

# Programación con OpenMP

Prof. Manuel Almeida V.

UAEM CU Zumpango

27 de abril de 2020



- 1 Introducción
  - Instrucciones de openMP
  - Cómo Funciona OpenMP
- 2 Cómo construir un programa con OpenMP
  - Un Programa con OpenMP
- 3 Estructuras Paralelas con OpenMP
- 4 Referencias de OpenMP
- 5 Práctica



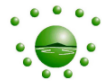
# Introducción.

- Vamos a comenzar a programar con openMP, como se mencionó en la presentación anterior, una de las ventajas es que nos quita el trabajo de programar a detalle cada hilo.
- Otra ventaja que se maneja, es que se puede paralelizar a partir del algoritmo secuencial, que ya está probado.
- Pero, créo lo más importante, que cuenta con estructuras más avanzadas para paralelizar problemas más complejos.



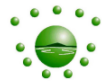
# Introducción.

- Vamos a comenzar a programar con openMP, como se mencionó en la presentación anterior, una de las ventajas es que nos quita el trabajo de programar a detalle cada hilo.
- Otra ventaja que se maneja, es que se puede paralelizar a partir del algoritmo secuencial, que ya está probado.
- Pero, créo lo más importante, que cuenta con estructuras más avanzadas para paralelizar problemas más complejos.



# Introducción.

- Vamos a comenzar a programar con openMP, como se mencionó en la presentación anterior, una de las ventajas es que nos quita el trabajo de programar a detalle cada hilo.
- Otra ventaja que se maneja, es que se puede paralelizar a partir del algoritmo secuencial, que ya está probado.
- Pero, créo lo más importante, que cuenta con estructuras más avanzadas para paralelizar problemas más complejos.

CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

# Instrucciones de openMP.

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



# Cómo Funciona OpenMP.

- Recordemos que un hilo es una entidad en tiempo de corrida de un programa, que es capaz de ejecutar un conjunto 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, esto es, comparten la memoria, del proceso correspondiente.
- Múltiples threads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- Algo atomar en cuenta, es que no se permiten ramificaciones de salida o hacia otro bloque de código.





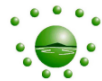
# Cómo Funciona OpenMP.

- Si múltiples hilos colaboran para ejecutar un programa, compartirán los recursos incluyendo, el espacio de direcciones, esto es, comparten la memoria, del proceso correspondiente.
- Múltiples threads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- Algo atomar en cuenta, es que no se permiten ramificaciones de salida o hacia otro bloque de código.



# Cómo Funciona OpenMP.

- Si múltiples hilos colaboran para ejecutar un programa, compartirán los recursos incluyendo, el espacio de direcciones, esto es, comparten la memoria, del proceso correspondiente.
- Múltiples threads corriendo simultáneamente en múltiples procesadores o núcleos pueden trabajar concurrentemente para ejecutar un programa paralelo.
- Algo atomar en cuenta, es que no se permiten ramificaciones de salida o hacia otro bloque de código.



# Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

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



CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

# Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

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



CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

# Cómo Funciona OpenMP.

OpenMP provee al usuario de los medios para

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



CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

# 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 en varios procesadores. Los programas donde un ciclo for, es la parte principal, son los más adecuados, y fáciles de paralelizar.





# Cómo construir un programa con OpenMP.

- **2o paso.** Usando las directivas de OpenMP, expresar 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.

- Veamos el producto de vector por vector.

En la parte de preparación, requerimos a librería `omp.h`

```
Pasar parametros arch vectors tamaño*/  
#include<iostream>  
#include<fstream>  
#include<cstdio>  
#include<cstdlib>  
#include<omp.h>  
#include<math.h>  
using namespace std;  
float* lee_vec(char *,int );  
float * crea_vec(int );
```



CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

# Un Programa con OpenMP.

- En la función `main`, establecemos cuántos hilos queremos.

```
int main(int argc, char * argv[]){  
    int i,n;  
    float prod=0.0;  
    float *v1,*v2;  
    omp_set_num_threads(8);  
    n=atoi(argv[3]);  
    v1=new float[n];  
    v2=new float[n];  
    v1=lee_vec(argv[1],n);  
    v2=lee_vec(argv[2],n);
```



