

MPI :

Message Passing Interface MPI

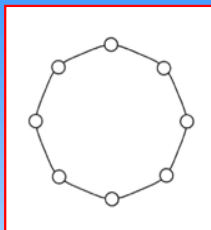
1

Topologie MPI

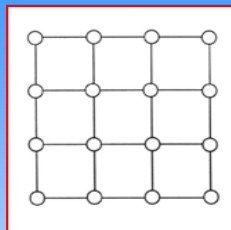
Le topologie.

2

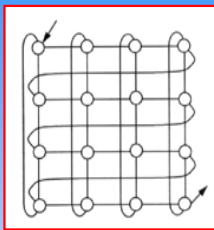
Esempi di topologie



Anello



Griglia



Toro

L'utilizzo di una topologia per la progettazione di un algoritmo in ambiente MIMD è spesso **legata** alla geometria "intrinseca" del problema in esame.

3

Definizione: *topologia*

Una topologia è
la geometria "virtuale"
in cui si immaginano disposti
i processori.



La topologia "virtuale" in cui sono disposti
i processori può **non avere alcun nesso**
con la disposizione "reale" degli stessi!

4

Funzioni per la creazione di topologie

Funzione per la creazione di un *grafo* di processori

MPI_Graph_create

Funzione per la creazione di una *griglia* di processori (periodica e non)

MPI_Cart_create

5

Esempio: creazione di un grafo

1/4

```
#include <stdio.h>
#include "mpi.h"
/* Scopo: definizione di un grafo */
main(int argc, char **argv)
{ int nenum,nproc;
  int ord,*index,*comunico;
  MPI_Comm comm_graph;

  ...
  ord=0;
  a1=4; /* numero di nodi del grafo */
  index=(int *)calloc(a1,sizeof(int));
  /* Assegnazione del numero di nodi adiacenti */
  index[0]=2; index[1]=3; index[2]=4; index[3]=6
  a2=a1-1;
  a3=index[a2];
  edg=(int *)calloc(a3,sizeof(int));
  /* Assegnazione dei nodi adiacenti */
  edg[0]=1; edg[1]=3; edg[2]=0; edg[3]=3; edg[4]=0; edg[5]=2;

  MPI_Graph_create(MPI_COMM_WORLD,a1,index,edg,ord,&Graph);

  MPI_Finalize();
  return 0; }
```

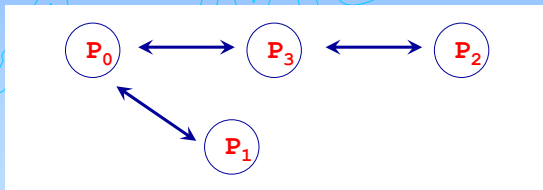
6

Nel programma ... :

2/4

MPI_Graph_create(MPI_COMM_WORLD,a1,index,edg,ord,&Graph);

- Ogni processo dell'ambiente **MPI_COMM_WORLD** definisce il graph denominato **Graph** costituito da 4 nodi (**a1**). Secondo quanto memorizzato nei vettori **index** e **edg** il processo **P₀** è adiacente a **P₁** e **P₃**, **P₁** a **P₀**, **P₂** a **P₃** e **P₃** a **P₀** e **P₂**. I processi non sono riordinati secondo un particolare schema (**ord=0**).



7

Funzione per creare un grafo di processori

3/4

MPI_Graph_create(MPI_Comm comm_old,int nodes, int *index, int *edges, int *reorder, MPI_Comm *new_comm);

- Operazione collettiva che restituisce un nuovo communicator **new_comm** in cui i processori sono organizzati in un grafo con numero di nodi pari a **nodes**.
- index** ha dimensione **nodes**, **index[i]** indica il numero dei nodi adiacenti all'**i**-esimo nodo. **index[0]** indica il grado del nodo 0; **index[i] - index[i-1]** il grado dell'**i**-esimo nodo, con **i > 0**
- Per il nodo 0 la lista dei nodi adiacenti è contenuta in **edges[j]** per **0 ≤ j ≤ index[0]-1**. Per il nodo **i**, con **i > 0**, la lista è contenuta in **edges[j]**, con **index[i-1] ≤ j ≤ index[i]-1**.

8

In dettaglio...

