SISTEMAS DISTRIBUIDOS Y PARALELOS

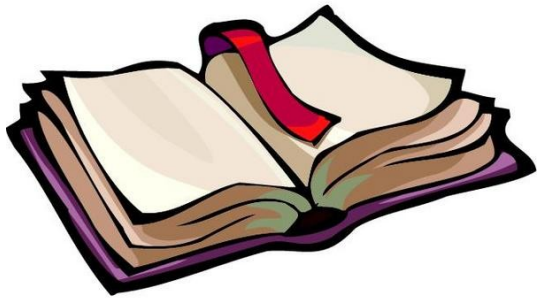
Carrera: Ingeniería en computación
Facultad de Informática – Universidad Nacional de La Plata

Dr. Adrián Pousa

Modelos híbridos

Agenda

I. *Modelo de programación híbrido: Introducción*
II. *MPI-OpenMP*
III. *MPI-Pthreads*
IV. *Modularidad*
    I. *MPI-OpenMP*
    II. *MPI-Pthreads*



Agenda

I. ***Modelo de programación híbrido: Introducción***

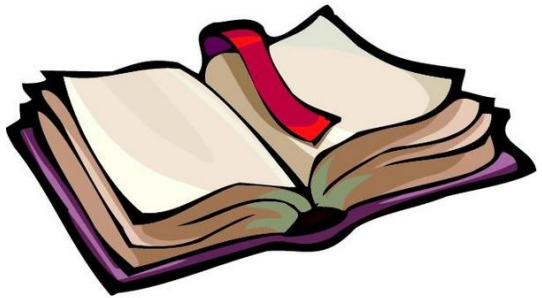
II. *MPI-OpenMP*

III. *MPI-Pthreads*

IV. *Modularidad*

    I. *MPI-OpenMP*

    II. *MPI-Pthreads*



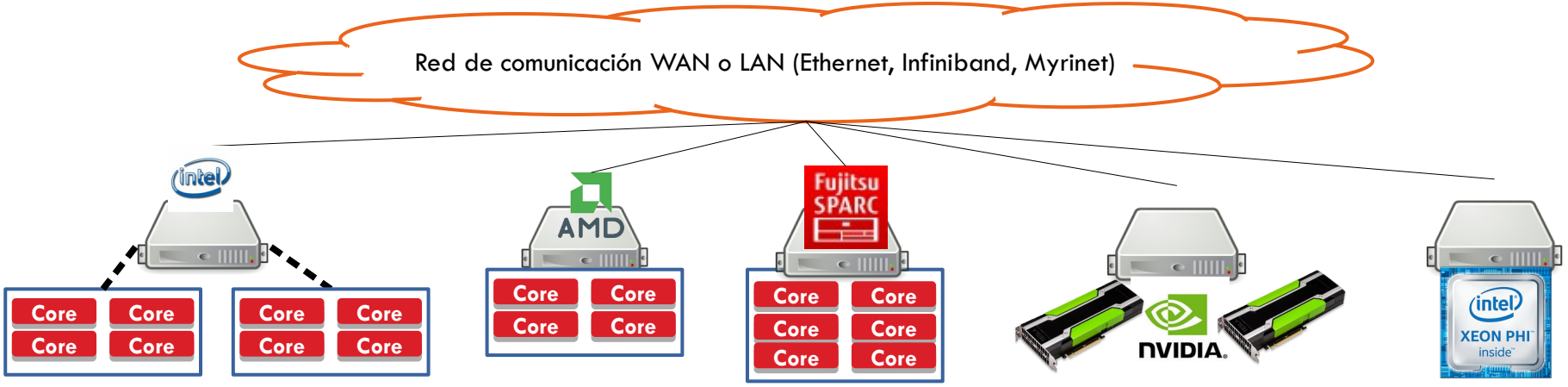
Modelo de programación

- Paralelismo Implícito

- Paralelismo Explicito:
  - Modelo de programación sobre memoria compartida
  - Modelo de programación sobre memoria distribuida
  - Modelos híbridos

Modelos híbridos

- Diversidad de arquitecturas actuales.
- Arquitecturas de memoria compartida y memoria distribuida pueden combinarse formando un modelo híbrido con mayor potencia de cómputo.
- Arquitecturas paralelas heterogéneas.



