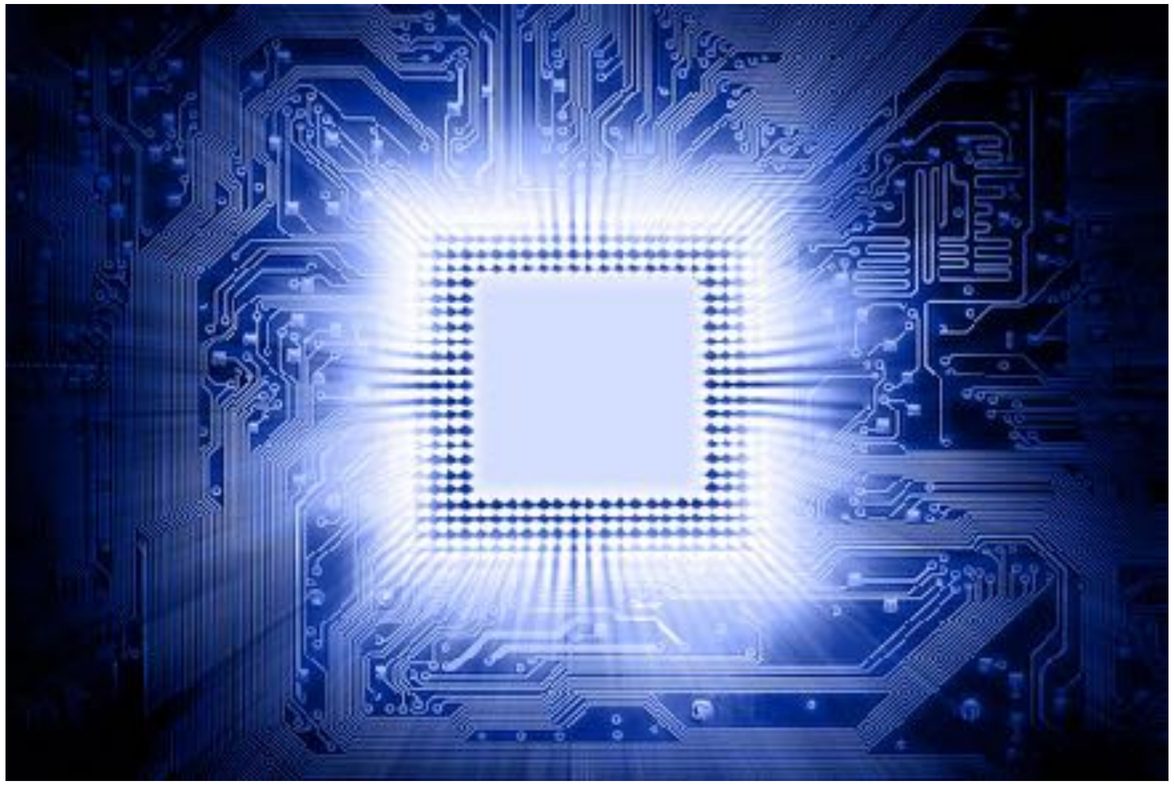
OpenMP & MPI

Group 5: QIANG SUN

Content

- High Performance Computing
- Multiple Threads
- OpenMP
- MPI
- Run openMP & MPI together
- Reference

HPC



Ref: http://hpc-asia.com/altair-partners-south-africas-centre-high-performance-computing/

CPU / Processor

- Processors is more and more powerful but seem to reach its limit.
- We are trying to solve larger and larger problems.
- And we want to save our time, make program run faster.
- So, Dividing a task into sub-tasks and run them on different cores or machines at the same time.
- Generally, it is parallelism, it is **HPC.**

Multiple Threads

- Process & Thread

- A process creates multiple threads and these threads share the address space of the parent process (Shared Memory)

