Programación con OpenMP

Prof. Manuel Almeida V.

UAEM CU Zumpango

23 de abril de 2020

Paralelismo en Hardware.

- Los sistemas SIMD son ideales para paralelizar simples loops, que operen sobre arrays de datos grandes.
- El paralelismo que es obtenido dividiendo los datos entre los procesadores y haciendo que todos los procesadores apliquen las mismas instrucciones a sus subconjuntos de datos, es llamado Paralelismo de Datos.
- El paralelismo SIMD puede ser muy eficiente sobre grandes problemas de datos paralelos.

Paralelismo en Hardware.

- Los sistemas SIMD son ideales para paralelizar simples loops, que operen sobre arrays de datos grandes.
- El paralelismo que es obtenido dividiendo los datos entre los procesadores y haciendo que todos los procesadores apliquen las mismas instrucciones a sus subconjuntos de datos, es llamado Paralelismo de Datos.
- El paralelismo SIMD puede ser muy eficiente sobre grandes problemas de datos paralelos.

Paralelismo en Hardware.

- Los sistemas SIMD son ideales para paralelizar simples loops, que operen sobre arrays de datos grandes.
- El paralelismo que es obtenido dividiendo los datos entre los procesadores y haciendo que todos los procesadores apliquen las mismas instrucciones a sus subconjuntos de datos, es llamado Paralelismo de Datos.
- El paralelismo SIMD puede ser muy eficiente sobre grandes problemas de datos paralelos.

Memoria Compartida.

- Shared-memory systems. Los sistemas más ampliamente disponibles son, lo sistemas de memoria compartida, que usa uno o más procesadores multicore.
- Un procesador multicore tiene multiples CPUs o cores en un solo chip.

Memoria Compartida.

- Shared-memory systems. Los sistemas más ampliamente disponibles son, lo sistemas de memoria compartida, que usa uno o más procesadores multicore.
- Un procesador multicore tiene multiples CPUs o cores en un solo chip.

Memoria Compartida.

- Típicamente, los núcleos tienen nivel 1 de caché, mientras otros cachés pueden o no ser ser compartidos entre los núcleos.

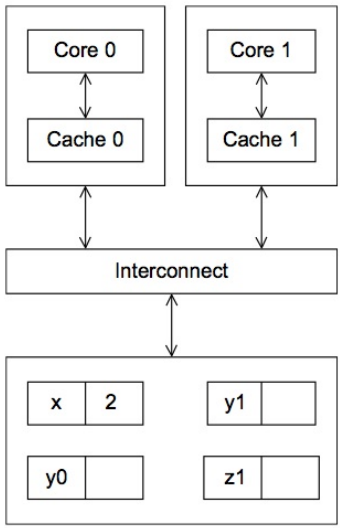


FIGURE 2.17

A shared-memory system with two cores and two caches

Memoria Compartida.

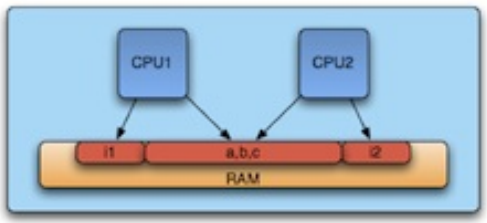
- En sistemas shared-memory con multiples procesadores, la interconexión puede comunicar todos los procesadores directamente a la memoria principal.
- O cada procesador puede tener una conexión directa a un block de memoria principal.

Threads have access to the same address space

- Communication is implicit

Programmer needs to define

- local data
- shared data



Qué es OpenMP?.

- OpenMP es una aplicación para interfaz de programación (API), enfocada a la programación con memoria compartida. No es un lenguaje, es un conjunto de:
  - Directivas del compilador, que le dicen que partes del programa ejecutar en paralelo, y cómo distribuirlas entre los hilos. (Pragmas en C y $C++$.)
  - Funciones que sirven para controlar la actuación de los hilos.
  - Variables de Ambiente

Cómo Funciona OpenMP.

- Un hilo es una entidad en tiempo de corrida de un programa, que es capaz de de ejecutar un stream de instrucciones. OpenMP construye un grupo de trabajo, que soporta las especificaciones de programas, para ser ejecutados por una colección de hilos que cooperan entre sí.

Cómo Funciona OpenMP.

- Si múltiples hilos colaboran para ejecutar un programa, compartirán los recursos incluyendo, el espacio de direcciones, del proceso correspondiente.
- Multithreads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- No se permiten ramificaciones de salida o hacía otro bloque de código.

Cómo Funciona OpenMP.

