Parallel Machine Learning and Artificial Intelligence

Dr. Handan Liu

h.liu@northeastern.edu

Northeastern University

Parallel Implementations
-- MPI Programming

What is MPI?

- An Interface Specification:
  - M P I = Message Passing Interface
  - MPI is a specification for the developers and users of message passing libraries. By itself, it is NOT a library - but rather the specification of what such a library should be.
  - MPI primarily addresses the message-passing parallel programming model: data is moved from the address space of one process to that of another process through cooperative operations on each process.

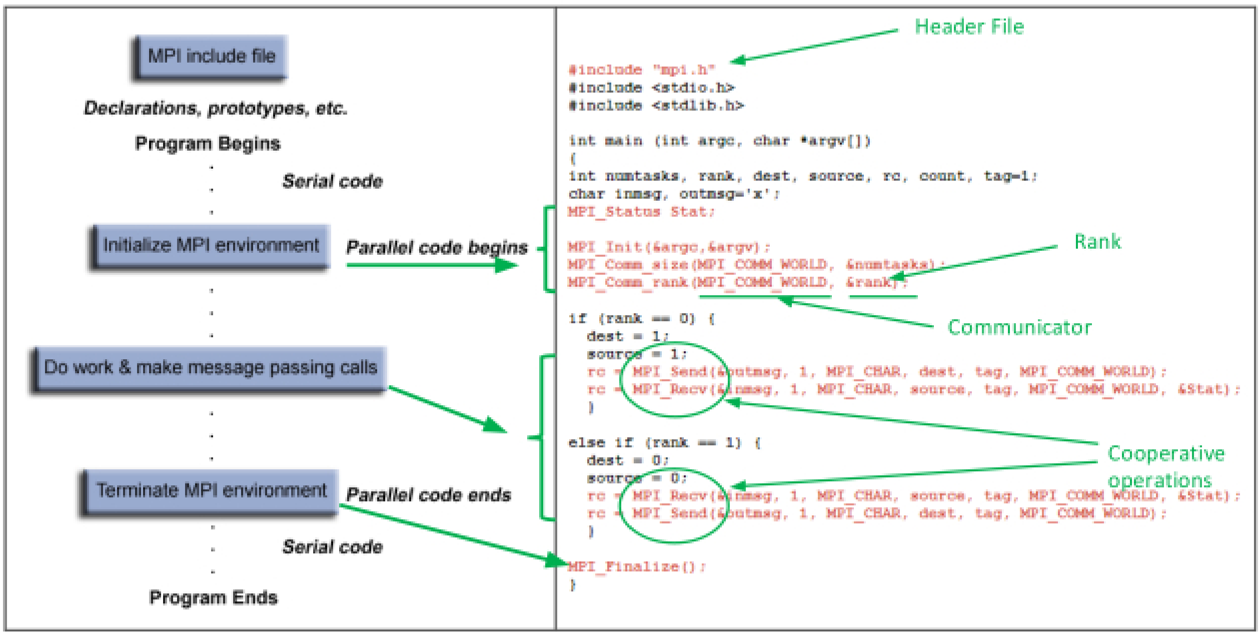
What is MPI?

- Programming Model:
  - Today, MPI runs on virtually any hardware platform:
    - ✔ Distributed Memory
    - ✔ Shared Memory
    - ✔ Hybrid
  - The programming model clearly remains a distributed memory model however, regardless of the underlying physical architecture of the machine.

MPI Implementations and Compilers

MPI Library	Where?	Compilers
MVAPICH	Linux clusters	GNU, Intel, PGI, Clang
Open MPI	Linux clusters	GNU, Intel, PGI, Clang
Intel MPI	Linux clusters	Intel, GNU
IBM Spectrum MPI	Coral Early Access and Sierra clusters	IBM, GNU, PGI, Clang

General MPI Program Structure



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

int main (int argc, char *argv[])
{
int numtasks, rank, dest, source, rc, count, tag=1;
char inmsg, outmsg='x';
MPI_Status Stat;

MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

if (rank == 0) {
  dest = 1;
  source = 1;
  rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
  }

else if (rank == 1) {
  dest = 0;
  source = 0;
  rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
  rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  }