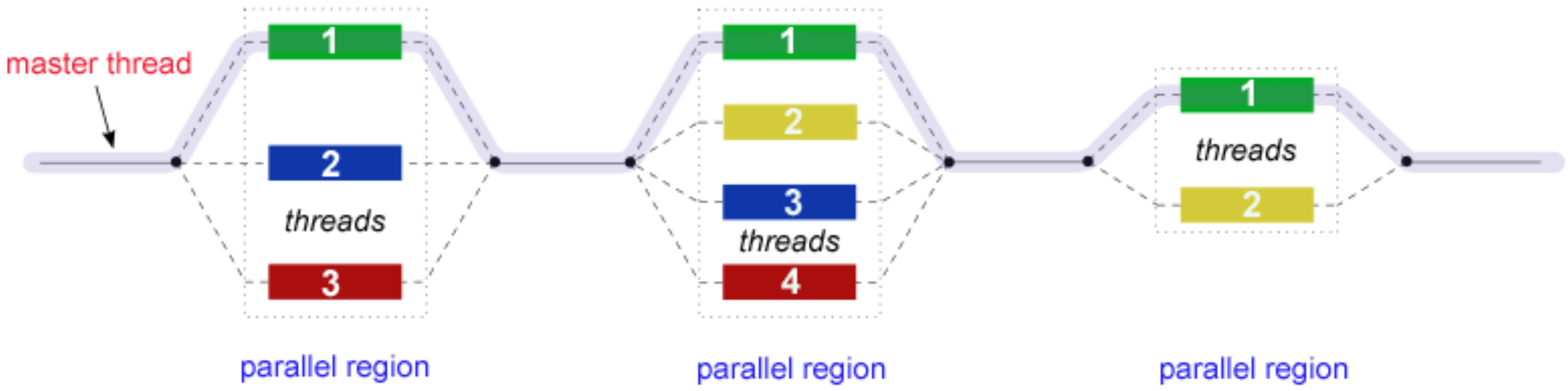
- Each language has its own built-in multi-threading interface

- However, they can’t make the most of the resources.

OpenMP

- An API for Writing Multithreaded Applications
  - A set of compiler directives and library routines for parallel application programmer
  - Greatly simplifies writing multi-threaded programs in Fortan, C and C++
  - With this, Multi-core resources can be fully taken advantage of.
  - Easy to use

OpenMP



:

- Fork-Join model

- **FORK:** the master thread then creates a team of parallel ***threads***

- **JOIN:** When the team threads complete the statements in the parallel region construct, they synchronize and terminate, leaving only the master thread.

Ref: https://computing.llnl.gov/tutorials/openMP/

OpenMP core syntax

- include the library: #include <omp.h>
- General Compiler directives: #pragma omp construct [clause [clause]]
- Compile openMP programs: gcc -fopenmp file.c

OpenMP example

```
#include <omp.h>

 main ()  {

 int var1, var2, var3;

 Serial code
       .
       .
       .

 Beginning of parallel region. Fork a team of threads.
 Specify variable scoping

 #pragma omp parallel private(var1, var2) shared(var3)
    {

    Parallel region executed by all threads
               .
    Other OpenMP directives
               .
    Run-time Library calls
               .
    All threads join master thread and disband

    }

 Resume serial code
       .
       .
       .

 }
```

OpenMP example

```
#include <stdio.h>
#include <omp.h>
int main(void)
{
    omp_set_num_threads(4); //I have set the number of threads =4, you can change this

    #pragma omp parallel   //Directive show the block below will be executed with openmp
    printf("Hello, world.\n");
    return 0;
}
```

OpenMP example

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

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

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }

  }  /* All threads join master thread and disband */

}
```

OpenMP example

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

int main (int argc, char *argv[])
{
int nthreads, tid;
omp_set_num_threads(4);
int k =0;
/* Fork a team of threads giving them their own copies of variables */

#pragma omp parallel for  private(nthreads, tid) shared (k)
for(k=0;k<8;k++)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d and %d\n", tid,k);


  }  /* All threads join master thread and disband */

}
```