4/4

```
MPI_Graph_create(MPI_Comm comm_old, int nodes,
                 int *index, int *edges,
                 int *reorder,
                 MPI_Comm *new_comm);
```

comm_old communicator di input

nodes numero di nodi del grafo

***index** vettore di dimensione **nodes** contenente per ciascun nodo il numero di nodi adiacenti ad esso

***edges** vettore contenente per ciascun nodo la lista dei nodi adiacenti

***reorder** permesso di riordinare i **menum** (1=si; 0=no)

***new_comm** communicator di output associato al grafo

9

Esempio: creazione di una griglia bidimensionale

1/2

```
#include <stdio.h>
#include "mpi.h"
/* Scopo: definizione di una topologia
   a griglia bidimensionale nproc=row*col */
main(int argc, char **argv)
{ int menum, nproc, row, col;
  int dim, *ndim, reorder, *period;
  int coordinate[2];
  MPI_Comm comm_grid;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &menum);
  MPI_Comm_size(MPI_COMM_WORLD, &nproc);
  /* Numero di righe della griglia di processo */
  if (menum == 0)
  { printf("Numero di righe della");
    scanf("%d", &row); }
  /* Spedizione di row da parte di 0 a tutti i processo */
  MPI_Bcast(&row, 1, MPI_INT, 0, MPI_COMM_WORLD);
  /* Definizione del numero di colonne della griglia */
  col = nproc/row;
  /* Numero di dimensioni della griglia */
  dim = 2;
```

10

Esempio

2/2

```
/* vettore contenente le lunghezze di ciascuna dimensione */
ndim = (int*)calloc(dim, sizeof(int));
ndim[0] = row;
ndim[1] = col;
/* vettore contenente la periodicità delle dimensioni */
period = (int*)calloc(dim, sizeof(int));
period[0] = period[1] = 0;
reorder = 0;
/* Creazione della griglia bidimensionale */
MPI_Cart_Create(MPI_COMM_WORLD, dim, ndim, period, reorder,
               &comm_grid);

MPI_Comm_rank(MPI_COMM_WORLD, &menum_grid);
/* Definizione delle coordinate di ciascun processo
   nella griglia bidimensionale */
MPI_Cart_coords(comm_grid, menum, dim, coordinate);
/* Stampa delle coordinate */
printf("Processore %d coordinate nella griglia\n",
       menum, *coordinate, *(coordinate+1));
MPI_Finalize();
return 0; }
```

11

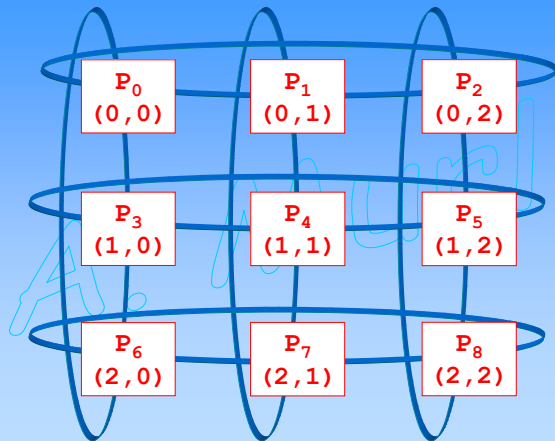
Nel programma ... :

MPI_Cart_Create(MPI_COMM_WORLD, dim, ndim, period, reorder, &comm_grid);

- Ogni processo dell'ambiente **MPI_COMM_WORLD** definisce la griglia denominata **comm_grid**, di dimensione 2 (**dim**) e non periodica lungo le due componenti (**period**_i=0, i=0,1). Il numero di righe e di colonne della griglia sono memorizzati rispettivamente nella prima e nella seconda componente del vettore **ndim**. I processi non sono riordinati secondo un particolare schema (**reorder**_i=0, i=0,1).

12

Griglia periodica (3,3):



13

In generale ... :

```
MPI_Cart_create(MPI_Comm comm_old, int dim,
                int *ndim, int *period,
                int *reorder,
                MPI_Comm *new_comm);
```

- Operazione collettiva che restituisce un nuovo communicator **new_comm** in cui i processi sono organizzati in una griglia di dimensioni **dim**.
- L'*i*-esima dimensione ha lunghezza **ndim[i]**.
- Se **period[i]=1** indica che la *i*-esima dimensione della griglia è periodica, altrimenti **period[i]=0**.

14

In dettaglio...

```
MPI_Cart_create(MPI_Comm comm_old, int dim,
                int *ndim, int *period,
                int *reorder,
                MPI_Comm *new_comm);
```

comm_old communicator di input
dim numero di dimensioni della griglia
***ndim** vettore di dimensione **dim** contenente le lunghezze di ciascuna dimensione
***period** vettore di dimensione **dim** contenente la periodicità di ciascuna dimensione
***reorder** permesso di riordinare i **menum** (1=si; 0=no)
***new_comm** communicator di output associato alla griglia

