

# Cluster Computing

Weijie Zhao

09/15/2022

# Message Passing Interface (MPI)

- Introduced in early 90's
- Each process may have multiple threads
- Each process has its own address space
- Inter-process communication

# MPI Example

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

# MPI Example

```
#include <stdio.h>
#include <mpi.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("hello world from %d of %d!\n", rank, size);
    MPI_Finalize();
    return 0;
}
```

# MPI Communications

```
int MPI_Send(  
    void* data,  
    int count,  
    MPI_Datatype datatype,  
    int destination,  
    int tag,  
    MPI_Comm communicator)
```

```
int MPI_Recv(  
    void* data,  
    int count,  
    MPI_Datatype datatype,  
    int source,  
    int tag,  
    MPI_Comm communicator,  
    MPI_Status* status)
```

# MPI Communications

```
int MPI_Probe(  
    int source,  
    int tag,  
    MPI_Comm comm,  
    MPI_Status* status)
```

```
int MPI_Get_count(  
    MPI_Status* status,  
    MPI_Datatype datatype,  
    int* count)
```

# MPI Communications

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

# MPI Communications

```
int MPI_Wait(  
    MPI_Request *request,  
    MPI_Status *status)
```

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



# Communicator

```
int MPI_Comm_split(  
    MPI_Comm comm,  
    int color,  
    int key,  
    MPI_Comm * newcomm)
```

```
int MPI_Comm_free(MPI_Comm *comm)
```

# Compilation and Execution

- MPICH, OpenMPI
- mpicc, mpiCC, mpic++
- mpiexec, mpirun
- mpiexec -np 4 ./a.out
- mpiCC a.cc --showme
  
- SLURM
  - sbatch
  - srun

# MPI Configuration

- For each node, create a user that can ssh to all other nodes
- Install MPICH/OpenMPI
- `mpirun -np 4 --hostfile myhost_file ./a.out`
  - `node1 slots=2 max_slots=10`
  - `node2 slots=2 max_slots=10`
- `mpirun -np 4 --hostfile myhost_file --byslot ./a.out`
- `mpirun -np 4 --hostfile myhost_file --bynode ./a.out`

# MPI Collective Communications

- MPI\_Barrier(MPI\_Comm communicator)
- MPI\_Bcast(void\* data, int count, MPI\_Datatype datatype, int root, MPI\_Comm communicator)
- MPI\_Reduce(const void \*sendbuf, void \*recvbuf, int count, MPI\_Datatype datatype, MPI\_Op op, int root, MPI\_Comm comm)
  - MPI\_MIN, MPI\_MAX, MPI\_MINLOC, MPI\_MAXLOC, MPI\_BOR, MPI\_BXOR, MPI\_LOR, MPI\_LXOR, MPI\_BAND, MPI\_LAND, MPI\_SUM and MPI\_PROD
- MPI\_Allreduce(const void\* send\_buffer, void\* receive\_buffer, int count, MPI\_Datatype datatype, MPI\_Op operation, MPI\_Comm communicator)

# Cluster Computing

- MPI
  - Inter-node communication
  - High-performance computing
- Node failure
  - Broken hardware
  - Software bugs
  - Insufficient resources
- Node failure happens commonly for clusters with 1,000+ nodes
  - $(1 - p)^{1000}$

# Cluster Computing

- MPI
  - Inter-node communication
  - High-performance computing
- Node failure
  - Broken hardware
  - Software bugs
  - Insufficient resources
- Node failure happens commonly for clusters with 1,000+ nodes
  - $(1 - p)^{1000}$

We need a system to  
handle these failures!

# Distributed File System

- Decouple data and computing resources
- Replication to take care of node/disk failures
- HDFS
  - Name node
  - Data node

# Common Data Analysis Tasks

- Given a large data, find some statistics
- Given a page view log, find the number of users
- Given a page view log, find the number of users group by browser
- Given a page view log, find the number of users from NY state group by browser



# Spark

- Spark
- Resilient Distributed Dataset (RDD)
  - Immutable
  - Transformations
    - map
    - filter
    - reduceByKey
    - join
    - ...
  - Actions
    - count
    - collect
    - ...