HP-SEE DISSEMINATION & TRAINING

Tirana, 17 November 2011

MPI

EC FP7 Project HP-SEE

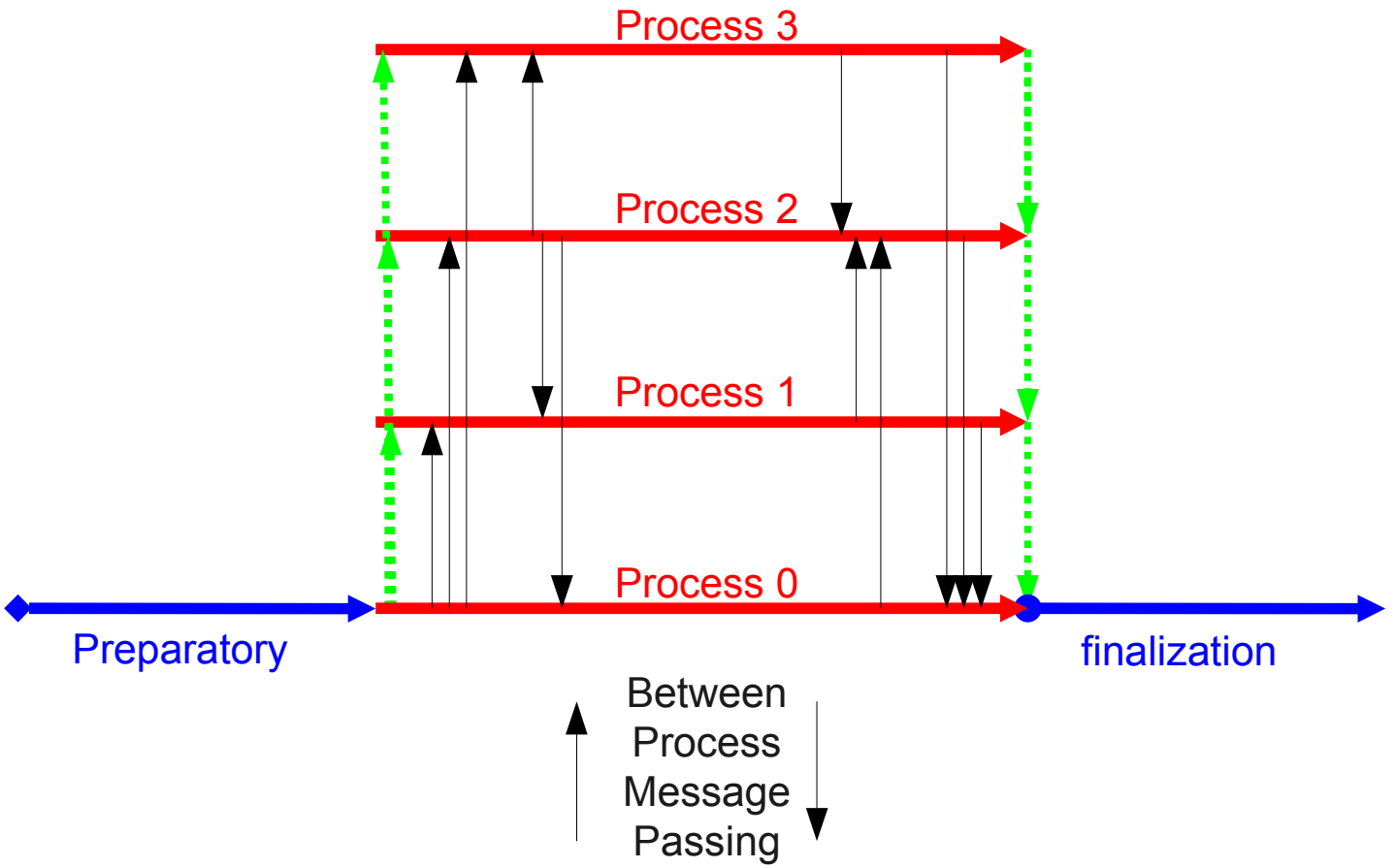
http://www.hp-see.eu/

N. Frasheri

Polytechnic University of Tirana

Parallelization ~ MPI

MESSAGE PASSING INTERFACE



Parallel Processes

- Parallel processes may run in separate nodes
  - Clusters of PCs in LAN => grids
  - Parallel systems
- Data are exchanged between processes through exchange of messages
  - process – to – process
  - process – to – group of processes
  - group of processes – to – process
  - interprocess synchronization

Basic Routines

MPI_COMM_WORLD : predefined communicator ~ context

MPI_Init (`int *argc, char *argv`)

MPI_Comm_size (`MPI_COMM_WORLD, &size`) => no.processes

MPI_Comm_rank (`MPI_COMM_WORLD, &rank`) => process id

MPI_Send (`void *buffer, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm`)

MPI_Recv (`void *buffer, int count, MPI_Datatype type, int src, int tag, MPI_Comm comm, MPI_Status status`)

MPI_Finalize ()

More Routines

- Group communication
  - MPI_Scatter (&sendbuf, sendcnt, sendtype, &recvbuf, recvcnt, recvtype, source, comm)
  - MPI_Gather (&sendbuf, sendcnt, sendtype, &recvbuf, recvcount, recvtype, destination, comm)
  - other ...
- Blocking versus Not Blocking
  - Blocking requested for synchronization
  - Different routines for communication modes

Example 1

```
#include <stdio.h>
#include <mpi.h>
int main (int argc, char * argv[])
{   int procs, nodes;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &nodes);
    MPI_Comm_rank(MPI_COMM_WORLD, &procs);
    printf("Hello MPI process %d of %d \n",procs,nodes);
