Cómputo paralelo con openMP y C

Sergio Ivvan Valdez Peña

Guanajuato, México.

13 de Marzo de 2012

Contenido

¿Que es cómputo paralelo?

Serial

- Suponga que Ud. fabrica zapatos, y necesita fabricar un “lote” de 100 pares.
- Suponga que cada par le toma un día de trabajo.
- Esto requiere de 100 días de trabajo por su parte.

Paralelo

- Si Ud. tuviera 100 trabajadores podría dividir la tarea y hacer que cada trabajador fabrique un par de zapatos.
- Entonces el tiempo del procesamiento (fabricación) del trabajo sería (en teoría) de solo un día.

¿Que es cómputo paralelo?

Serial

- Suponga que Ud. fabrica zapatos, y necesita fabricar un “lote” de 100 pares.
- Suponga que cada par le toma un día de trabajo.
- Esto requiere de 100 días de trabajo por su parte.

Paralelo

- Si Ud. tuviera 100 trabajadores podría dividir la tarea y hacer que cada trabajador fabrique un par de zapatos.
- Entonces el tiempo del procesamiento (fabricación) del trabajo sería (en teoría) de solo un día.

¿Que es cómputo paralelo?

Cómputo paralelo.

Es una forma de cómputo en donde varios cálculos se realizan de forma simultanea por diferentes procesos(procesadores). Es decir, se divide una tarea de cómputo en varias subtareas, de forma tal que cada subtarea es realizada por una unidad de cómputo diferente.

¿Que es cómputo paralelo?

Ejemplo: Realizar la siguiente operación: $e = a * b + c * d$, donde a, b, c, d, e son escalares. La operación serial podría llevarse a cabo de la siguiente forma:

Procesador único:
e=a*b;
e1=c*d;
e=e+e1;
2 procesadores:
*i*f (miprocesador==1)
e1=a*b;
if (miprocesador==2)
e2=c*d;

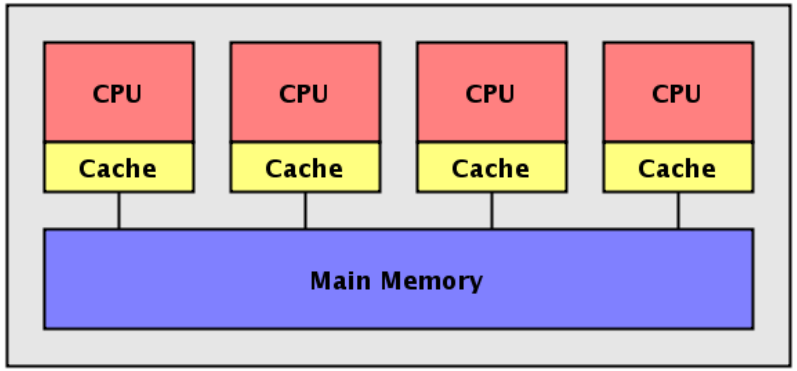
if (miprocesador==1)
e=e1+e2;

Si ahora a, b, c, d, e son matrices de 1000000×1000000. El beneficio de dividir la operación es bastante relevante.

Paradigmas de paralelización

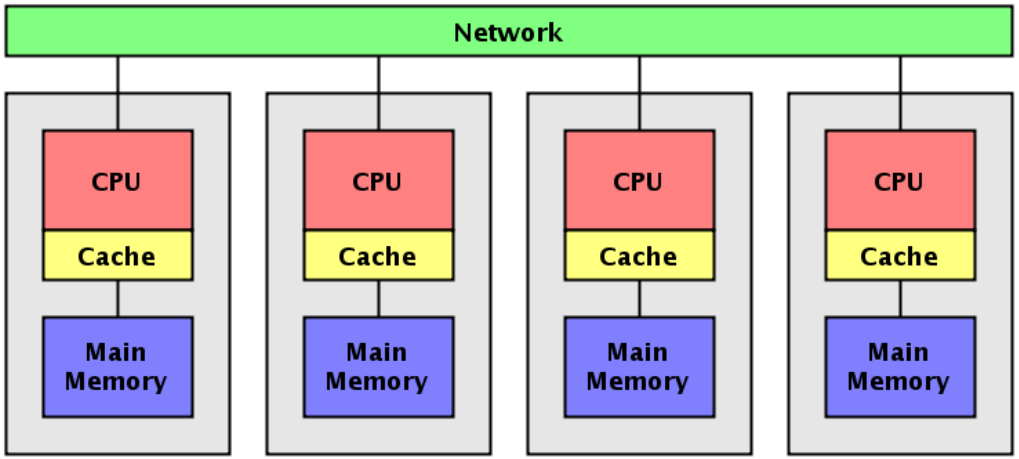
Con respecto a la memoria

Memoria Compartida



(Una computadora multicore)