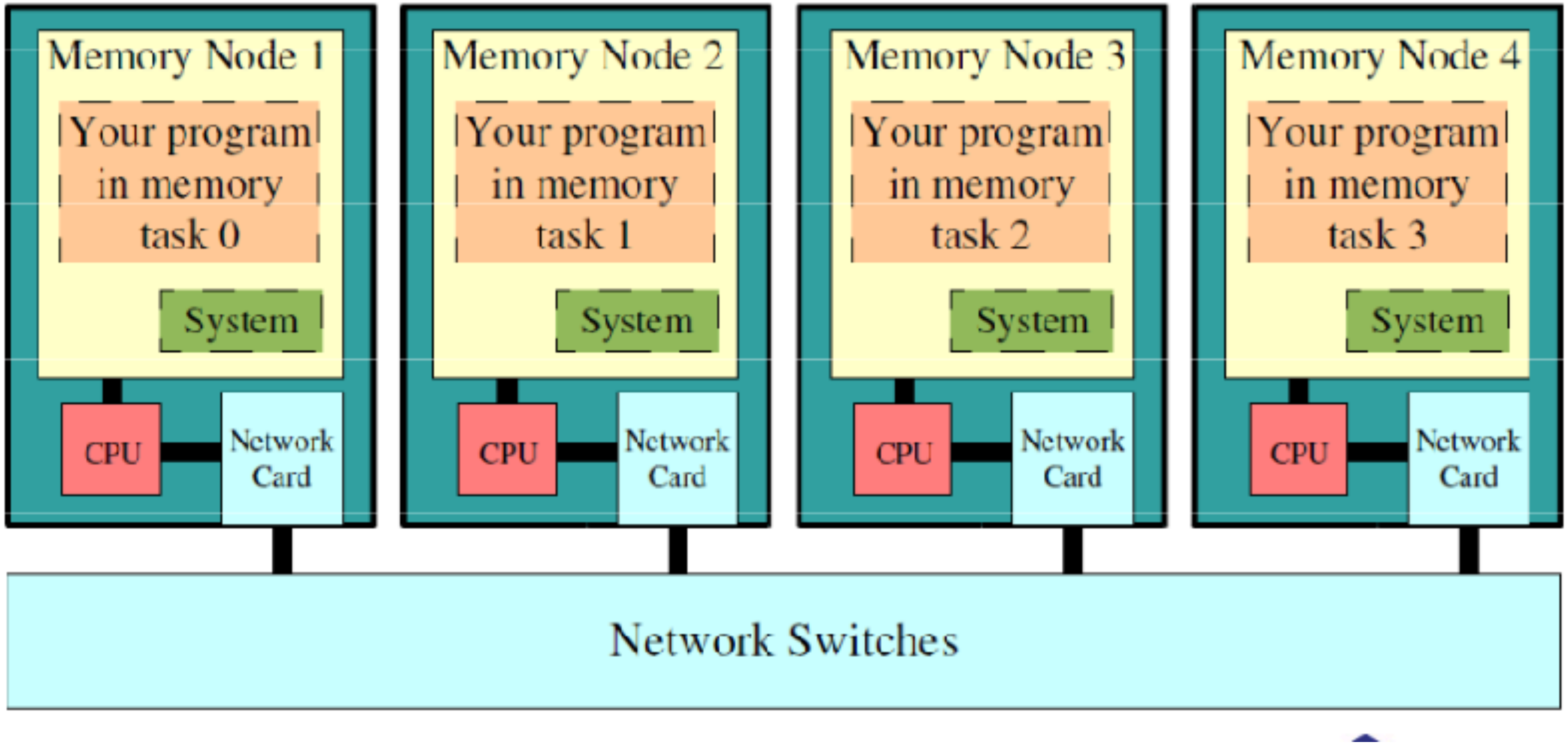
OpenMP core syntax

- include the library: #include <omp.h>
- General Compiler directives: #pragma omp construct [clause [clause]]
- Compile openMP programs: gcc -fopenmp file.c
- More references and tutorial:
  - https://computing.llnl.gov/tutorials/openMP/

MPI

- OpenMP: Threads share memory
- However, usually one PC is not powerful enough
- So can we combine machines together to do the calculation.
- Yes, and Something like this called **Distributed Memory Clusters**

