1 Bases en MPI

Hello world en MPI

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
int main( int argc, char** argv ) {
  int size, rank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size( MPI_COMM_WORLD, &size);
  MPI_Comm_rank( MPI_COMM_WORLD, &rank);

  printf("I am rank %d in %d\n", rank, size);

  MPI_Finalize();
  return EXIT_SUCCESS;
}
```

Exécution réelle

```
  # Compilation (gcc avec les bonnes options)
$ mpicc -o helloworld helloworld.c
  # gcc -L/my/mpi/lib -lmpi -I/my/mpi/include ..

  # Exécution
$ mpirun -np 4 -hostfile machines ./helloworld
I am rank 3 in 4
I am rank 2 in 4
I am rank 1 in 4
I am rank 0 in 4
```

Fichier machines

```
  host-0
  host-1
  host-2
  ...
```

Exécution simulée avec SMPI

```
  # Compilation (lance gcc avec les bonnes options)
$ smpicc -o helloworld helloworld.c

  # Exécution
$ smpirun -np 4 -hostfile machines -platform ./cluster.xml ./helloworld
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'smpi/privatization' to '1'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'smpi/np' to '4'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'smpi/hostfile' to 'cluster_hostfile.txt'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'surf/precision' to '1e-9'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'network/model' to 'SMPI'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'smpi/tmpdir' to '/tmp'
[0.000000] [smpi_config/INFO] You did not set the power of the host running the simulation.
           The timings will certainly not be accurate.  Use the option "--cfg=smpi/host-speed:<flops>" to set its
           Check https://simgrid.org/doc/latest/Configuring_SimGrid.html#automatic-benchmarking-of-smpi-code for m
I am rank 0 in 4
I am rank 1 in 4
I am rank 2 in 4
I am rank 3 in 4
```

2 Communications point-à-point bloquantes

```c
int MPI_Send( void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm )
```

```c
int MPI_Recv( void *buf, int count, MPI_Datatype datatype, int orig, int tag, MPI_Comm comm, MPI_Status *status )
```

Exemple de ping-pong en MPI

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
int main( int argc, char** argv ) {
  int rank;
  int token = 42;
  MPI_Status status;
  MPI_Init( &argc, &argv );
  MPI_Comm_rank( MPI_COMM_WORLD, &rank );
  if( 0 == rank ) {
    MPI_Send( &token, 1, MPI_INT, 1, 0, MPI_COMM_WORLD );
    MPI_Recv( &token, 1, MPI_INT, 1, 0, MPI_COMM_WORLD, &status );
  } else if ( 1 == rank ) {
    MPI_Recv( &token, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status );
    MPI_Send( &token, 1, MPI_INT, 0, 0, MPI_COMM_WORLD );
  }
  MPI_Finalize();
  return EXIT_SUCCESS;
}
```

Synchronisation par vague

```
nclude <stdio.h>
nclude <stdlib.h>
nclude <mpi.h>

efine N (1024 * 1024 * 1)

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

int size, rank;
struct timeval start, end;

MPI_Init(&argc, &argv);

MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

char *buf = malloc(sizeof(char) * N); //10MiB

// Communicate along the ring
if (rank == 0) {
  gettimeofday(&start,NULL);
  printf("Rank %d: sending the message rank %d\n",rank,1);
  MPI_Send(buf, N, MPI_BYTE, 1, 1, MPI_COMM_WORLD);
  MPI_Recv(buf, N, MPI_BYTE, size-1, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  printf("Rank %d: received the message from rank %d\n",rank,size-1);
  gettimeofday(&end,NULL);
  printf("%f\n",(end.tv_sec*1000000.0 + end.tv_usec -
         start.tv_sec*1000000.0 - start.tv_usec) / 1000000.0);
} else {
  MPI_Recv(buf, N, MPI_BYTE, rank-1, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  int dest = (rank+1)%size;
  printf("Rank %d: sending the message to rank %d\n",rank,dest);
  MPI_Send(buf, N, MPI_BYTE, dest, 1, MPI_COMM_WORLD);
}

MPI_Finalize();
return 0;

```

3 Communications point-à-point asynchrones

```
int MPI_Isend(void *buf, int count, MPI_Datatype dt, int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Irecv(void *buf, int count, MPI_Datatype dt, int orig, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Wait( MPI_Request* request, MPI_Status* status )
int MPI_Waitall(int count, MPI_Request* reqs, MPI_Status *stats)
int MPI_Waitany(int count, MPI_Request* reqs, int *index, MPI_Status *stat)

int MPI_Test( MPI_Request *request, int *flag, MPI_Status, *status )
```

4 Quelques sites web utiles

- `https://theartofhpc.com/`: Plusieurs livres très complets sur le sujet. MPI est dans le tome 2; Makefile, cmake et git sont dans le tome 4.
- `https://web.corral.tacc.utexas.edu/CompEdu/pdf/pcse/mpi_c_course.pdf` 400 slides du même auteur: Cours de MPI en C.
- `http://condor.cc.ku.edu/~grobe/docs/intro-MPI-C.shtml` Une autre introduction au MPI en C.