Memoria Distirbuida



(Un cluster de computadoras)

Tipos de paralelización

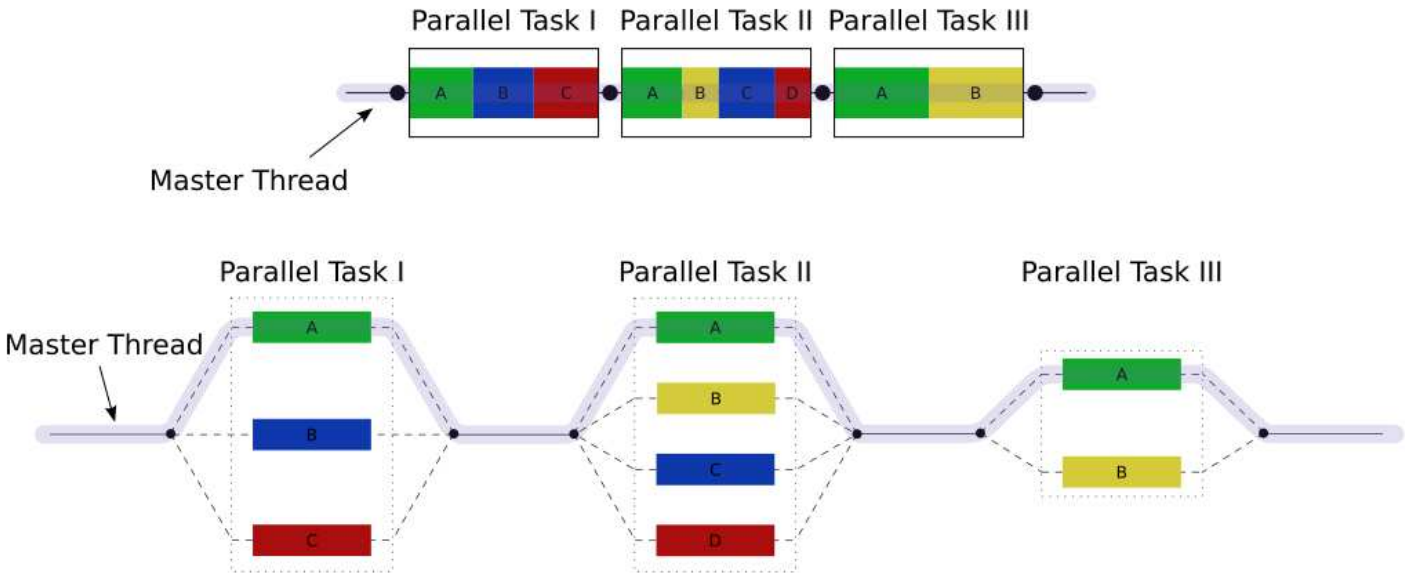
- Paralelización por tareas. Hacer diferentes tareas en diferentes procesadores.
- Paralelización por datos. Procesar cantidades similares de datos con operaciones similares en diferentes procesadores (dividir un vetors en conjuntos de elementos, o una matriz en conjuntos de renglones.)
- Otros tipos de paralelización: por bits y por instrucciones.

Tareas paralelizables

En general son conjuntos de operaciones en donde cada una se puede procesar sin necesidad de conocer el resultado de alguna otra.
En particular, ejemplos comunes son:
Ciclos **for** donde un indice no depende de otro.

OpenMP

OpenMP: Open Multiprocessing. Es un estándar para C/C++ y FORTRAN para realizar cómputo paralelo en memoria compartida utilizando multi-hilos.



Estructura de un programa en C

Programa normal

```
#include <stdio.h>


int main()
{
int n,identificador;

/ / Código serial aquí


}
```

Programa paralelo

```
#include <stdio.h>
#include <omp.h>

int main()
{
int n,identificador;

/ Código paralelo

#pragma omp parallel private(identificador)
}
```

Directivas de OpenMP

Inicia región paralela

```
#pragma omp parallel
```

Alcance de las variables

```
#pragma omp parallel private(var01,var02) shared(var03,var04)
```

Las variables: var01 y var02 son privadas, lo que quiere decir que se genera memoria para cada una de estas variables para cada proceso, cuando término la región paralela se regresa la memoria.