MPI_Finalize();
}
```

Environment Management Routines

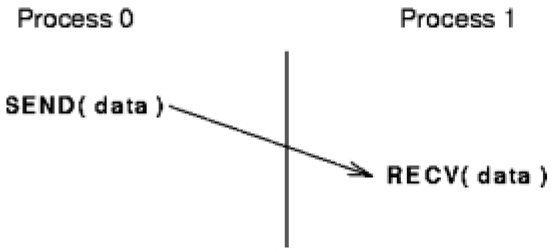
- MPI_Init(): must be called, once, and before others.
- MPI_Comm_size()
- MPI_Comm_rank()
- MPI_Get_processor_name()
- MPI_Get_version()
- MPI_Wtime()
- MPI_Finalize()
- Example: Hello for MPI parallel implementation

MPI Communication

- MPI is a communication protocol for programming parallel computers, and supports both:
  - Point to Point Communication
  - Collective Communication

Point-to-Point Communication

- MPI point-to-point operations typically involve message passing between two, and only two, different MPI tasks.
- One task is performing a send operation and the other task is performing a matching receive operation.
    - a popular example is the pair of MPI_Send and MPI_Recv. ⍰ MPI Cooperative operations



Collective Communication

- Collective functions involve communication among all processes in a process group.

- Types of Collective Communication
  - Synchronization
    - ✔ Blocks until all processes have reached a synchronization point
  - Data Movement (or Global Communication)
    - ✔ Broadcast, Scatters, Gather, All to All transmission of data across the communicator.
  - Collective Computation (or Global Reduction)
    - ✔ One process from the communicator collects data from each process and performs an operation (min, max, add, multiply, etc.) on that data to compute a result.

Collective Communication Routines (10)

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Redue
- MPI_Allreduce
- MPI_Redue_Scatter
- MPI_Alltoall
- MPI_Scan

Collective Communication Routines

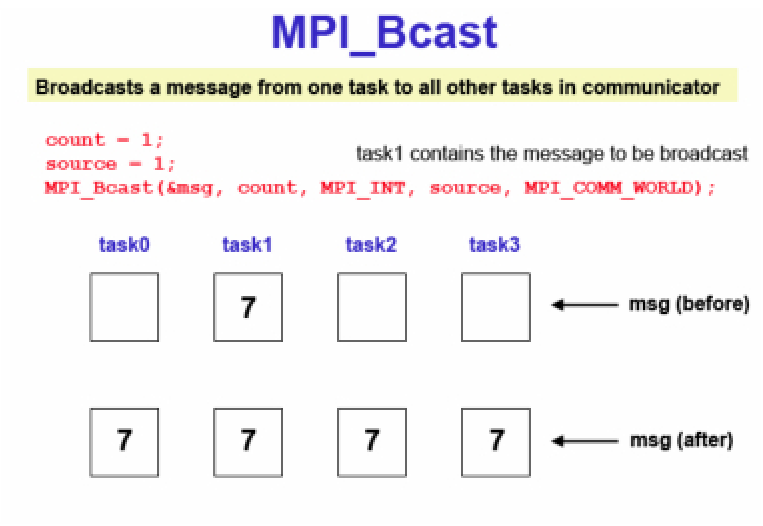
- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

- Synchronization operation.
- Creates a barrier synchronization in a group: MPI_Barrier (comm)
- Each task, when reaching the MPI_Barrier call, blocks until all tasks in the group reach the same MPI_Barrier call. Then all tasks are free to proceed.

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Bcast (&buffer,count,datatype,root,comm)
```



MPI_Bcast

Broadcasts a message from one task to all other tasks in communicator

```
count = 1;
source = 1;
MPI_Bcast(&msg, count, MPI_INT, source, MPI_COMM_WORLD);
```