# Un Programa con OpenMP.

- Paralelizamos usando el **pragma omp for**, además se incluye la instrucción **reduce**, que acumula el resultado de los ocho productos parciales, en la variable **prod**, con **(+:prod)**.

```
#pragma omp parallel for reduction(+:prod)
    for(i=0;i<n;i++)
    {prod += v1[i]*v2[i];}
cout<<"El producto es igual a: "<<prod<<endl;
return 0;
}
```



# Un Programa con OpenMP.

- Podemos ver en código anterior, cómo a partir del programa secuencial de vector por vector, al agregarle tres líneas, la librería, la instrucción de declaración de hilos a funcionar y la directiva pragma; hemos paralelizado de forma simple rápida.
- Obviamente no todos los programas se pueden paralelizar, tan fácilmente.  
Por esto necesitamos otras directivas.





# Un Programa con OpenMP.

- Podemos ver en código anterior, cómo a partir del programa secuencial de vector por vector, al agregarle tres líneas, la librería, la instrucción de declaración de hilos a funcionar y la directiva pragma; hemos paralelizado de forma simple rápida.
- Obviamente no todos los programas se pueden paralelizar, tan fácilmente.  
Por esto necesitamos otras directivas.



# 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
<code>void omp_set_num_threads( int )</code>	Establece el número de hilos.
<code>int omp_get_num_threads( void )</code>	Regresa el número de hilos.
<code>int omp_get_thread_num( void )</code>	Regresa el identificador del hilo.
<code>int omp_get_num_procs( void )</code>	Regresa el número de procs.
Variables de Ambiente	Ejemplo
<code>OMP_NUM_THREADS</code>	8



## Práctica.

- Para lograr cierto dominio de la programación con openmp, hay que trabajar varios problemas, que les son conocidos; por ejemplo, matriz por vector, matriz por matriz.
- Pero para ir más adelante, es necesario enfrentarse con algoritmos no tan simples. A continuación les doy una lista de programas a paralelizar.



## Práctica.

- Para lograr cierto dominio de la programación con openmp, hay que trabajar varios problemas, que les son conocidos; por ejemplo, matriz por vector, matriz por matriz.
- Pero para ir más adelante, es necesario enfrentarse con algoritmos no tan simples. A continuación les doy una lista de programas a paralelizar.





# Práctica.

- K-means
- Algoritmo evolutivo para optimizar funciones de varias variables.
- Una red neuronal, simple.
- Regresión lineal para varias variables.
- Método de gradiente descendente, para varias variables.



# Práctica.

- K-means
- Algoritmo evolutivo para optimizar funciones de varias variables.
- Una red neuronal, simple.
- Regresión lineal para varias variables.
- Método de gradiente descendente, para varias variables.



# Práctica.

- K-means
- Algoritmo evolutivo para optimizar funciones de varias variables.
- Una red neuronal, simple.
- Regresión lineal para varias variables.
- Método de gradiente descendente, para varias variables.



# Práctica.

- K-means
- Algoritmo evolutivo para optimizar funciones de varias variables.
- Una red neuronal, simple.
- Regresión lineal para varias variables.
- Método de gradiente descendente, para varias variables.



# Práctica.

- K-means
- Algoritmo evolutivo para optimizar funciones de varias variables.
- Una red neuronal, simple.
- Regresión lineal para varias variables.
- Método de gradiente descendente, para varias variables.



## Práctica.

- Podemos poner otros, como el convex hull.
- Así que vayan escogiendo combinaciones de tres de los problemas listados.
- Lo haremos primero con openmp, después con python.



CENTRO UNIVERSITARIO UAEM  
Z U M P A N G O

## Práctica.

- Podemos poner otros, como el convex hull.
- Así que vayan escogiendo combinaciones de tres de los problemas listados.
- Lo haremos primero con openmp, después con python.





## Práctica.

- Podemos poner otros, como el convex hull.
- Así que vayan escogiendo combinaciones de tres de los problemas listados.
- Lo haremos primero con openmp, después con python.





## Bibliografía y Referencias

-  Barbara Chapmann , R. van Der Pas.  
"Using OpenMP", CRC USA 2010.
-  M. J. Quinn.  
"Parallel Programming in C with MPI and OpenMPI",  
McGrawHill USA 2004.