Las variables: var03 y var04, son variables compartidas que quiere decir que todos los procesos accesan a la misma memoria, que es la memoria que tenia el hilo principal.

Directivas OMP: región paralela

```
 //Inicio de región paralela
#pragma omp parallel private(identificador)
{


/* Obtiene el numero identificador de cada hilo*/
identificador = omp_get_thread_num();


/* Obtiene el número de hilos que se levantaron*/
nhilos = omp_get_num_threads();
} //fin de región paralela
```

Ejercicio 1

Hola Mundo Paralelo

Hacer que cada proceso diga que número de proceso es, e imprima: Hola Mundo!

Directivas OMP: for paralelo

```
 #pragma omp for schedule(static,chunk) nowait
for (i=0; i < N; i++)
```

chunk: Se procesarán “chunk” índices en cada proceso por vez.
static: quiere decir que se repartirá el trabajo de manera estática un chunk para cada proceso por vez, hasta que se término de procesar todos los datos.

Otra opción básica es: “dynamic” y “guided”

nowait: indica que no es necesario esperar a que todos los procesos terminen para continuar procesando.

Ejercicio 2

a) Hacer un for paralelo para la suma de dos vectores, mostrando en que hilo se procesa cada indice.

b) Comparar el tiempo de proceso serial vs paralelo de la suma de 2000 vectores de longitud 230000.

Secciones paralelas

```
 #pragma omp parallel shared(a,b,c,chunk)
private(i,j)
{
#pragma omp sections nowait
{
#pragma omp section
.//operaciones que realiza el primer proceso
#pragma omp section
.//operaciones que realiza el segundo proceso
} //fin de las secciones paralelas
}//fin de la región paralela
```

Ejercicio 3

Realizar una suma de vectores en dos secciones paralelas la mitad de los indices en una sección y la otra mitad en otra.

omp master

La directiva omp master indica que cierto trabajo solo será llevado a cabo por el proceso maestro, es util en operaciones criticas como lectura y escritura a disco.

La directiva omp barrier realiza una sincronización de los hilos esperando hasta que todos los hilos estén en el mismo segmento de ejecución.

```
 #pragma omp parallel shared(salida,a) private(i)
{
/*Sincronización de los hilos*/
#pragma omp barrier #pragma omp master
{
fprint(....);
}
}
```

Ejercicio

Realizar la suma de vectores en paralelo y imprimir el resultado en un archivo utilizando solo el nodo maestro.

omp single

```
 #pragma omp single nowait
{
for (i=0; i<N; i++)
fprintf(salida2,‘‘El hilo %d imprime %f
’’,identificador,a[i]);
}
```

La directiva single indica que solo un proceso realizará la parte indicada de código.

Ejercicio

Imprimir un vector en dos archivos diferentes, un proceso imprime al archivo 1 y otro al archivo 2, mostrar que elementos del vector imprimio cada proceso.

omp critical

Indica que cierta parte del código solo será ejecutada por un proceso a la vez.

```
#pragma omp critical
```

Ejercicio. omp critical

Imprimir un vector a un archivo utilizando todos los procesos y la directiva omp critical para que no impriman al mismo tiempo.

reduction

Si dos o más procesos trabajan sobre una variable la clausula reduction le aplica un operador a el valor que tenga la variable al final de la región paralela, el resultado de la operación es el valor con el que queda la variable al salir de la región paralela.

```
#pragma omp for reduction(+:resultado)
```

Reduction

Sumar los elementos de un vector en paralelo utilizando reduction para reducir la suma.

Ejercicios finales

Hacer un conjunto de funciones en paralelo que realicen las siguientes operaciones en paralelo:

1. Suma de dos vectores.
2. Suma de dos matrices.
3. Solución de un sistema de ecuaciones lineales diagonalmente dominante por el método de Jacobi/Seidel.
4. Integración numérica por el método del trapecio.
5. Filtrado del ruido de una imagen con un filtro de mediana en paralelo.

Comentarios finales

- OpenMP es un estándar para programación en paralelo en maquinas multicore o multiprocesador.
- La paralelización es relativamente sencilla en comparación con MPI por ejemplo.
- Aunque no sirve para memoria distribuida (un cluster), se pueden utilizar al mismo tiempo que MPI.
- El estándar está definido para FORTRAN y C pero hay extensiones para lenguajes de alto nivel como R y Python.