Programación Híbrida e Introducción a la Programación de GPUs

Fernando Robles Morales

Instituto Nacional de Medicina Genómica

Enrique Cruz Martínez

Universidad Autónoma de la ciudad de México

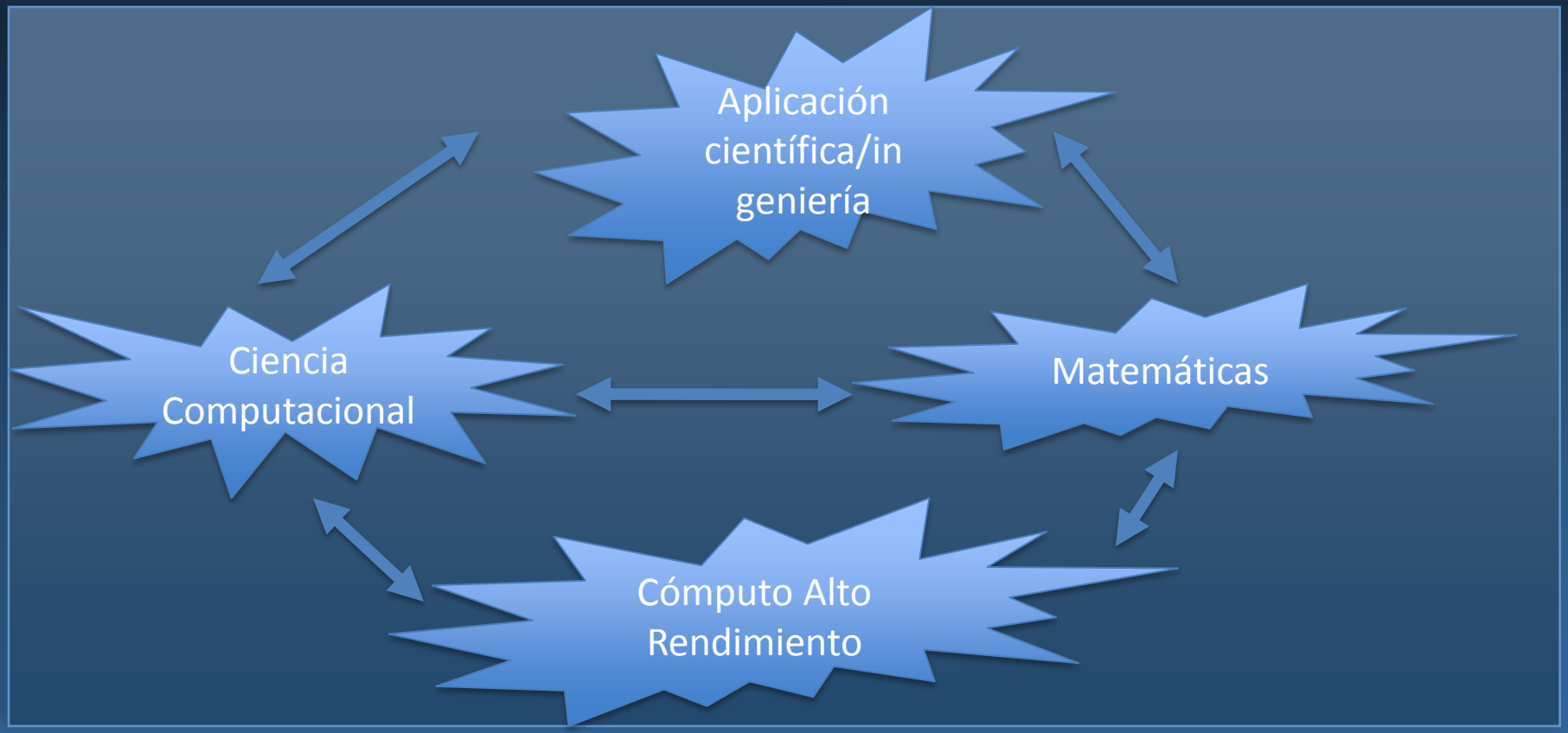
CONTENIDO

1. **Introducción al Cómputo de Alto Rendimiento.**
    a. **Sistemas de Memoria Compartida.**
    b. **Sistemas de Memoria Distribuida.**
2. **Paralelismo.**
    a. **Importancia.**
    b. **Procesos paralelos y distribuidos.**
3. **Programacion Paralela.**
4. **Programación de GPUs con OpenACC.**
5. **Evaluación del Rendimiento en Aplicaciones Científicas.**

ANTECEDENTES

- **Paradigma tradicional en las ciencias e ingeniería:**
  - ✓ **Hacer teoría o diseños en papel.**
  - ✓ **Realizar experimentos.**
- **Paradigma de ciencias computacionales:**
  - ✓ **Simular el fenómeno con base en las leyes físicas y métodos numéricos eficientes.**

CIENCIA COMPUTACIONAL



Instituto Nacional de Medicina Genómica
MEXICO

COMPUTO DE ALTO RENDIMIENTO

- **Formulación de un problema para su tratamiento computacional.**
- **Representación del dominio de un problema para su procesamiento.**
- **Arquitecturas de computadoras que proporcionan un buen rendimiento.**
- **Algoritmos que proporcionen la mejor aproximación y menor complejidad.**
- **Herramientas de software existentes que proporcionen mejores espectativas para solucionar problemas.**

Instituto Nacional de Medicina Genómica
MEXICO

PROBLEMAS DEL GRAN RETO

- **Escencial para descubrimientos científicos.**
- **Críticos para seguridad nacional.**
- **Contribuye fundamentalmente para la economía y competitividad a través de su uso en la ingeniería y en la manufactura.**
- **Las computadoras de alto rendimiento son las herramientas que resuelven estos problemas a través de simulaciones.**