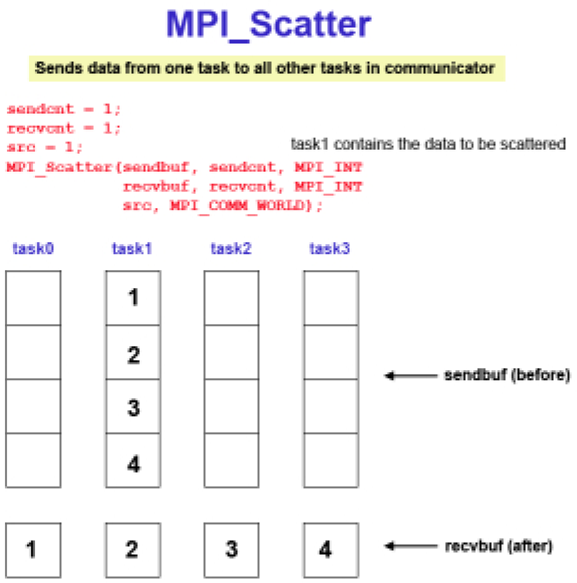
task1 contains the message to be broadcast

Northeastern University
College of Engineering

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Scatter (&sendbuf, sendcnt, sendtype,
             &recvbuf, recvcnt,recvtype,root,comm)
```



MPI_Scatter

Sends data from one task to all other tasks in communicator

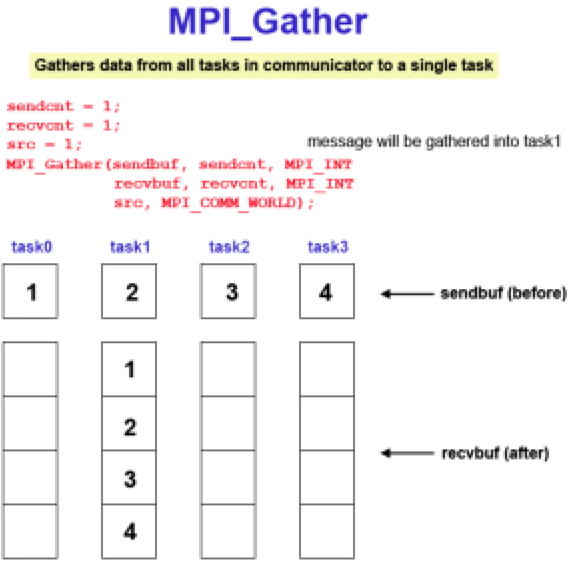
```
sendcnt = 1;
recvcnt = 1;
src = 1;
MPI_Scatter(sendbuf, sendcnt, MPI_INT
            recvbuf, recvcnt, MPI_INT
            src, MPI_COMM_WORLD);
```

task1 contains the data to be scattered

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Gather (&sendbuf, sendcnt, sendtype,
            &recvbuf,recvcount,recvtype,root,comm)
```



MPI_Gather

Gathers data from all tasks in communicator to a single task

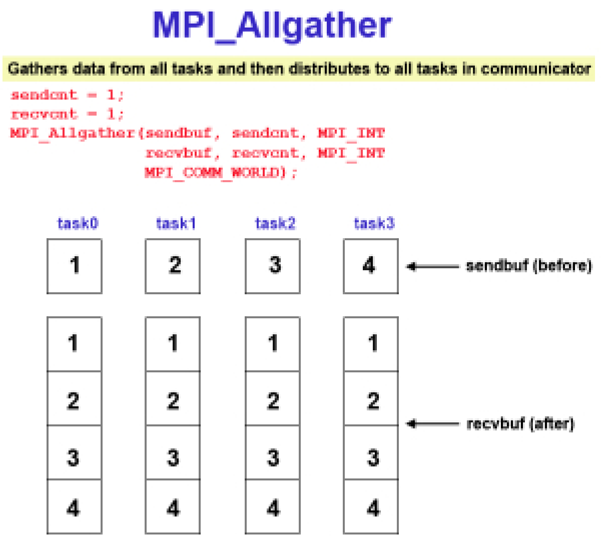
```
sendcnt = 1;
recvcnt = 1;
src = 1;
MPI_Gather(sendbuf, sendcnt, MPI_INT
           recvbuf, recvcnt, MPI_INT
           src, MPI_COMM_WORLD);
```