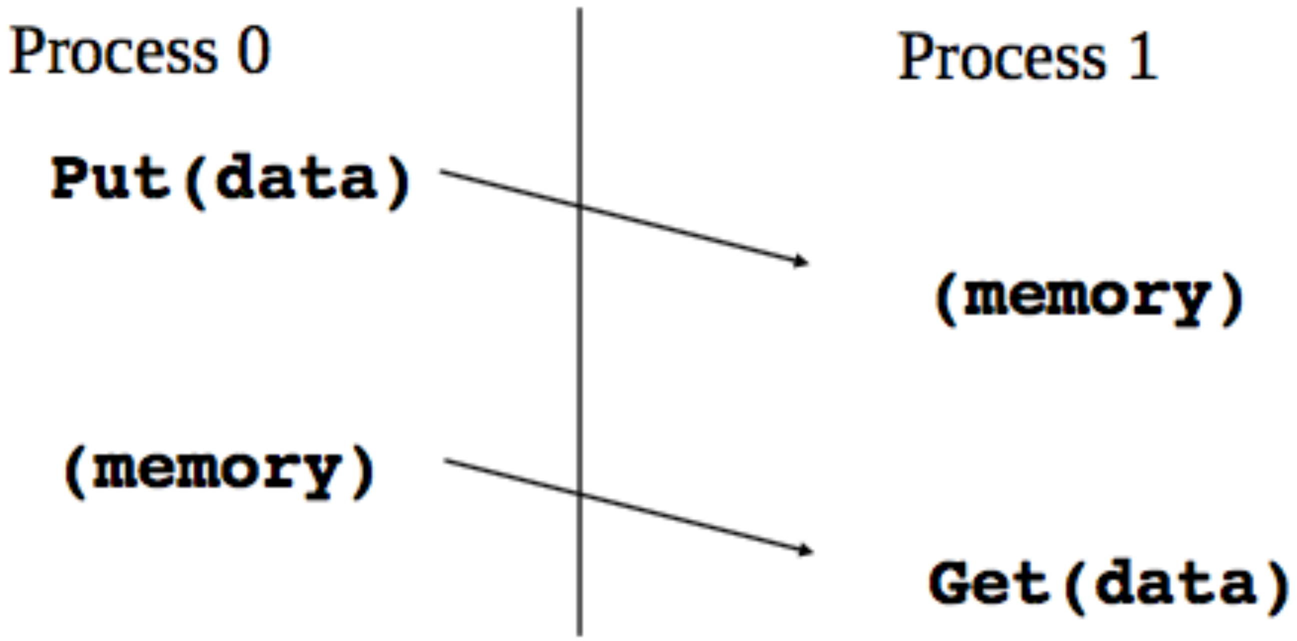
MPI



MPI: Message-Passing Interface is designed to communicate between process with separate address