Instituto Nacional de Medicina Genómica
MEXICO

COMPUTADORA DE ALTO RENDIMIENTO

- **Paralelismo en diferentes niveles (hardware y software).**
- **Dos o más niveles de memoria en función de sus tiempos de acceso, denominado memoria jerárquica.**
- **Interconexión de alta velocidad entre procesadores ó máquinas.**
- **Subsistemas especializados de entrada/salida.**
- **Uso de software especializado en tales arquitecturas (Sistema Operativo, Herramientas de Análisis, Compiladores, etc.).**

Instituto Nacional de Medicina Genómica
MEXICO

CLASIFICACIÓN DE COMPUTADORAS

- **Single Instruction, Single Data (PCs).**
- **Single Instruction, Multiple Data (Computadoras Vectoriales).**
- **Multiple Instruction, Single Data (Super pipeline).**
- **Multiple Instruction, Multiple Data (Computadoras Masivamente Paralelas, Clusters, Grids).**

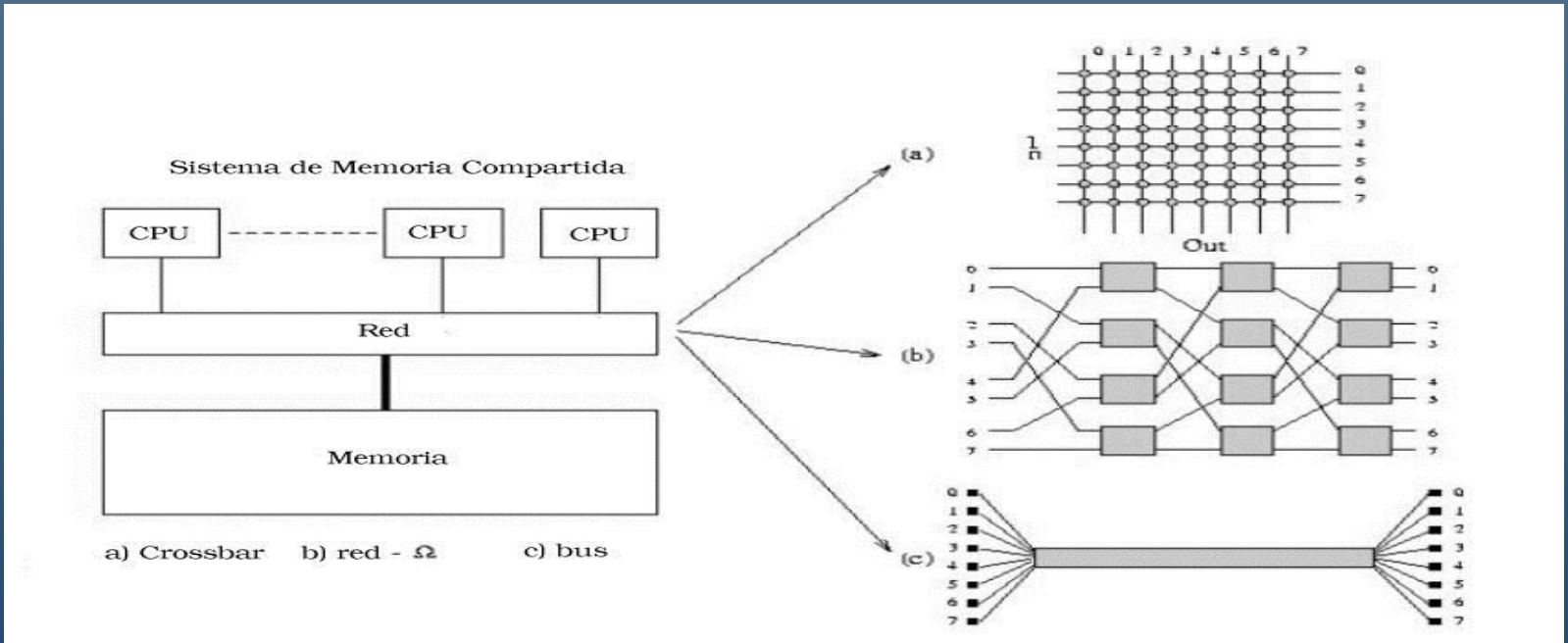
Instituto Nacional de Medicina Genómica
MEXICO

SISTEMAS DE MEMORIA COMPARTIDA

- **Ventajas:**
  - **Multiples procesadores que comparten el mismo espacio de direcciones de la memoria principal.**
  - **Evita pendientes en cuanto a la ubicación de los datos en la memoria principal.**
- **Desventajas:**
  - **Escalabilidad limitada en el número de procesadores, costo en interconexión de procesadores crece en $O(n^2)$ para un crecimiento de $O(n)$ procesadores.**

Instituto Nacional de Medicina Genómica
MEXICO

SISTEMAS DE MEMORIA COMPARTIDA



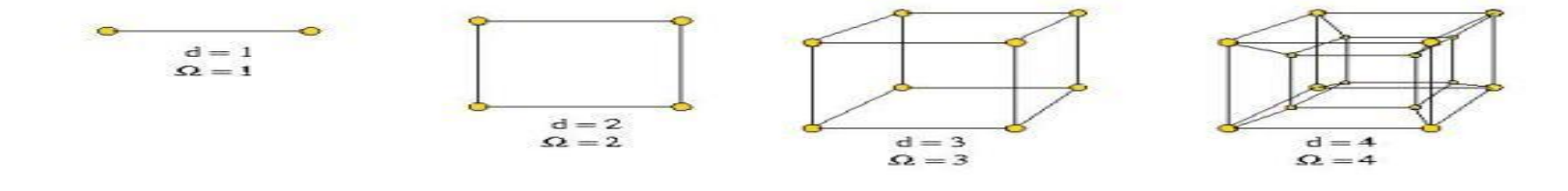
Instituto Nacional de Medicina Genómica
MEXICO