Modelos híbridos

- Para poder aprovechar la potencia de cómputo total es necesario integrar las distintas herramientas. Algunas combinaciones pueden ser:
  - MPI-OpenMP
  - MPI-Pthreads
  - **Otros:** MPI-Cuda, MPI-OpenMP-Cuda, OpenMP-Cuda etc.
- Estas combinaciones introducen complejidad a la hora de programar y compilar.

Agenda

I. *Modelo de programación híbrido: Introducción*

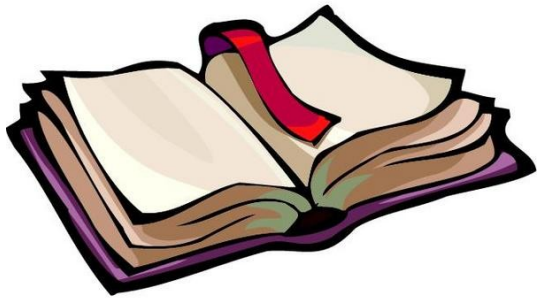
II. ***MPI-OpenMP***

III. *MPI-Pthreads*

IV. *Modularidad*

    I. *MPI-OpenMP*

    II. *MPI-Pthreads*



Modelos híbridos – MPI - OpenMP

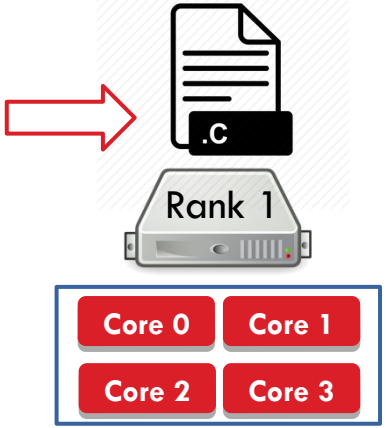
mpi_omp.c

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

int main(int argc, char* argv[]){

        int rank;
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        #pragma omp parallel
        {
                printf("Hello MPI rank %d thread %d\n",rank,omp_get_thread_num());
        }
        MPI_Finalize();
        return(0);
 }
```

Modelos híbridos – MPI - OpenMP

mpi_omp.c

Compilar: mpicc -fopenmp -o mpi_omp mpi_omp.c



Archivo de máquinas

Maquina1 slots=1
Maquina2 slots=1

Ejecutar: mpirun --bind-to none -np 2 –machinefile maquinas mpi_omp

Hello MPI rank 0 thread 0
Hello MPI rank 1 thread 2
Hello MPI rank 1 thread 0
Hello MPI rank 0 thread 1
Hello MPI rank 1 thread 3
Hello MPI rank 1 thread 1

Opcion para que no ejecute todos los hilos en el mismo core!!!

Agenda

I. *Modelo de programación híbrido: Introducción*

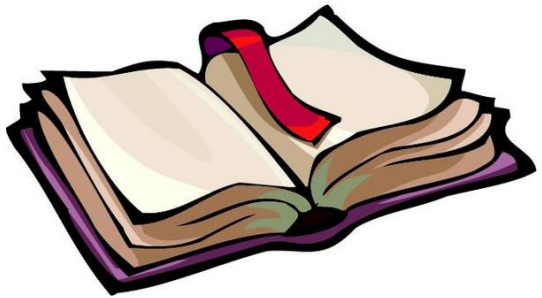
II. *MPI-OpenMP*

III. ***MPI-Pthreads***

IV. *Modularidad*

    I. *MPI-OpenMP*

    II. *MPI-Pthreads*



Modelos híbridos – MPI - Pthreads

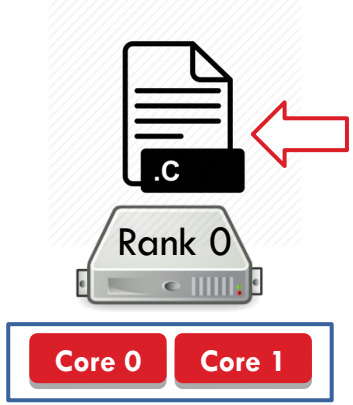
mpi_pthreads.c