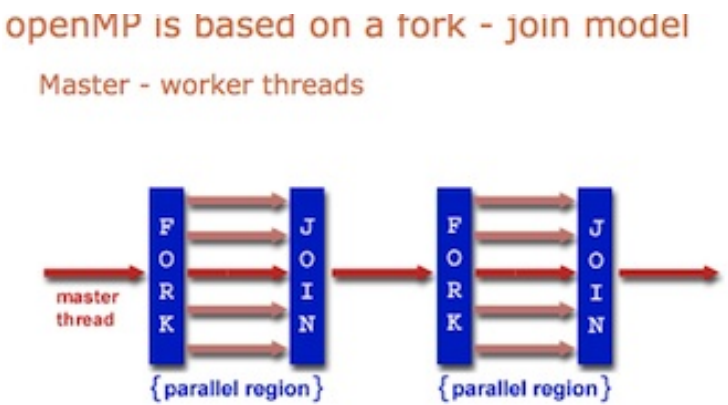
- Si múltiples hilos colaboran para ejecutar un programa, compartirán los recursos incluyendo, el espacio de direcciones, del proceso correspondiente.
- Multithreads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- No se permiten ramificaciones de salida o hacía otro bloque de código.

Cómo Funciona OpenMP.

- Si múltiples hilos colaboran para ejecutar un programa, compartirán los recursos incluyendo, el espacio de direcciones, del proceso correspondiente.
- Multithreads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- No se permiten ramificaciones de salida o hacía otro bloque de código.

Cómo Funciona OpenMP.

- Llamamos modelo fork-join al proceso que genera un hilo inicial, el cero, y un equipo de hilos es generado al principio de la sección en paralelo.



Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

- Crear equipos de de hilos para la ejecución en paralelo.
- Específicar cómo repartir el trabajo entre los miembros del equipo.
- Declarar variables compartidas y privadas.
- Sincronizar hilos y habilitarlos para ejecutar ciertas operaciones exclusivas.

Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

- Crear equipos de de hilos para la ejecución en paralelo.
- Específicar cómo repartir el trabajo entre los miembros del equipo.
- Declarar variables compartidas y privadas.
- Sincronizar hilos y habilitarlos para ejecutar ciertas operaciones exclusivas.

Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

- Crear equipos de de hilos para la ejecución en paralelo.
- Específicar cómo repartir el trabajo entre los miembros del equipo.
- Declarar variables compartidas y privadas.
- Sincronizar hilos y habilitarlos para ejecutar ciertas operaciones exclusivas.

Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

- Crear equipos de de hilos para la ejecución en paralelo.
- Específicar cómo repartir el trabajo entre los miembros del equipo.
- Declarar variables compartidas y privadas.
- Sincronizar hilos y habilitarlos para ejecutar ciertas operaciones exclusivas.

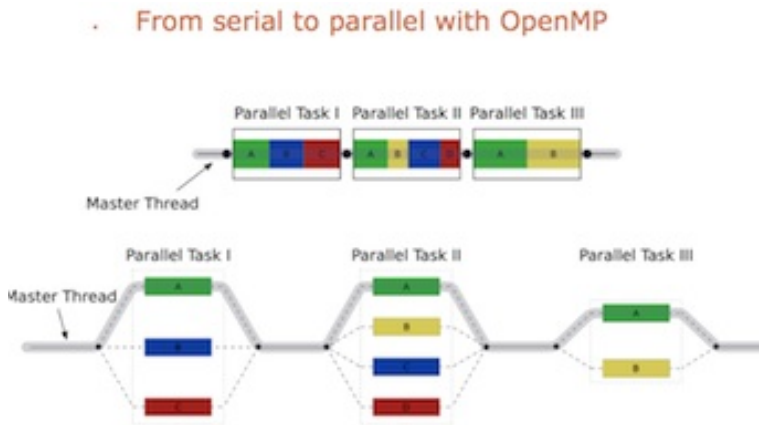
Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

- Crear equipos de de hilos para la ejecución en paralelo.
- Específicar cómo repartir el trabajo entre los miembros del equipo.
- Declarar variables compartidas y privadas.
- Sincronizar hilos y habilitarlos para ejecutar ciertas operaciones exclusivas.

Cómo construir un programa con OpenMP.

- 1er paso. Partiendo del programa secuencial, se deberá identificar cuánto paralelismo contiene. Esto significa, determinar el grupo de instrucciones que pueden ejecutarse simultáneamente por varios procesadores. Los programas donde un ciclo for es la parte principal, son los más adecuados.



Cómo construir un programa con OpenMP.

- 2o paso. Expresar usando las directivas de OpenMP, el paralelismo identificado.

Cómo construir un programa con OpenMP.

- Uno de los grandes beneficios de usar OpenMP; es el poder paralelizar, partiendo de un programa secuencial, en forma incremental.
- Podemos insertar las directivas donde se debe ejecutar en forma paralela y dejando el resto en forma secuencial.
- Una vez compilado y probado exitosamente, otra porción del código podrá ser paralelizado.