SISTEMAS DE MEMORIA DISTRIBUIDA

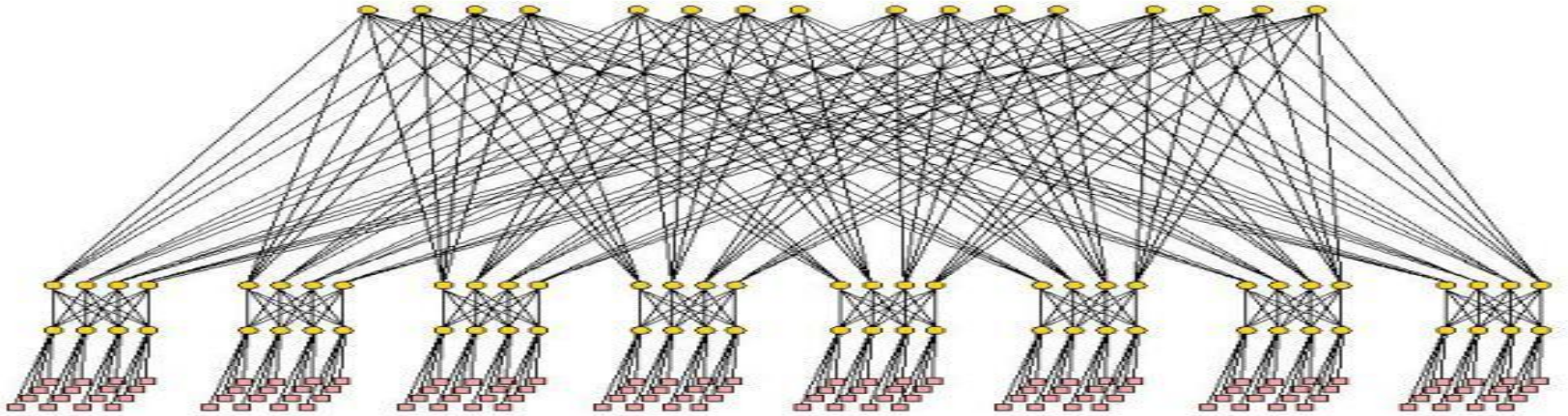
- **Ventajas:**
  - **Ancho de banda escalable en proporción al número de procesadores.**
- **Desventajas:**
  - **Comunicación entre procesadores más lenta que en un sistema de memoria compartida.**

Instituto Nacional de Medicina Genómica
MEXICO

SISTEMAS DE MEMORIA DISTRIBUIDA



a) Hipercubo de dimensiones 1 a 4



b) Red de arbol para 128 nodos

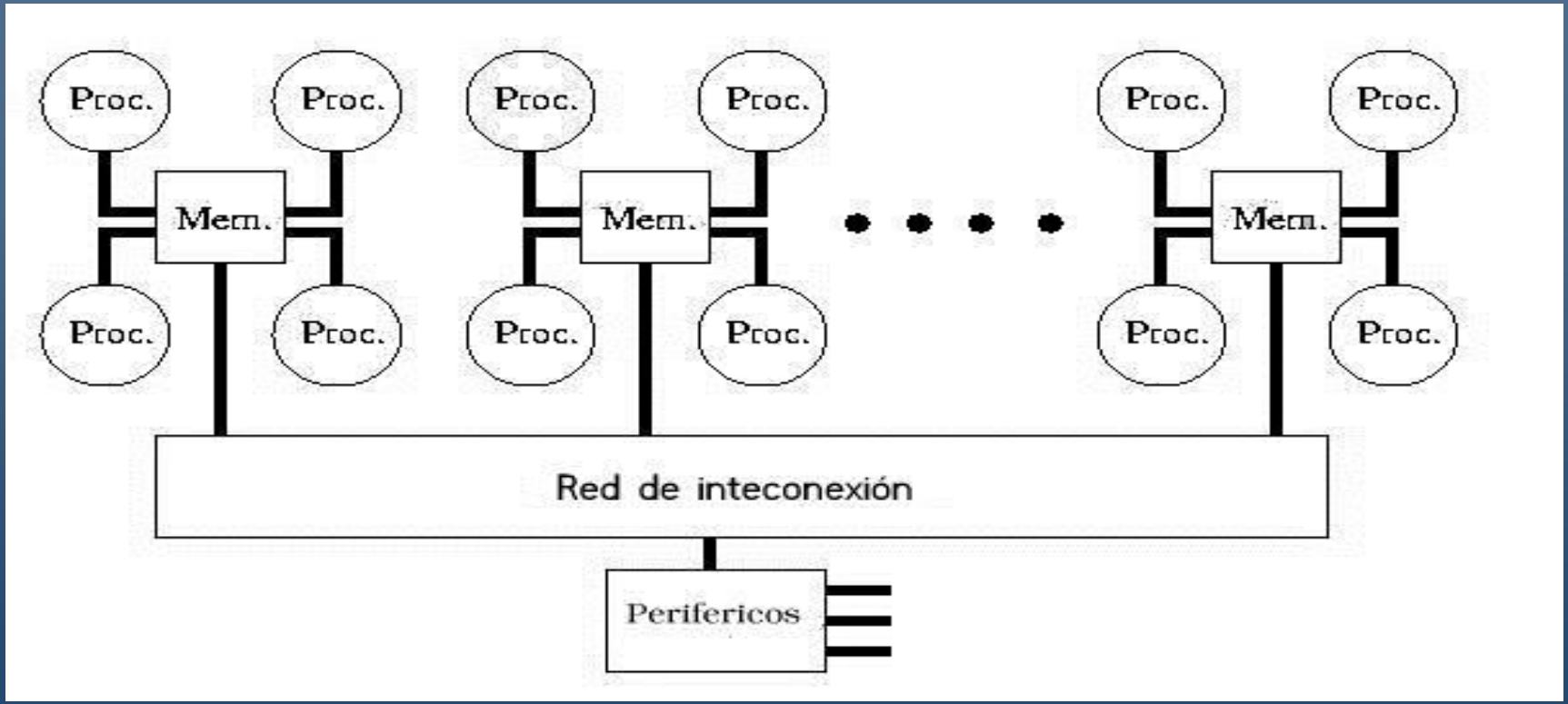
Instituto Nacional de Medicina Genómica
MEXICO