```
#include <mpi.h>
#include <stdio.h>
#include <pthread.h>
#include <sys/sysinfo.h>

int rank;
int T;

void* funcion(void *arg){
 int id = *(int*)arg;
 printf("Hello MPI rank %d thread %d\n",rank,id);
 pthread_exit(NULL);
}
```

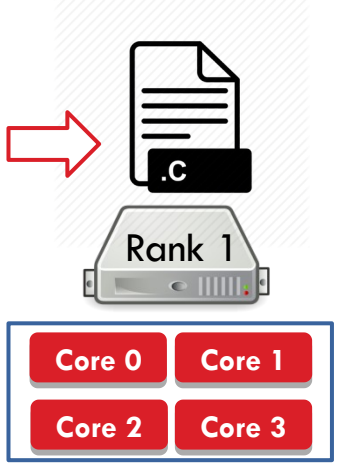
```
int main(int argc, char* argv[]){

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        T=get_nprocs ();
        pthread_t misHilos[T];
        int threads_ids[T];
        int id;
        for(id=0;id<T;id++){
                threads_ids[id] = id;
                pthread_create(&misHilos[id], NULL, &funcion,(void*)&threads_ids[id]);
        }

        for(id=0;id<T;id++)
                pthread_join(misHilos[id],NULL);

        MPI_Finalize();
        return(0);
}
```

Modelos híbridos – MPI - Pthreads

mpi_pthreads.c



Compilar: mpicc -lpthread -o mpi_pthreads mpi_pthreads.c



Archivo de máquinas

Maquina1 slots=1
Maquina2 slots=1

Ejecutar: mpirun -np 2 –machinefile maquinas mpi_pthreads

Hello MPI rank 0 thread 0
Hello MPI rank 0 thread 1
Hello MPI rank 1 thread 0
Hello MPI rank 1 thread 1
Hello MPI rank 1 thread 3
Hello MPI rank 1 thread 2

Agenda

I. *Modelo de programación híbrido: Introducción*

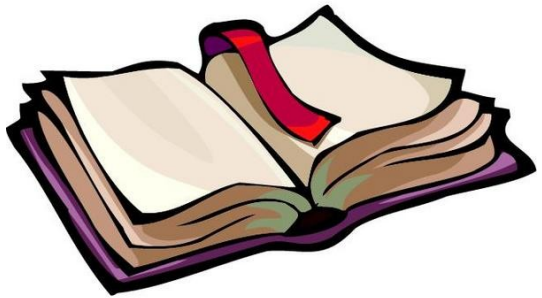
II. *MPI-OpenMP*

III. *MPI-Pthreads*

IV. ***Modularidad***

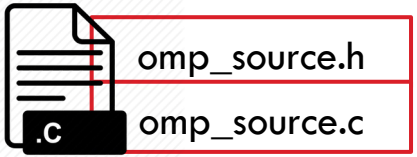
   I. ***MPI-OpenMP***

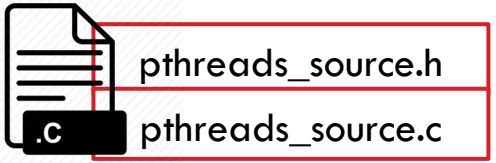
   II. ***MPI-Pthreads***



Modelos híbridos – Modularidad

- Una forma de organizar el código híbrido es tener el código de cada herramienta por separado (mayor modularidad y portabilidad).





- En ocasiones no contamos con el código fuente.
- Esto nos obliga a trabajar a nivel de compilador:
  - Utilizar un mismo compilador para compilar todos los módulos juntos
  - Compiladores diferentes para cada módulo y luego linkear código objeto (.o)

Agenda

I. *Modelo de programación híbrido: Introducción*

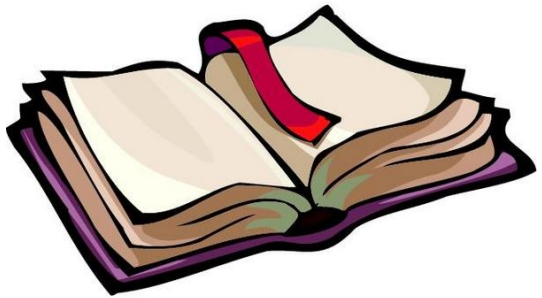
II. *MPI-OpenMP*

III. *MPI-Pthreads*

IV. ***Modularidad***

   I. ***MPI-OpenMP***

   II. *MPI-Pthreads*



Modelos híbridos – Modularidad MPI - OpenMP

mpi_source.c