Cómo construir un programa con OpenMP.

- Uno de los grandes beneficios de usar OpenMP; es el poder paralelizar, partiendo de un programa secuencial, en forma incremental.
- Podemos insertar las directivas donde se debe ejecutar en forma paralela y dejando el resto en forma secuencial.
- Una vez compilado y probado exitosamente, otra porción del código podrá ser paralelizado.

Cómo construir un programa con OpenMP.

- Uno de los grandes beneficios de usar OpenMP; es el poder paralelizar, partiendo de un programa secuencial, en forma incremental.
- Podemos insertar las directivas donde se debe ejecutar en forma paralela y dejando el resto en forma secuencial.
- Una vez compilado y probado exitosamente, otra porción del código podrá ser paralelizado.

Un Programa con OpenMP.

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

using namespace std;

int main(int argc, char* argv[])
{
  double start = omp_get_wtime();
  if( !omp_in_parallel() )
  {
    printf("Number of processors is: %d\n", omp_get_num_procs());
    printf("Number of max threads is: %d\n", omp_get_max_threads());
  }
  sleep(1);
  double end = omp_get_wtime();
  printf("start = %.16g\nend = %.16g\ndiff = %.16g\n",
              start, end, end - start);
  return 0;
}
```

Suma Vectorial con OpenMP.

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

using namespace std;

int main(int argc, char **argv)
{
  int n = atoi(argv[1]);
  double *x, *y;

  x = new double [n]; for(int i=0; i<n; i++) x[i] = (double) (i+2);
  y = new double [n]; for(int i=0; i<n; i++) y[i] = (double) (i*3);

  double start = omp_get_wtime();
  #pragma omp parallel for
    for (int i=0; i<n; i++) x[i] = x[i] + y[i];
  double end = omp_get_wtime();
  printf("diff = %.16g\n", end - start);

  return 0;
}
```

Producto Punto con OpenMP.

```
#include < stdio.h >
#include < stdlib.h >
#include < omp.h >
double sum;
int main(int argc, char *argv[]){
double A[256]; B[256];
int i;
int n = 256;
for(i = 0; i < n; i++){
A[i] = i * 0,5;
B[i] = i * 2,0;}
sum = 0;
```

Producto Punto con OpenMP.

```
#pragma omp for reduction(+ : sum)
for(i = 0; i < n; i++){
sum = sum + A[i] * B[i]; }
printf("sum = %f", sum); }
```

Estructuras Paralelas con OpenMP

- Cuando un hilo alcanza una directiva PARALLEL, éste crea un equipo de hilos y queda como el hilo master del equipo. El master es un miembro del equipo y tiene el número cero dentro del equipo.
- Iniciando desde el principio de esta región paralela, el código se multiplica , y todos los hilos ejecutan ese código.
- Hay una barrera implícita al final de cada sección paralela. Solo el hilo master continua en ejecución a partir de este punto.

Estructuras Paralelas con OpenMP

- Cuando un hilo alcanza una directiva PARALLEL, éste crea un equipo de hilos y queda como el hilo master del equipo. El master es un miembro del equipo y tiene el número cero dentro del equipo.
- Iniciando desde el principio de esta región paralela, el código se multiplica , y todos los hilos ejecutan ese código.
- Hay una barrera implícita al final de cada sección paralela. Solo el hilo master continua en ejecución a partir de este punto.

Estructuras Paralelas con OpenMP

- Cuando un hilo alcanza una directiva PARALLEL, éste crea un equipo de hilos y queda como el hilo master del equipo. El master es un miembro del equipo y tiene el número cero dentro del equipo.
- Iniciando desde el principio de esta región paralela, el código se multiplica , y todos los hilos ejecutan ese código.
- Hay una barrera implícita al final de cada sección paralela. Solo el hilo master continua en ejecución a partir de este punto.

Referencias de OpenMP

Sintaxis
#pragma omp nombre – directiva [clausulas]
#pragma omp parallel [clausulas] block estructurado
#pragma omp for [clausulas] for loop
Estructuras Combinadas
#pragma omp parallel for [clausulas] for loop

Referencias de OpenMP

Funciones Run time	Descripción
void omp_set_num_threads(int)	Establece el número de hilos.
int omp_get_num_threads(void)	Regresa el número de hilos.
int omp_get_thread_num(void)	Regresa el identificador del hilo.
int omp_get_num_procs(void)	Regresa el número de procs.
Variables de Ambiente	Ejemplo
OMP_NUM_THREADS	8

Bibliografía y Referencias


M. J. Quinn. "Parallel Programming in C with MPI and OpenMPI", McGrawHill USA 2004.