



Introduction to Distributed Memory Programming

University of Wyoming
Advanced Research Computing Center (ARCC)

arcc-info@uwyo.edu

(307) 766-7748

<https://arcc.uwyo.edu/>



Distributed Parallel Programming

- Problems no longer fit onto a single compute node in time and/or space
- Want a faster solution to the problem attempting to be solved
- Choices are becoming more popular today:
 - Message Passing Interface (MPI)
 - Partitioned Global Address Space (PGAS languages / Libraries)
 - Map/Reduce Workloads & Analytics (Hadoop, Spark, etc.)



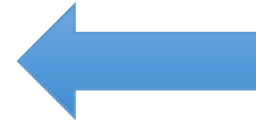
Levels of Parallel Execution

- **There are several levels that can experience parallel execution**
- **Distributed Memory Parallelism**
 - **Message Passing Interface**
 - **OpenSHMEM**
 - **Others?**
- **Shared Memory Parallelism**
 - **Threading**
 - **Multiprocessing**
- **Chip Level Parallelism**
 - **Vector units**
 - **Pipelining**



Brief Overview of MPI

1. Use multiple nodes and/or cores with separate process spacing
2. Launch same application on every node
3. Use different data on each process
 - a. Divide and conquer approach
 - b. Parameterization studies with analysis
4. Communicate only where necessary
 - a. Synchronization
 - b. Data exchange



**Single Program,
Multiple Data:**

SPMD



Remember the “Fabric”?

- MPI will generally want to use the fastest mode of data transport that is available
- High performance Fabric / Interconnect
 - InfiniBand
 - Ethernet
 - TrueScale/Omnipath
 - Others
- These networks are generally tuned by the system administrators to perform well for MPI workloads within a compute cluster
- Key characteristics of MPI fabric:
 - very low latency
 - very high bandwidth



MPI Basics

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

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

```
int main(int argc, char* argv[]) {
```

```
    int error;
```

```
    int thread_req = MPI_THREAD_SINGLE;
```

```
    int thread_prov;
```

```
    error = MPI_Init_thread( &argc, &argv, thread_req, &thread_prov);
```

```
    if (MPI_SUCCESS != error)
```

```
        MPI_Abort(MPI_COMM_WORLD, error );
```

```
    MPI_Finalize();
```

```
    return 0;
```

```
}
```



Hello World

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

```
int main(int argc, char* argv[]) {
```

```
    int error;
    int thread_req = MPI_THREAD_SINGLE;
    int thread_prov;
```

```
    int rank, nprocs;
```

```
    MPI_Init_thread(&argc, &argv, thread_req, &thread_prov);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
    printf("Hello from rank %d out of %d total processes!\n", rank, nprocs);
```

```
    MPI_Finalize()
```

```
    return 0;
```

```
}
```

Running MPI Applications:

```
mpirun
mpiexec
srun
```



Compiling MPI applications

There are wrapper scripts for compiling MPI Applications:

mpicc
mpicxx,mpic++
mpif77,mpif90,mpifort

C Programming
C++ Programming
Fortran Programming

For the test previous test case:

```
$ mpicc -o my_first_mpi hello_mpi.c
```

```
$ mpirun -n 4 my_first_mpi
```




Mount Moran

Slurm Workload Manager & Job Scheduler

- MPI is directly integrated
- Works with placement optimization library
- use 'srun' instead of 'mpirun' or 'mpiexec'



Work on codes on Mt. Moran

Ask questions if needed ☺



Introduction to Distributed Memory Computing

University of Wyoming
Advanced Research Computing Center (ARCC)

arcc-info@uwyo.edu

(307) 766-7748

<https://arcc.uwyo.edu/>