CLUSTERS SMP

- **Sistema de Multiprocesamiento Simetrico (SMP)**
  - **Bloques de procesamiento multicore (16 cores o más).**
  - **Interconexión de bajo costo entre bloques de procesamiento.**
  - **Alcanzan el rendimiento de una computadora de memoria compartida de mayor costo.**

Instituto Nacional de Medicina Genómica
MEXICO

CLUSTERS SMP



Instituto Nacional de Medicina Genómica
MEXICO

CONTENIDO

1. Introducción al Cómputo de Alto Rendimiento.
   a. Sistemas de Memoria Compartida.
   b. Sistemas de Memoria Distribuida.
2. Paralelismo.
   a. Importancia.
   b. Procesos paralelos y distribuidos.
3. Programacion Paralela.
4. Programación de GPUs con OpenACC.
5. Evaluación del Rendimiento en Aplicaciones Científicas.

Instituto Nacional de Medicina Genómica
MEXICO

IMPORTANCIA DEL PARALELISMO

- **El mundo real es inherentemente paralelo, la naturaleza realiza sus procesos en forma paralela o de manera que no impida el paralelismo.**
- **El paralelismo dispone de un rendimiento mayor al realizado por un solo procesador.**
- **Cada generación de procesadores aumenta en un 10% su rendimiento en comparación con su versión anterior. En consecuencia varios sistemas interconectados con procesadores de generaciones previas, se pueden obtener rendimientos aceptables.**

Instituto Nacional de Medicina Genómica
MEXICO

DIFERENCIAS PROCESOS DISTRIBUIDOS Y PARALELOS

- **Tiene un número fijo de procesos concurrentes que inician simultáneamente y permanecen mientras la computadora este en funcionamiento. Un proceso puede acceder solamente a sus propias variables, no hay variables en común para otros procesos.**
- **Permitir que los procesos ejecuten procedimientos en común.**
- **Permitir la sincronización de procesos por medio de sentencias no deterministas, llamadas regiones críticas.**

Instituto Nacional de Medicina Genómica
MEXICO

CONTENIDO

1. Introducción al Cómputo de Alto Rendimiento.
   a. Sistemas de Memoria Compartida.
   b. Sistemas de Memoria Distribuida.
2. Paralelismo.
   a. Importancia.
   b. Procesos paralelos y distribuidos.
3. Programacion Paralela.
4. Programación de GPUs con OpenACC.
5. Evaluación del Rendimiento en Aplicaciones Científicas.

Instituto Nacional de Medicina Genómica
MEXICO

COMUNICACIÓN ENTRE PROCESOS

- **Envio de mensajes: Un proceso envía el mensaje por medio de paquetes con una cabecera indicando el proceso receptor, el proceso destino y los datos que enviará.**
- **Transferencias a través de la memoria compartida: La comunicación entre procesos en un arquitectura de memoria compartida consiste en asignar o tomar valores de ciertas localidades de memoria, a esta comunicación se le denomina directa.**
- **Acceso directo a memoria: La comunicación entre procesos en una maquina SMP, que cuenta con uno o más procesadores para las aplicaciones y para atender las peticioes de la red. De este modo un mensaje lo trata como un acceso a memoria de otro procesador.**

Instituto Nacional de Medicina Genómica
MEXICO

ESTRATEGIA PARA EL DESARROLLO DE APLICACIONES

- **Paralelización Automática: Consiste en relevar al programador de las tareas de paralelización. Por ejemplo, un compilador puede aceptar un código obsoleto (dusty-check) y producir un código objeto paralelo y eficiente sin (o con muy poco) trabajo adicional por parte del programador.**
- **Portabilidad del código paralelo: Consiste en el uso de bibliotecas paralelas. La idea básica es encapsular el código paralelo que sea común a varias aplicaciones en una biblioteca paralela, para que este pueda implementarse eficientemente y reutilizar su código.**
- **Desarrollo de una aplicación paralela: Consiste en escribir una aplicación paralela desde el principio, lo que otorga mayor grado de libertad al programador para que escoga el lenguaje y su paradigma. Sin embargo, su nivel de portabilidad es limitado.**

Instituto Nacional de Medicina Genómica
MEXICO

GRANULARIDAD

- **Idendificar pedazos de código a paralelizar:**
  - **Granos muy finos (hardware pipeline)**
  - **Granos finos (repartir datos o bloques de datos a los procesadores)**
  - **Granos medios (nivel de control escencialmente en ciclos)**
  - **Granos grandes (subrutinas y procedimientos)**

Instituto Nacional de Medicina Genómica
MEXICO

DISEÑO DE ALGORITMOS PARALELOS

- **Proceso de diseño en 4 pasos:**
  - **Particionamiento: Descomposicion de datos y funciones relacionadas con el problema en varios subproblemas.**
  - **Comunicación: Determinar el flujo de datos y la coordinación de los subproblemas creados en el particionamiento.**
  - **Aglomeración: Evaluar los suproblemas y su patrón de comunicación en términos de rendimiento y costo de implementación.**
  - **Mapeo: Asignar cada subproblema a un procesador para maximizar los recursos del sistema y minimizar los costos de comunicación.**

Instituto Nacional de Medicina Genómica
MEXICO

PARADIGMAS DE PROGRAMACIÓN PARALELA

- **Metodologías de alto nivel para algoritmos eficientes:**
  - **Redes de procesos (Proceso maestro, proceso esclavo)**
  - **Proceso con múltiples datos (SPMD, descomposición de dominio)**
  - **Línea de ensamble de datos (Pipeline de datos, granularidad fina)**
  - **Divide y vencerás (división problema, combina resultados parciales)**
  - **Paralelismo especulativo (empleo de diferentes algoritmos para mismo problema)**
  - **Modelos híbridos (mezcla de paradigmas para utilizarse en aplicaciones masivamente paralelas).**