    MPI_Finalize();
    return 0;
}
```

Compilation & Execution

- Compilation (script `build.sh`)

```
mpicc $1.c -o $1
```

  where $1 ~ source name

- Execution (script `run.sh`)

```
mpirun -np $2 ./$1 $3
```

  where $1 ~ code; $2 ~ processes; $3 ~ ???

Example 1 Output

- `test-mpi$ ./run.sh test2 5`

```
Hello MPI process 1 of 5
Hello MPI process 3 of 5
Hello MPI process 4 of 5
Hello MPI process 0 of 5
Hello MPI process 2 of 5
```

- `test-mpi$`

Example 2

```
#include “mpi.h” ...
int main(int argc, char **argv){
int rank, size, n, i;
double *x, *y, *data1, *data2;
n = atoi(argv[1]) ...
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
chunk = n / size ...
MPI_Scatter(&data1[rank*chunk],chunk,MPI_DOUBLE,x,chunk,
 MPI_DOUBLE,0,MPI_COMM_WORLD);
MPI_Scatter(&data2[rank*chunk],chunk,MPI_DOUBLE,y,chunk,
 MPI_DOUBLE,0,MPI_COMM_WORLD);
for (i=0; i<chunk; i++) x[i] = x[i] + y[i];
MPI_Gather(x,chunk,MPI_DOUBLE,buff,chunk,MPI_DOUBLE,0,
 MPI_COMM_WORLD);
MPI_Finalize(); }
```

If (rank==0)
input(data1,data2)

Download & Install

- http://www.open-mpi.org/
- Linux binaries from repositories
- Runtime from http://software.intel.com/
- http://www.openchannelsoftware.com/
- ...

Conclusions

- Compared with OpenMP
  - Much more complicated
  - Much more powerful
  - Works in distributed environment
    - No need for shared memory
    - Grids
    - HPC
  - May be combined with OpenMP
    - MPI to split separate processes to individual nodes
    - OpenMP to split processes in threads per core in nodes

Thank You

- Q & A

HP-SEE

High-Performance Computing Infrastructure
for South East Europe's Research Communities