Ref: http://www.ssc.net.cn/files/pdf/28.pdf

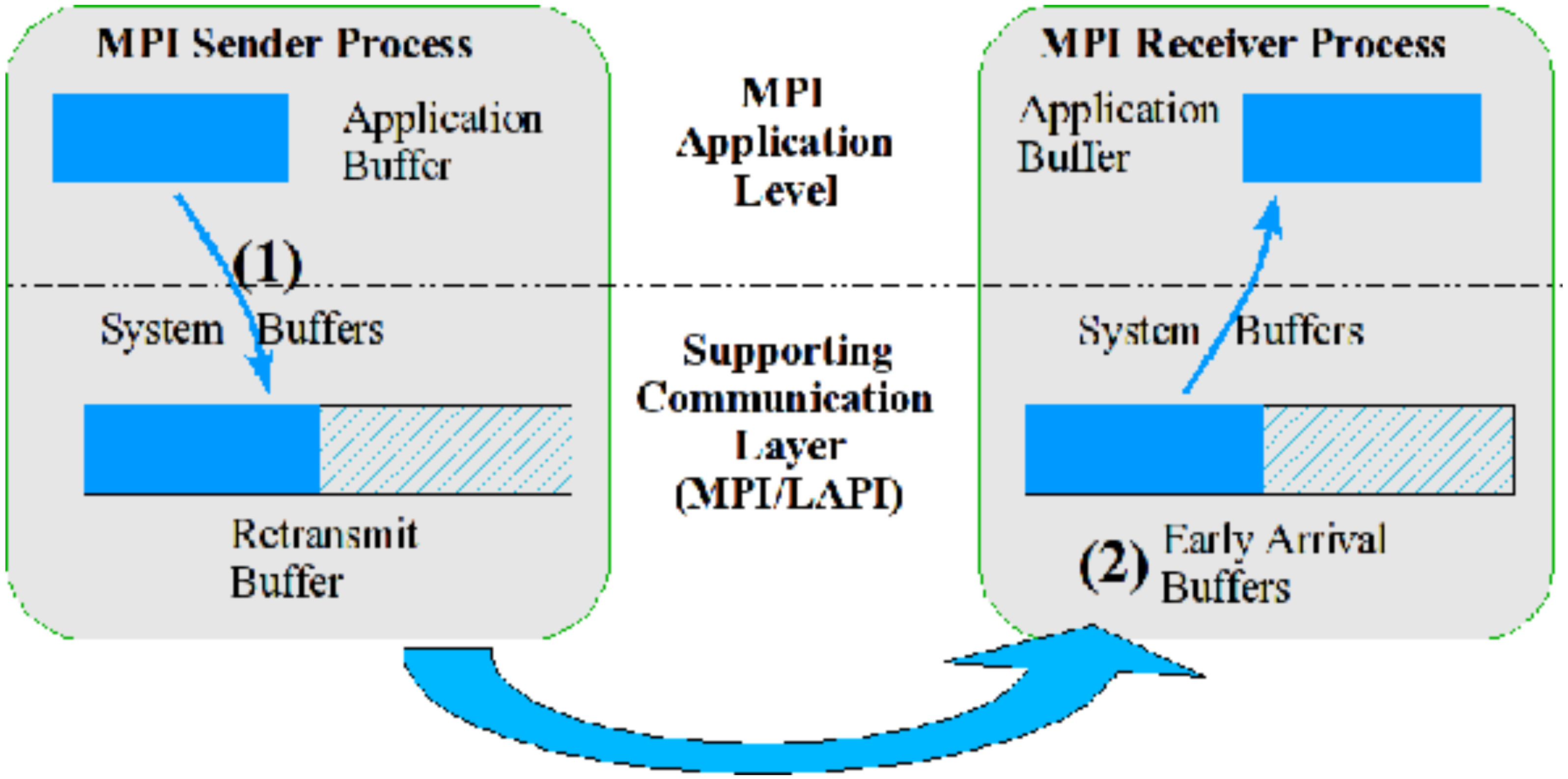
MPI



MPI: Message-Passing Interface is designed to communicate between process with separate address

Ref: http://www.ssc.net.cn/files/pdf/28.pdf

MPI



MPI: Message-Passing Interface is designed to communicate between process with separate address