Instituto Nacional de Medicina Genómica
MEXICO

BIBLIOTECAS DE PROGRAMACION PARALELA

- **MPI (Message Passing Interface):**
  - **Estándar en la comunicación de procesos, con base en envío de mensajes por más de 40 organizaciones de estados unidos y europa.**
  - **Uso común en computadoras de alto rendimiento con memoria distribuida.**
  - **Versión actual 2.0 http://www.mcs.anl.gov/research/projects/mpi/**

- **OpenMP (Multiprocesamiento en Memoria Compartida):**
  - **Estándar de paralelización automática avalado por los fabricantes de maquinas SMP ó de memoria compartida.**
  - **Uso común en computadoras de alto rendimiento con memoria compartida.**
  - **Versión actual 4.0 http://openmp.org**

Instituto Nacional de Medicina Genómica
MEXICO

MPI

- **Ejemplo:**

```
#include<stdio.h>
#include<mpi.h >
int main ( int argc , char  argv [ ] )
{
int rank , nproc , nombre_len ;
char nombre_proc [MPI_MAX_PROCESSOR_NAME] ;
MPI_Init(&argc ,&argv ) ;
MPI_Comm_size( MPI_COMM_WORLD ,&nproc ) ;
MPI_Comm_rank( MPI_COMM_WORLD ,&rank ) ;
MPI_Get_processor_name ( nombre_proc ,&nombre_len ) ;
printf( "Hola  proceso   %d de  %d en  % s  \n" , rank , nproc , nombre_proc ) ;
MPI_Finalize ( ) ;
return 0 ;
}
```

Instituto Nacional de Medicina Genómica
MEXICO

MPI

- **Compilación:**

  mpicc -o hola_mpi hola_mpi.c

- **Ejecución:**

  mpirun -np N hola_mpi

  N -> numero de procesos

Instituto Nacional de Medicina Genómica
MEXICO

MPI

- **Funciones mínimas:**
  - **MPI_Init: inicia el ambiente paralelo para el envio de mensajes.**
  - **MPI_Comm_size: establece el numero de procesos a utilizar determinado por el usuario al ejecutar mpirun.**
  - **MPI_Comm_rank: Se especifica el rango con base en el numero de procesos, para enumerarlos e identificarlos durante el envío de mensajes.**
  - **MPI_Send: Envia mensaje a un proceso, puede ir etiquetado y con un tipo de dato definido por el usuario.**
  - **MPI_Recv: Recibe mensajes de un proceso, este es asignado a las variables definidas por el usuario.**
  - **MPI_Finalize: termina el ambiente paralelo para el envio de mensajes.**

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP

- **Ejemplo:**

```
#include<omp.h>
#include <stdio.h>
#include<stdlib. h>
int main ( int argc , char  argv [ ] )
{
int nthreads , tid ;
/* Crear un grupo de thread con sus propias copias de variables*/
#pragma omp parallel private( nthreads , tid )
{
/*Obtener numero del thread*/
tid = omp_get_thread_num ( ) ;
printf ( ”Hola  Mundo desde  thread  =  %d \n" , tid ) ;
/* Proción de Código solo para el thread maestro 0 */
if( tid == 0)
{
nthreads = omp_get_num_threads ( ) ;
printf ( ”Numero de   threads  =  %d\n" , nthreads ) ;
}
} /*Todos los hilos se unen al hilo maestro y finaliza ambiente paralelo */
}
```

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP

- **Compilación:**

  gcc -o omp_hola -fopenmp omp_hola . c

- **Ejecución:**

  export OMP_NUM_THREADS=N

  N -> numero de procesos

  ./omp_hola

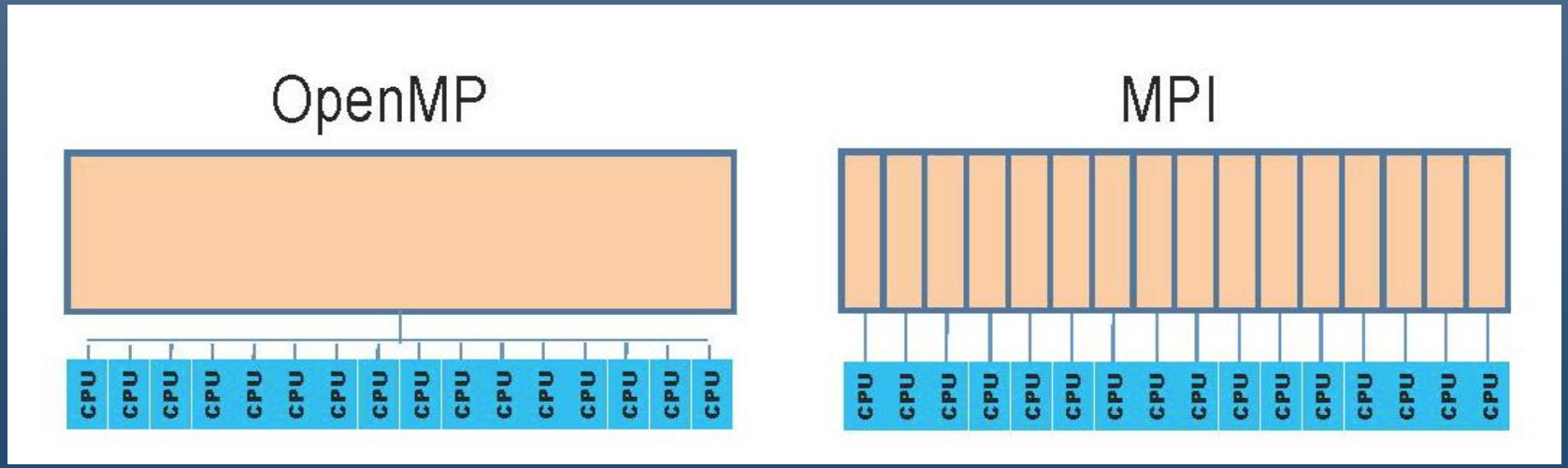
Instituto Nacional de Medicina Genómica
MEXICO

OpenMP

- **Funciones Mínimas:**
  - **#pragma** omp parallel private(var1,..,varN): inicia región paralela con variables copia para los hilos de ejecución.
  - **#pragma** omp parallel shared(var1,…,varN): inicia región paralela con variables compartidas para los hilos de ejecución.
  - **#pragma** omp parallel private(var1,…,varN) shared(var1,…,varN): inicia region paralela con variables copia y compartidas para los hilos de ejecución.

