!convergence && i<N)  {
  convergence = TRUE;
#pragma omp parallel private(j,tmp)
   for (j=N-1;j>0;j--) {
      if (a[j]<a[j-1]) {
         tmp= a[j-1];
         a[j-1]=a[j];
         a[j]=tmp;
         convergence=FALSE;
      }
   }
   i++;
}
```

   Comentar la necesidad o no de privatizar convergence.

5. Se observa que la paralelización anterior da a veces resultados incorrectos (i.e. vector de entrada con valores no repetidos y salidas repetidas).Si se ejecuta (a.out) mediante la linea:

   a.out | sort -n | uniq | wc

   (sort –n ordenación numérica, uniq elimina líneas repetidas, wc cuenta líneas) en una máquina SMP de 16 CPUs pero que tiene algo de carga puede salir:

   N=1024, OMP_NUM_THREADS=1: 1024 lineas unicas
   N=1024, OMP_NUM_THREADS=2: 1024 lineas unicas
   N=1024, OMP_NUM_THREADS=4: 1024 lineas unicas
   N=1024, OMP_NUM_THREADS=8: 1014 lineas unicas
   N=1024, OMP_NUM_THREADS=8: 1003 lineas unicas
   N=1024, OMP_NUM_THREADS=16: 993 lineas unicas
   N=1024, OMP_NUM_THREADS=16: 1007 lineas unicas

   N=15000, OMP_NUM_THREADS=1: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=2: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=2: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=4: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=4: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=8: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=8: 15000 lineas unicas
   N=15000, OMP_NUM_THREADS=16: 14998 lineas unicas
   N=15000, OMP_NUM_THREADS=16: 14993 lineas unicas

   Dar una explicación de los resultados.

Problema: mas Sort

Tenemos otro programa de ordenación paralelizado con OpenMP para ejecutar en una máquina SMP en que en una línea de cache caben 8 elementos del vector a. Los vectores a y b están alineados a línea de cache..

```
#pragma omp parallel for private (i,j,tmp)
   for (i=0;i<N;i++) {
      posicion=0;
      for (j=0;j<i;j++) if (a[j]<=a[i])  posicion++;
      for (j=i+1;j<N;j++) if (a[j]<a[i])  posicion++;
      b[posicion]=a[i];
   }
```

1. Está bien paralelizado el programa? Por que? Cómo habría que modificarlo?
2. Pues me funciona!!!! Por que?
3. Si N=256 y se hace la siguiente inicialización de a[i]

   ```
   for (i=0;i<N;i++) a[i]=(8*i)%N;
   ```

   Se producirá false sharing? Explicar la respuesta.

4. Y si la inicialización es

   ```
   for (i=0;i<N;i++) a[i]=(i+N/2)%N;
   ```

5. Que característica del sistema basado en bus de ciclo partido como el descrito en clase podría ser beneficiosa para este algoritmo en el caso en que el vector a no quepa en la cache?