```
#include <mpi.h>
#include <stdio.h>
#include <omp_source.h>

static void mpi_function(int rank){
  //Comunicacion con otros procesos MPI
  omp_function(rank);
  //Comunicacion con otros procesos MPI
}

int main(int argc, char* argv[]){

        int rank;
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);

        mpi_function(rank);

        MPI_Finalize();
        return(0);
 }
```

omp_source.h

```
#ifndef __OMP_FUNCTIONS__
#define __OMP_FUNCTIONS__

extern void omp_function(int rank);

#endif
```

omp_source.c

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

void omp_function(int rank){

#pragma omp parallel
{
   printf("Hello MPI rank %d thread %d\n",rank,omp_get_thread_num());
}

}
```

Modelos híbridos – Modularidad MPI - OpenMP

- Compilación unificada con el compilador de MPI:

```
mpicc -fopenmp -o mpi_omp mpi_source.c omp_source.c
```

- Compilación separada en 3 pasos:

Compilar sólo MPI:

```
mpicc -c mpi_source.c -o mpi_source.o
```

Compilar sólo OpenMP:

```
gcc -fopenmp -c omp_source.c –o omp_source.o
```

Linkear con MPI:

```
mpicc -lgomp -o mpi_omp mpi_source.o omp_source.o
```

Agenda

I. *Modelo de programación híbrido: Introducción*

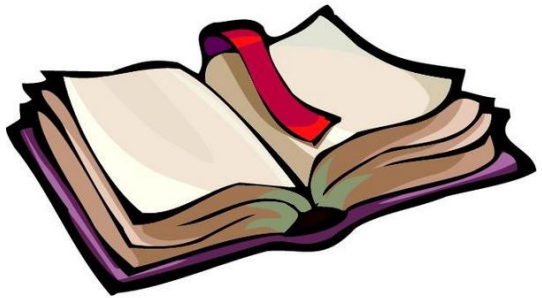
II. *MPI-OpenMP*

III. *MPI-Pthreads*

IV. ***Modularidad***

   I. *MPI-OpenMP*

   II. ***MPI-Pthreads***



Modelos híbridos – Modularidad MPI - Pthreads

mpi_source.c

```
#include <mpi.h>
#include <stdio.h>
#include <pthreads_source.h>

static void mpi_function(int rank){
  //Comunicacion con otros procesos MPI
  pthreads_function(rank);
  //Comunicacion con otros procesos MPI
}

int main(int argc, char* argv[]){
  int rank;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);

  mpi_function(rank);

  MPI_Finalize();
  return(0);
 }
```

pthreads_source.h

```
#ifndef __PTHREADS_FUNCTIONS__
#define __PTHREADS_FUNCTIONS__

extern void pthreads_function(int miRank);

#endif
```

pthreads_source.c

```
#include <pthread.h>
#include<stdio.h>
int rank;
int T;

void* funcion(void *arg){
 int id = *(int*)arg;
 printf("Hello MPI rank %d thread %d\n",rank,id);
 pthread_exit(NULL);
}

void pthreads_function(int miRank){
 rank=miRank;  T=get_nprocs (); pthread_t misHilos[T];
 int threads_ids[T];  int id;

 for(id=0;id<T;id++){
   threads_ids[id] = id;
   pthread_create(&misHilos[id], NULL, &funcion,(void*)&threads_ids[id]);
 }
 for(id=0;id<T;id++)
   pthread_join(misHilos[id],NULL);
}
```

Modelos híbridos – Modularidad MPI - Pthreads

- Compilación unificada con el compilador de MPI:

```
mpicc -lpthread -o mpi_pthreads mpi_source.c pthreads_source.c
```

- Compilación separada en 3 pasos:

Compilar sólo MPI:

```
mpicc -c mpi_source.c -o mpi_source.o
```

Compilar sólo Phtreads:

```
gcc -lpthread -c pthreads_source.c -o pthreads_source.o
```

Linkear con MPI:

```
mpicc -o mpi_pthreads mpi_source.o pthreads_source.o
```