Instituto Nacional de Medicina Genómica
MEXICO

Vista a Nodos de Procesamiento en OpenMP y MPI



Instituto Nacional de Medicina Genómica
MEXICO

Modelo de Programación Híbrida

- Iniciar con el ambiente MPI
- Crear regiones paralelas con OpenMP en los procesos MPI
  - La región serial es el proceso MPI.
  - EL MPI_rank debe ser conocido por todos los hilos.
- Realizar llamadas de envio de mensajes con MPI en regiones seriales y paralelas.
- Finalizar MPI.

Instituto Nacional de Medicina Genómica
MEXICO

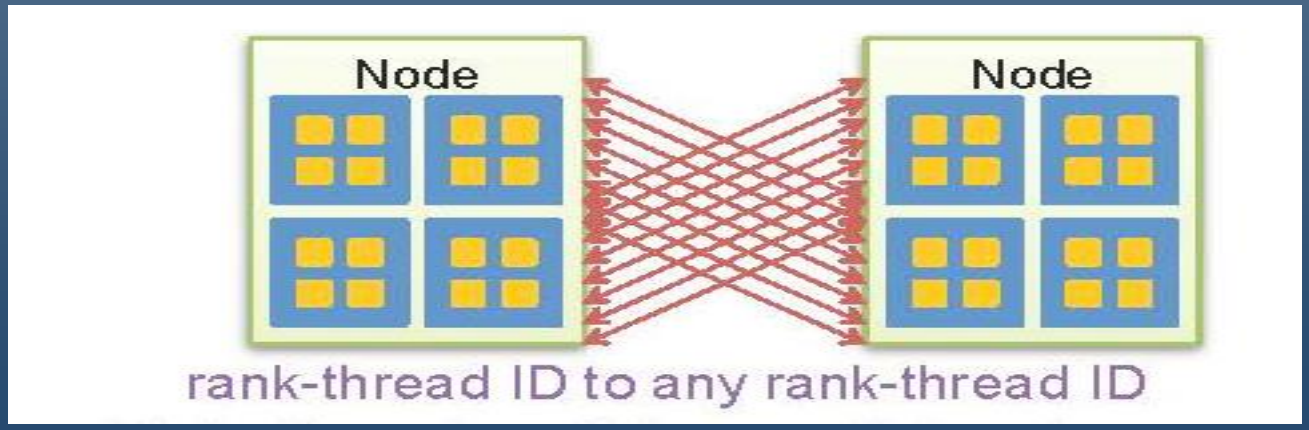
Modelos MPI con OpenMP

- **Paso de mensajes entre procesos MPI.**

Instituto Nacional de Medicina Genómica
MEXICO

Modelos MPI con OpenMP

- **Paso de mensajes entre hilos OpenMP.**



Instituto Nacional de Medicina Genómica
MEXICO

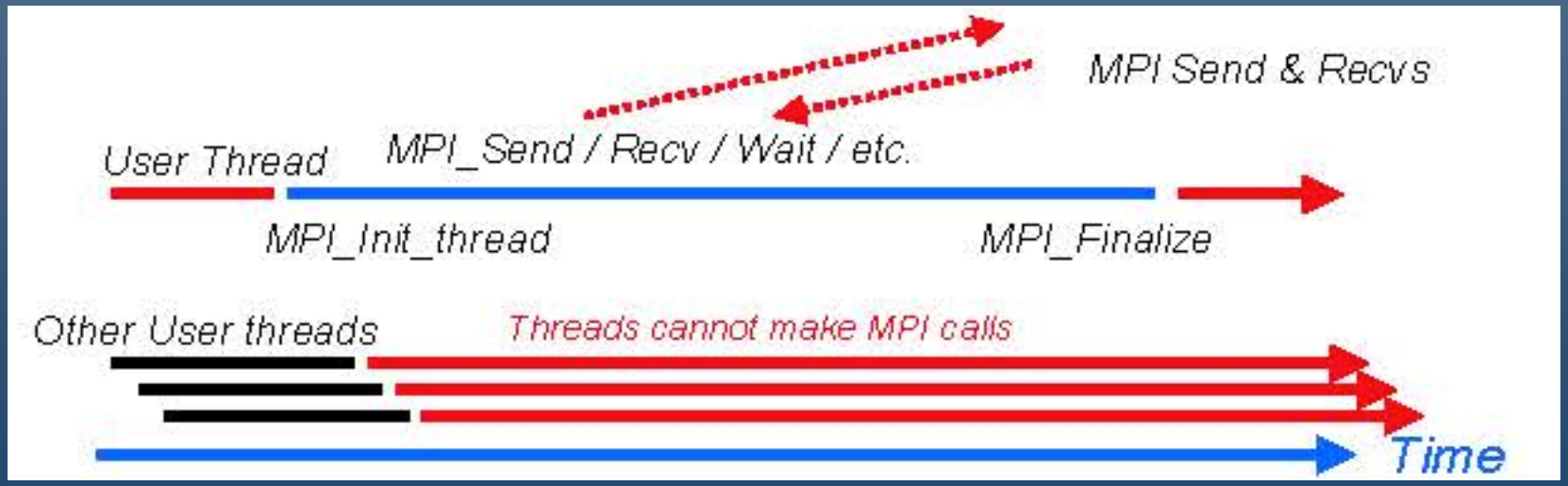
Thread Safe MPI con OpenMP

- **MPI_Init_thread: Determina el nivel de seguridad del thread:**
  - **Single: no hay multithreading.**
  - **Funneled: solo el hilo maestro puede realizar llamadas a MPI.**
  - **Serialized: cualquier hilo puede realizar envio de mensajes pero sólo uno a la vez se puede comunicar durante la ejecución.**
  - **Multiple: no hay restricciones, cualquier hilo puede realizar envio de mensajes.**