message will be gathered into task1

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Allgather (&sendbuf, sendcount, sendtype,
               &recvbuf,recvcount,recvtype,comm)
```



MPI_Allgather

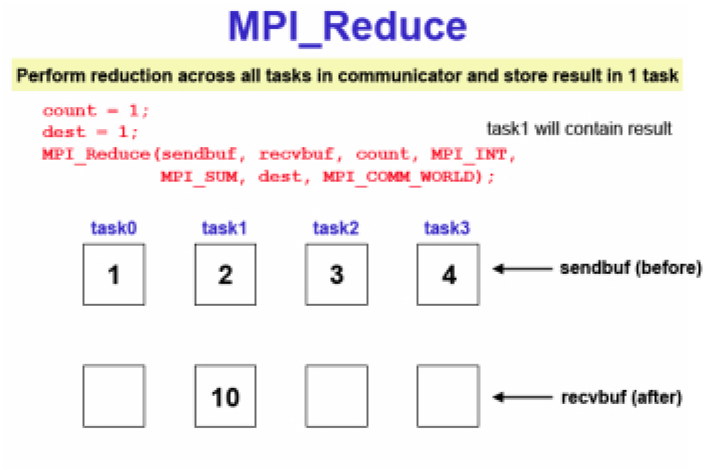
Gathers data from all tasks and then distributes to all tasks in communicator

```
sendcnt = 1;
recvcnt = 1;
MPI_Allgather(sendbuf, sendcnt, MPI_INT
              recvbuf, recvcnt, MPI_INT
              MPI_COMM_WORLD);
```

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Reduce (&sendbuf, &recvbuf, count, datatype,
            op, root,comm)
```



MPI_Reduce

Perform reduction across all tasks in communicator and store result in 1 task

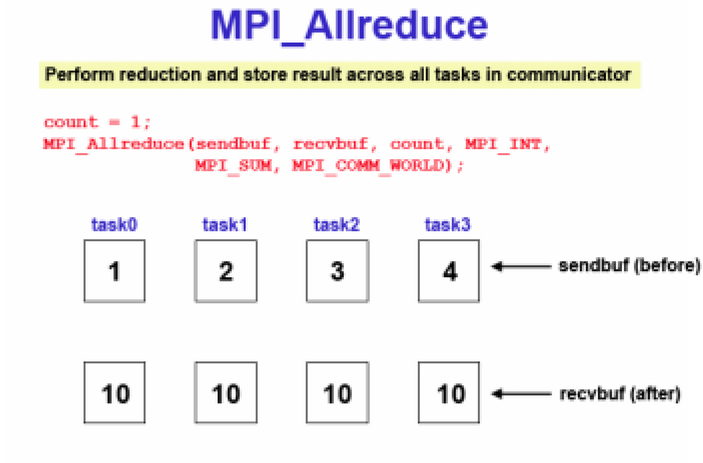
```
count = 1;
dest = 1;
MPI_Reduce(sendbuf, recvbuf, count, MPI_INT,
           MPI_SUM, dest, MPI_COMM_WORLD);
```

task1 will contain result

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Allreduce(&sendbuf,&recvbuf,
                        count,datatype,op,comm)
```



MPI_Allreduce

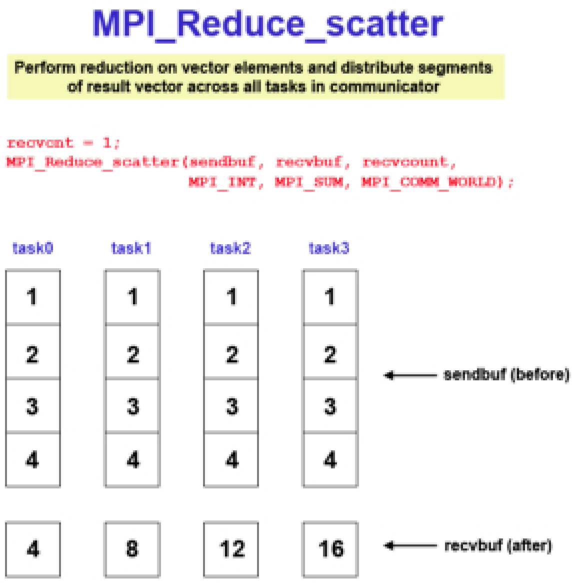
Perform reduction and store result across all tasks in communicator

