Examen Multiprocesadores

Otoño 2004

Preguntas/Problemas (sin apuntes 40')

Responder razonadamente las siguientes cuestiones:

- Describir y comparar las directivas OpenMP critical y atomic.
- Que es un programa moldable? Y maleable?
- Describir que tipos de hilos (y cuantos de cada tipo) pondrías en un bus para soportar la conexión de 8 procesadores a un máximo de 8 GB de memoria física con líneas de 128 bytes y el protocolo MESI de coherencia. Suponer que el bus no esta segmentado.
- Que es el diámetro, la distancia media y el ancho de banda de bisección de una red de interconexión.

Problema

Se tiene el programa paralelo:

```
         :
      real told(0:130,0:130,0:130, 16)
      integer nx(100),ny(100),nz(100),link(13000,2),count
      integer loc(100)
      integer numiters
      integer nblock

         :
      do while ((tres.gt.tol).and.(count.lt.numiters))
         count = count+1
         tres=0.0
C$OMP PARALLEL DO PRIVATE (iblock, res) REDUCTION (+:tres)
         do iblock=1,nblock
            call solve(a(loc(iblock)),nx(iblock),ny(iblock),
     *                 nz(iblock),res,tol, told(0,0,0,iblock))
            tres=tres+res
         enddo
c     perform inter block interactions
C$OMP PARALLEL DO PRIVATE (i, val)
         do i=1,nconnect
            val=(a(link(i,1))+a(link(i,2)))/2.0
            a(link(i,1))=val
            a(link(i,2))=val
         enddo
      enddo
         :
```

:

```
 subroutine solve(t,nx,ny,nz,res,tol,told)
 real t(nx,ny,nz),told(0:130,0:130,0:130)
 real res,tol
 integer count

 res=1000000.0
 count=0
 do while ((res.gt.tol*100).and.(count.lt.3 ))
    count = count+1
    res=0.0

    do k=1,nz
      do j=1,ny
        do i=1,nx
          told(i,j,k)=t(i,j,k)
        enddo
      enddo
    enddo

    do k=0,nz+1
      do j=0,ny+1
        told(0,j,k)=0
        told(nx+1,j,k)=0
      enddo
      do i=0,nx+1
        told(i,0,k)=0
        told(i,ny+1,k)=0
      enddo
    enddo

    do j=0,ny+1
      do i=0,nx+1
        told(i,j,0)=0
        told(i,j,nz+1)=0
      enddo
    enddo

    res=0.0

    do k=1,nz
       do j=1,ny
          do i=1,nx
             t(i,j,k)=(told(i,j,k-1)+told(i,j,k+1)+
+               told(i,j-1,k)+told(i,j+1,k)+
+               told(i-1,j,k)+told(i+1,j,k)+told(i,j,k)*6.0)/12.0
             res=res+(t(i,j,k)-told(i,j,k))**2
          enddo
       enddo
    enddo
    res=sqrt(res)
 enddo
```

La variable link(i,1..2) identifica elementos de un bloque que se corresponden al mismo elemento de otro bloque y por tanto se da como valor definitivo el promedio de lo calculado al tratar cada bloque. Suponer que cada procesador tiene una cache exageradamente grande que esta totalmente vacía al empezar el bucle while.

1. En qué partes del código se producirán fallos de cache?
2. En qué partes del código se producirán invalidaciones?
3. En qué partes del código se puede dar falsa compartición?
4. Que pasaría si para balancear ponemos schedule (dynamic)?
5. Se podría ejecutar el programa en 32 procesadores? Por qué? Que soluciones habría para hacer el código mas genérico?

Tenemos un sistema en el que no se garantiza consistencia secuencial. En su lugar, cada procesador puede cargar en su cache una línea para escritura sin que se invalide en otros. El directorio guarda el valor original de la línea y cuando un procesador se la devuelve parcialmente modificada el directorio actualiza solo los campos que ese procesador modificó.
Existe una instrucción especial en el lenguaje máquina del procesador cuyo efecto es garantizar que todas las líneas modificadas localmente son enviadas al directorio. El procesador no pasa a la instrucción siguiente hasta que tiene la garantía por parte del directorio de que se han llevado a cabo todas las invalidaciones y actualizaciones que eso implica.

6. Tiene alguna ventaja genérica este sistema de consistencia relajada?
7. Se puede ejecutar correctamente el programa anterior en este sistema? Qué habría que hacer?
8. Hay alguna directiva en OpenMP que soporte este tipo de sistemas.

Queremos hacer la versión MPI moldable. Para ello, una vez leído el fichero de entrada y antes de entrar en el bucle while, se ejecuta una función que de determina que bloques son ejecutados por cada proceso.

9. Dar la estructura del programa principal.

Un fichero de entrada indica que el problema tiene 7 bloques y de tamaño respectivo: 10000, 5000, 5000, 3000, 3000, 2000, 2000 elementos. El tiempo de cálculo es de 1 microsegundo/elemento. Suponer velocidad de comunicación muy alta (infinita) y overheads muy bajos (cero).

10. Que asignación de bloques debería hacerse para que la ejecución en 3 CPUs fuera eficiente?
11. Que eficiencia se podría conseguir en 6 CPUs?
12. Y en 10 CPUs?
13. Sería útil la versión que tuviera MPI a nivel externo y OpenMP a nivel interno? Que eficiencia se podría conseguir con 6 y 10 procesadores respectivamente?

Ejecutamos dos instancias idénticas del programa (solo MPI) con el fichero de entrada describiendo un problema que tiene tres bloques de 10000, 6000 y 5000 elementos respectivamente y que se ejecuta durante 1000 iteraciones. El sistema tiene 4 procesadores.

14. Cual es el tiempo medio de respuesta con planificación FIFO si la segunda aplicación llega al sistema 10 segundos después de la primera? Y con Gang (quantum 1 segundo)?
15. Cuanto duraría la ejecución total en una planificación de sistema FIFO si las dos aplicaciones llegan a la vez?
16. Si en este caso se usara Gang (quantum 1 segundo): tardaría lo mismo la ejecución total? Se tendría el mismo tiempo medio de respuesta? Que eficiencia se tendría?
17. Que componente/función descrito anteriormente debería incluirse en la versión MPI para hacerla moldable?
18. Se podría encontrar alguna planificación mas eficiente que las anteriores para el caso de llegar las dos instancias a la vez teniendo en cuenta la moldabilidad de las aplicación?