15

Esempio

2/2

```
/* vettore contenente le lunghezze di ciascuna dimensione */
ndim = (int*)calloc(dim, sizeof(int));
ndim[0] = row;
ndim[1] = col;
/* vettore contenente la periodicità delle dimensioni */
period = (int*)calloc(dim, sizeof(int));
period[0] = period[1] = 0;
reorder = 0;
/* Definizione della griglia bidimensionale */
MPI_Cart_Create(MPI_COMM_WORLD, dim, ndim, period, reorder,
               &comm_grid);
MPI_Comm_rank(MPI_COMM_WORLD, &menum_grid);
/* Definizione delle coordinate di ciascun processo
   nella griglia bidimensionale */
MPI_Cart_coords(comm_grid, menum, dim, coordinate);
/* Stampa delle coordinate */
printf("Processore %d coordinate nella griglia
       (%d,%d) \n", menum, *coordinate, *(coordinate+1));
MPI_Finalize();
return 0; }
```

16

Nel programma ... :

➔ `MPI_Cart_coords(comm_grid,menum,dim,coordinate);`

- Ogni processo `menum` calcola le proprie 2 (`dim`) coordinate (`coordinatei`, $i=0,1$) nell'ambiente `comm_grid`.

17

Funzione per definire le coordinate

```
MPI_Cart_coords(MPI_Comm comm_grid,  
                int menum_grid, int dim,  
                int *coordinate);
```

- Operazione collettiva che restituisce a ciascun processo di `comm_grid` con identificativo `menum_grid`, le sue coordinate all'interno della griglia predefinita.
- `coordinate` è un vettore di dimensione `dim`, i cui elementi rappresentano le coordinate del processo all'interno della griglia.

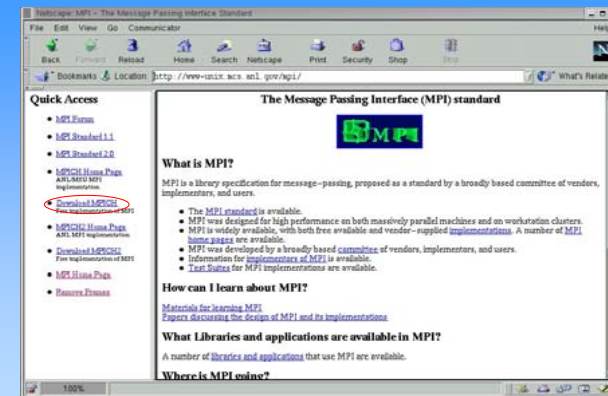
18

Fine Lezione

19

Dove recuperare MPI

1/2

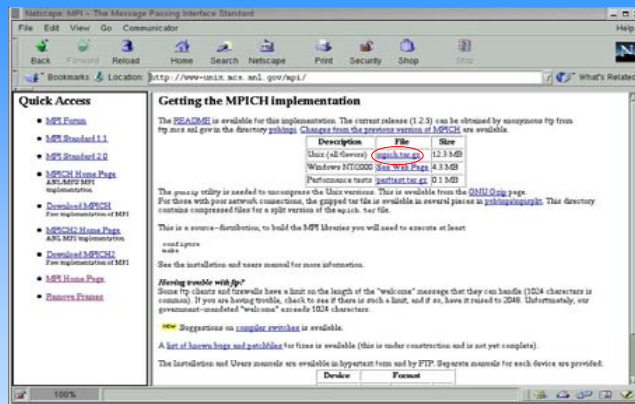


www-unix.mcs.anl.gov/mpi/

20

Dove recuperare MPI

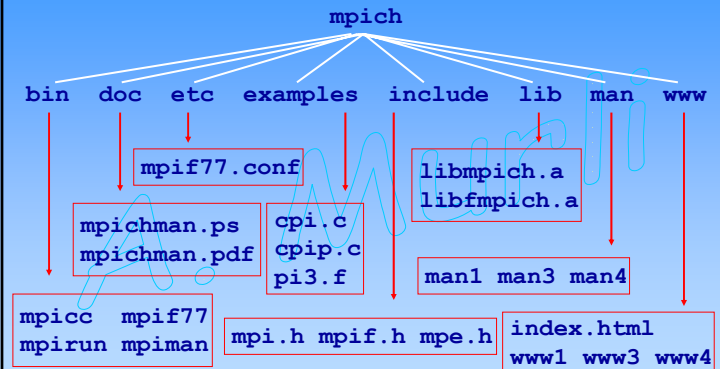
2/2



www-unix.mcs.anl.gov/mpi/

21

Esplorazione directory MPI



Alcune librerie MPI sono proprietarie e alcune sono di dominio pubblico

Message passing :

Il modello del **Message Passing** è oggi **reso disponibile** da differenti librerie software proprietarie e di pubblico dominio.



Librerie di dominio pubblico:

- **PVM** (Parallel Virtual Machine System)
- **P4** (Portable Programs for Parallel Processors)
- ...

23