Instituto Nacional de Medicina Genómica
MEXICO

Thread Safe MPI con OpenMP

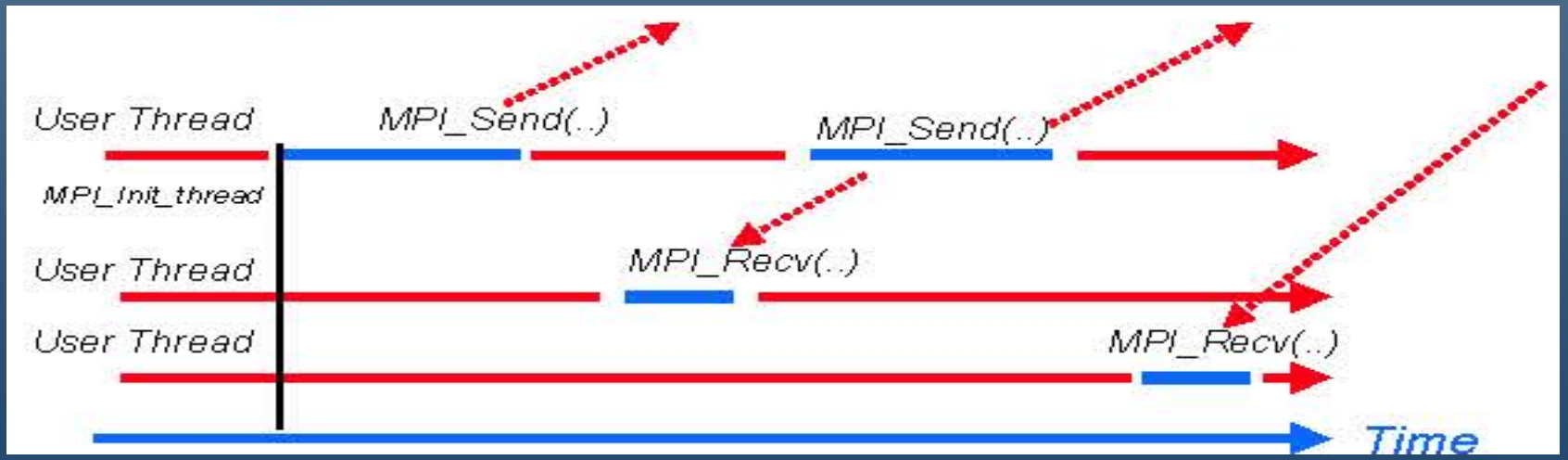
- **Ejemplo en Funneled:**



Instituto Nacional de Medicina Genómica
MEXICO

Thread Safe MPI con OpenMP

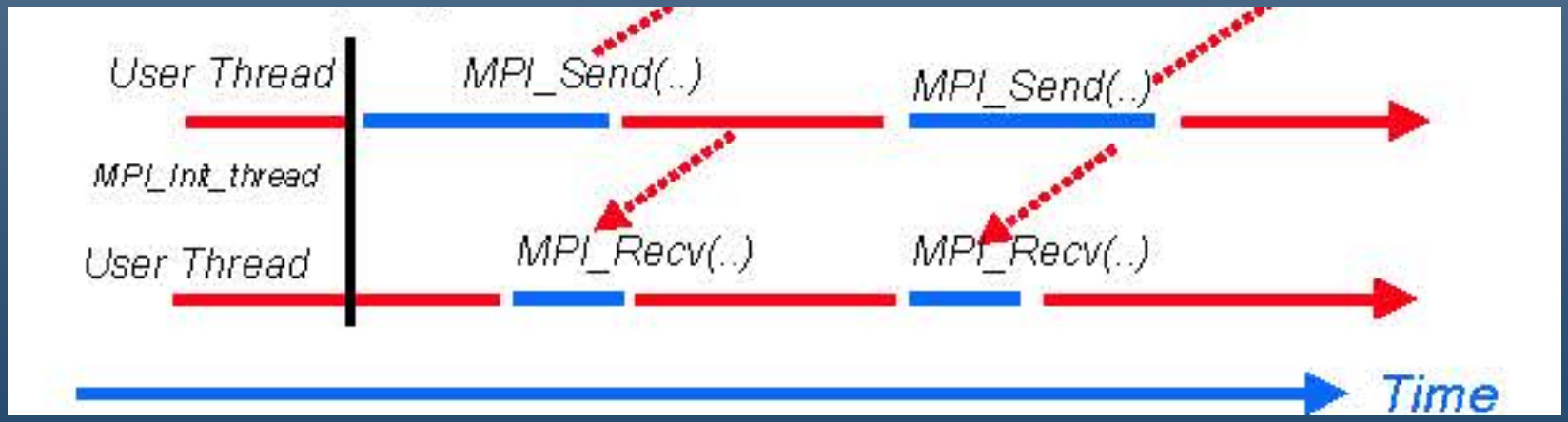
- **Ejemplo en Serialized:**



Instituto Nacional de Medicina Genómica
MEXICO

Thread Safe MPI con OpenMP

- **Ejemplo en Multiple:**



Instituto Nacional de Medicina Genómica
MEXICO

Thread Safe MPI con OpenMP

- **Ejemplo con Funneled (checar con SINGLE,MULTIPLE):**

```
#include <mpi.h>
#include <omp.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
int rank,omp_rank,mpisupport;
MPI_Init_thread(&argc,&argv,MPI_THREAD_FUNNELED, &mpisupport);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
omp_set_num_threads(atoi(argv[1]));
#pragma omp parallel private(omp_rank)
{
omp_rank=omp_get_thread_num();
printf("%d %d \n",rank,omp_rank);
}
MPI_Finalize();
}
```

Instituto Nacional de Medicina Genómica
MEXICO

CONTENIDO

1. Introducción al Cómputo de Alto Rendimiento.
   a. Sistemas de Memoria Compartida.
   b. Sistemas de Memoria Distribuida.
2. Paralelismo.
   a. Importancia.
   b. Procesos paralelos y distribuidos.
3. Programacion Paralela.
4. Programación de GPUs con OpenACC.
5. Evaluación del Rendimiento en Aplicaciones Científicas.

