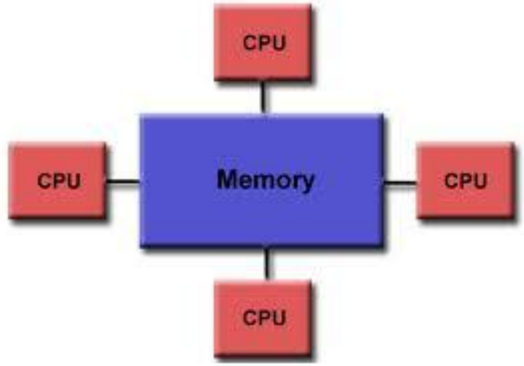
Parallel Computing

OpenMP and MPI

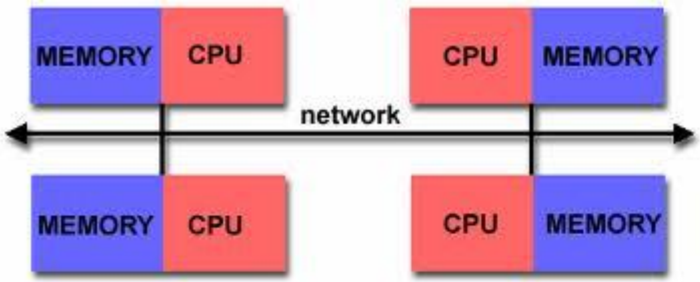
OpenMP

- API for shared memory programming
- Program the threads
- Supported by C/C++ and Fortran



MPI

- API for distributed memory programming
- Program the processes
- Works on shared memory parallel computers as well
- Used from C/C++, Fortran, Python, R etc



OpenMP

- Generally used for loop parallelization

```
const int n = 10000;
double x[n], y[n], a;
int i;

for (i = 0; i < n; i++) {
    y[i] = a*x[i] + y[i]
}
```

g++ main.cpp

```
const int n = 10000;
double x[n], y[n], a;
int i;

#pragma opm parallel for
for (i = 0; i < n; i++) {
    y[i] = a*x[i] + y[i]
}
```

g++ main.cpp -fopenmp

Compiler Directive
In C/C++ for OpenMP

Directive_name

Fork

Default Barrier and Join

- ‘i’ is **private** variable by default; ‘a’, ‘y’ and ‘x’ are **shared**

OpenMP

- Another way to parallelize a loop

By default only outer loop variable is *private.* In order to make any other variable private/shared among different threads it has to be specified explicitly.

Major part of OpenMP programming is deciding what would be shared and what would not be.

```
#pragma opm parallel
{
  #pragma opm for private(i)
  {
    for (i = 0; i < n; i++) {
      …
    }
  }
}
```

Clause

- Syntax

Directives: parallel; for/sections/single; parallel for; barrier/critical/atomic/ordered

Clauses: shared/private; schedule; nowait; if; reduction; num_threads …

```
#include <omp.h>
..
// Parallel Region
#pragma opm directive_name [Clauses…]
{
  …
} // end of parallel region
```

MPI

- Every processor runs the same code!
- Only considers process communication; no control over mapping processes to CPUs
- Communicator
  - Processes are numbered 0, 1, … to N-1
  - Default communicator (MPI_COMM_WORLD) contains all processes
  - Query functions
    - MPI_Comm_size(MPI_COMM_WORLD, nproc): gets the number of processes
    - MPI_Comm_rank(MPI_COMM_WORLD, rank): gets the process ID (rank)

```
#include “mpi.h”
#include <stdio.h>
main (int argc, char* argv[])
{
   int np, pid;
   MPI_Init(&argc, &argv);              // Initializes MPI

   MPI_Comm_size(MPI_COMM_WORLD, &np);
   MPI_Comm_size(MPI_COMM_WORLD, &pid);
   printf(“# Proc = %d, Proc ID = %d”, np, pid);

   MPI_Finalize();          // Clean Up
}
```

Compile: mpicxx main.cpp

Execute: mpiexec –n <num of proc> a.out

MPI

- MPI_Send(sendbuf, cnt, MPI_INT, des, tag, comm)

Starting address of send buffer | # Elems | Data Type | ID of dest proc | Message Tag | Communicator

- MPI_Recv(recvbuf, cnt, MPI_INT, src, tag, comm, &stat)

&stat: Status object

```
…
MPI_Comm_rank(comm, &rank);

if (rank == 0) {
   MPI_Send(sendbuf, cnt, MPI_INT, 1, 0, MPI_COMM_WORLD);
   MPI_Recv(recvbuf, cnt, MPI_INT, 1, MPI_ANY_TAG, MPI_COMM_WORLD, &stat);
}
else {          // Rank = 1
   MPI_Send(sendbuf, cnt, MPI_INT, 0, 0, MPI_COMM_WORLD);
   MPI_Recv(recvbuf, cnt, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &stat);
}
```

Comparison

- Pros of OpenMP
  - easier to program and debug than MPI
  - directives can be added incrementally - gradual parallelization
  - can still run the program as a serial code
  - serial code statements usually don't need modification
  - code is easier to understand and maybe more easily maintained
  - no need to install additional libraries, supported by compiler
- Cons of OpenMP
  - can only be run in shared memory computers (shared memory programming)
  - mostly used for loop parallelization
- Pros of MPI
  - runs on either shared or distributed memory architectures (distributed memory programming)
  - can be used on a wider range of problems than OpenMP
  - each process has its own local variables
  - distributed memory computers are less expensive than large shared memory computers
- Cons of MPI
  - requires more programming changes to go from serial to parallel version
  - can be harder to debug
  - performance is limited by the communication network between the nodes
- Source: http://www.dartmouth.edu/~rc/classes/intro_mpi/parallel_prog_compare.html

Resources

- OpenMP
  - www.openmp.org
- MPI
  - OpenMPI: www.open-mpi.org
  - MPICH2: www.mcs.anl.gov/research/projects/mpich2
  - Download – configure – make – make install