Ref: http://sc.tamu.edu/systems/hydra/hardware.php

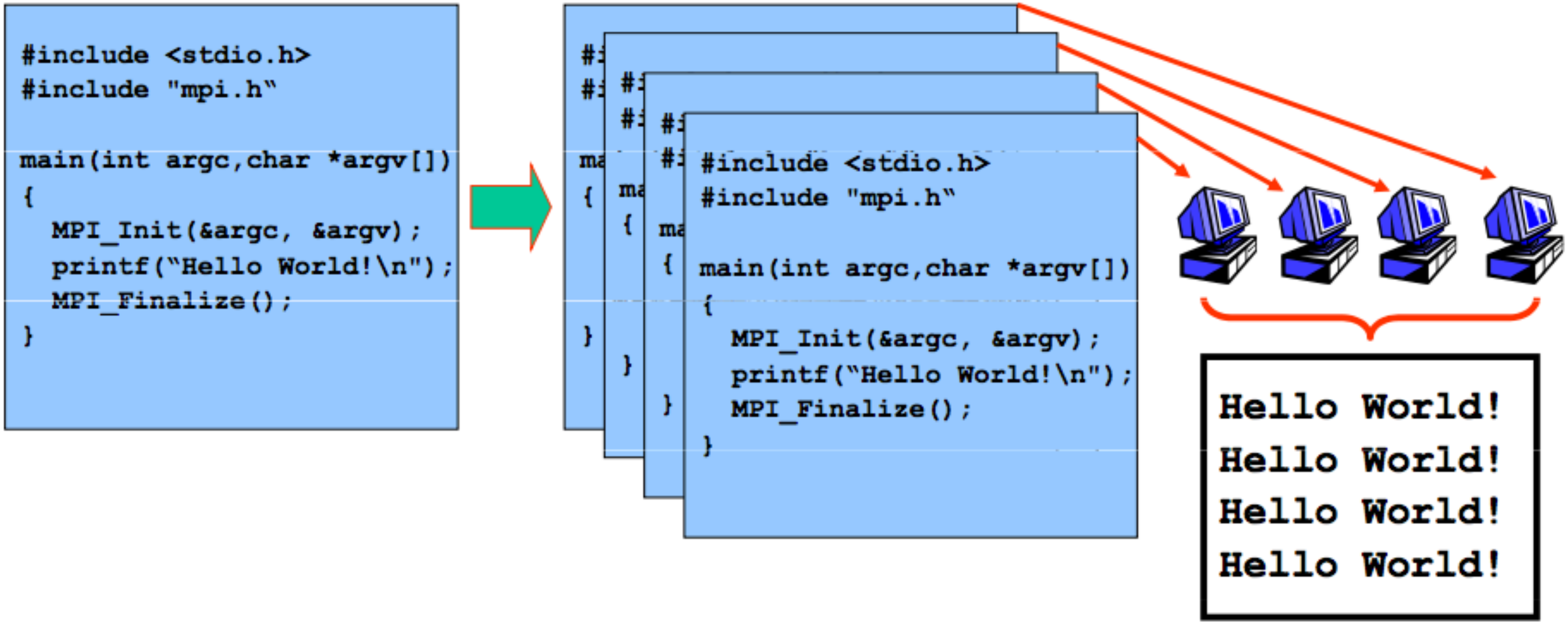
MPI syntax

- #include <mpi.h>
- MPI_Init($argc, $argv);
- MPI_Finalize():
- compile: mpicc file.c
- run: mpirun –np 4 hello or mpiexec \-n 8 a.out
- if run: mpirun hello, it will only run on the head machine
- run on cluster: mpirun --hostfile host myMPI

MPI basic example

```
#include "mpi.h"
#include <stdio.h>
int main( int argc, char *argv[] )
{
    MPI_Init( &argc, &argv );
    printf( "Hello, world!\n" );
    MPI_Finalize();
    return 0;
}
```

How it works



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

main(int argc,char *argv[])
{
  MPI_Init(&argc, &argv);
  printf(“Hello World!\n");
  MPI_Finalize();
}
```

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

main(int argc,char *argv[])
{
  MPI_Init(&argc, &argv);
  printf(“Hello World!\n");
  MPI_Finalize();
}
```

```
Hello World!
Hello World!
Hello World!
Hello World!
```

Six basic functions

- MPI_INIT: init the mpi
- MPI_COMM_SIZE: How many brothers totally I have.
- MPI_COMM_RANK: Who I am
- MPI_SEND: Send message
- MPI_RECV : Receive message
- MPI_FINALIZE: end mpi

MPI basic example

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

MPI basic example

- MPI_SEND (start, count, datatype, dest, tag, comm)
- MPI_RECV(start, count, datatype, source, tag, comm, status)
- Blocking and no-Blocking:
  - eg: MPI_SEND() / MPI_Isend()

MPI send & recv

```
#define TAG_PI 100
double pi = 3.1415926535;
MPI_Send(&pi, 1, MPI_DOUBLE, 0, TAG_PI, MPI_COMM_WORLD);


double num;
MPI_Status status;
MPI_Recv(&num, 1, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG,
         MPI_COMM_WORLD, &status);
```

How to run OpenMP with MPI

- Design it, and write the MPI code
- Add openMP directives
- Compile: mpicc -fopenmp file.c
- Run: mpiexec \-n 8 a.out

Reference

- CITS3402: http://teaching.csse.uwa.edu.au/units/CITS3402/
- OpenMP: https://computing.llnl.gov/tutorials/openMP/
- MPI: https://www.open-mpi.org/doc/current/
- Stack Overflow: http://stackoverflow.com/
- Shanghai Supercenter: http://www.ssc.net.cn/