Instituto Nacional de Medicina Genómica
MEXICO

Programación Paralela en GPUs

- **Alto grado de paralelismo**
  - **Gran numero de threads escalares**
  - **Organización en bloques/grupos de threads.**
    - **SIMD (Single Instruction, Multiple Data)**
  - **Bloques organizados en grids:**
    - **MIMD (Multiple Instruction, Multiple Data)**
  - **Bibliotecas Paralelas:**
    - **CUDA, OpenCL, OpenACC**
    - **Maneja tipos de vectores (entreros, flotantes)**

Instituto Nacional de Medicina Genómica
MEXICO

Programación Paralela en GPUs

- **Procedimiento de ejecución en un GPU:**
  - **Reservar memoria para datos en el GPU**
  - **Mover datos de la memoria del CPU hacia la memoria del GPU, o asignar datos en memoria del GPU.**
  - **Iniciar ambiente de ejecución (kernel):**
    - **Driver del GPU crea el código intermedio para ejecutarse en el GPU.**
    - **Preserva la compatibilidad en los tipos de datos.**
  - **Acumular datos procesados del GPU.**
  - **Liberar memoria para datos en el GPU.**

Instituto Nacional de Medicina Genómica
MEXICO

Ventajas OpenACC

- **Programación con base en directivas estilo OpenMP.**
- **Portabilidad para otro tipo de aceleradores (Xeon Phi).**
- **No es tan fácil como insertar directivas, un buen algoritmo para un CPU no puede tener el mismo rendimiento en un GPU.**

Instituto Nacional de Medicina Genómica
MEXICO

Programación en OpenACC

- **Principales Directivas:**
  - **Fortran:**
    - **!$acc *directive*[*clause*]….**
  - **C:**
    - **#pragma acc *directive*[*clause*]...**

Instituto Nacional de Medicina Genómica
MEXICO

Programación en OpenACC

- **Definición de regiones:**
  - **Delimitada por directivas.**
  - **Los ciclos iterativos se colocan en los threads de GPUs.**
  - **Los datos para las regiones críticas, se alojan en el GPU.**

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP comparación con OpenACC

- **Ejemplo con OpenMP:**

```
#define SIZE 1000
float a[SIZE][SIZE];
float b[SIZE][SIZE];
float c[SIZE][SIZE];

int main()
{
  int i,j,k;

  // Initialize matrices.
  for (i = 0; i < SIZE; ++i) {
    for (j = 0; j < SIZE; ++j) {
      a[i][j] = (float)i + j;
      b[i][j] = (float)i - j;
      c[i][j] = 0.0f;
    }
  }
```

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP comparación con OpenACC

```
 // Compute matrix multiplication.
#pragma omp parallel for default(none) shared(a,b,c) private(i,j,k)
  for (i = 0; i < SIZE; ++i) {
    for (j = 0; j < SIZE; ++j) {
      for (k = 0; k < SIZE; ++k) {
    c[i][j] += a[i][k] * b[k][j];
      }
    }
  }
  return 0;
}
```

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP comparación con OpenACC

- **Ejemplo con OpenACC:**

```
!    matrix-acc.f
     program example1
     parameter ( n_size=1000 )
     real*4, dimension(:,:) :: a(n_size,n_size)
     real*4, dimension(:,:) :: b(n_size,n_size)
     real*4, dimension(:,:) :: c(n_size,n_size)

!    Initialize matrices (values differ from C version)
     do i=1, n_size
       do j=1, n_size
         a(i,j) = i + j;
         b(i,j) = i - j;
         c(i,j) = 0.;
       enddo
     enddo
```

Instituto Nacional de Medicina Genómica
MEXICO

OpenMP comparación con OpenACC

```
!$acc data copyin(a,b) copy(c)
!$acc kernels loop
!     Compute matrix multiplication.
      do i=1, n_size
        do j=1, n_size
          do k = 1, n_size
            c(i,j) = c(i,j) + a(i,k) * b(k,j)
          enddo
        enddo
      enddo
!$acc end data
 end program example1
```

Instituto Nacional de Medicina Genómica MEXICO

OpenACC comparación con CUDA

- Ejemplo CUDA:

```
#include <stdio.h>
#include <assert.h>
#include <cuda.h>
int main(void) {
float *a_h, *b_h;
 // pointers to host memory
float *a_d, *b_d;
 // pointers to device memory
int N = 14; int i;
// allocate arrays on host
a_h = (float *)malloc(sizeof(float)*N);
b_h = (float *)malloc(sizeof(float)*N);
// allocate arrays on device
cudaMalloc((void **) &a_d, sizeof(float)*N);
cudaMalloc((void **) &b_d, sizeof(float)*N);
// initialize host data
for (i=0; i<N; i++) {
a_h[i] = 10.f+i;
b_h[i] = 0.f;
 }
```

Instituto Nacional de Medicina Genómica MEXICO

OpenACC comparación con CUDA

```
// send data from host to device: a_h to a_d
cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
// copy data within device: a_d to b_d
cudaMemcpy(b_d, a_d, sizeof(float)*N, cudaMemcpyDeviceToDevice);
// retrieve data from device: b_d to b_h
cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
// check result
for (i=0; i<N; i++)
assert(a_h[i] == b_h[i]);
// cleanup
free(a_h);
free(b_h);
cudaFree(a_d);
cudaFree(b_d);
return 0;
}
```