```
count = 1;
MPI_Allreduce(sendbuf, recvbuf, count, MPI_INT,
              MPI_SUM, MPI_COMM_WORLD);
```

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- **MPI_Reduce_scatter**
- MPI_Alltoall
- MPI_Scan

```
MPI_Reduce_scatter(&sendbuf,&recvbuf,
                   recvcount,datatype,op,comm)
```



MPI_Reduce_scatter

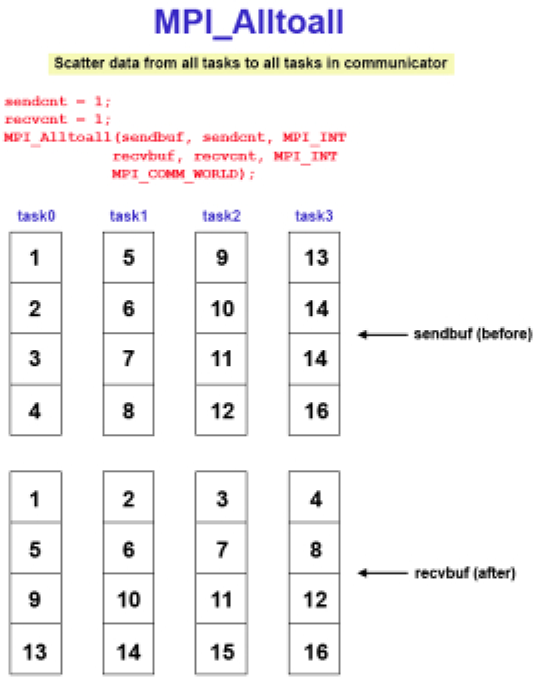
Perform reduction on vector elements and distribute segments of result vector across all tasks in communicator

```
recvcnt = 1;
MPI_Reduce_scatter(sendbuf, recvbuf, recvcount,
                   MPI_INT, MPI_SUM, MPI_COMM_WORLD);
```

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Alltoall(&sendbuf,sendcount,
     sendtype,&recvbuf,
     recvcnt,recvtype,comm)
```



MPI_Alltoall

Scatter data from all tasks to all tasks in communicator

```
sendcnt = 1;
recvcnt = 1;
MPI_Alltoall(sendbuf, sendcnt, MPI_INT
             recvbuf, recvcnt, MPI_INT
             MPI_COMM_WORLD);
```

task0	task1	task2	task3
1	5	9	13
2	6	10	14
3	7	11	14
4	8	12	16

sendbuf (before)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

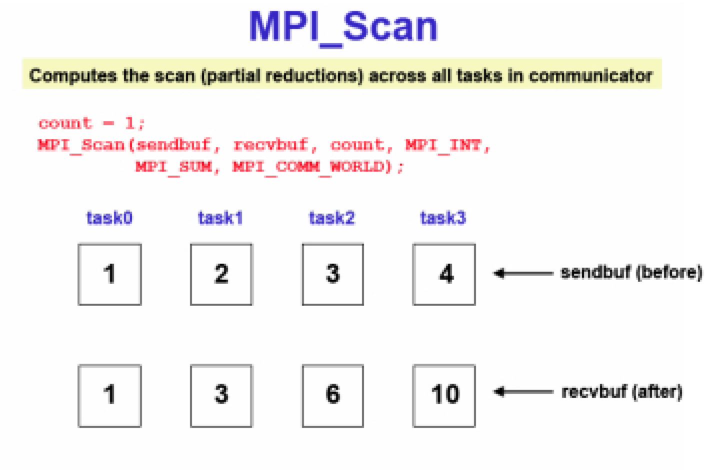
recvbuf (after)

Northeastern University
College of Engineering

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

```
MPI_Scan(&sendbuf,&recvbuf,count,
         datatype,op,comm)
```



MPI_Scan

Computes the scan (partial reductions) across all tasks in communicator

```
count = 1;
MPI_Scan(sendbuf, recvbuf, count, MPI_INT,
         MPI_SUM, MPI_COMM_WORLD);
```

task0	task1	task2	task3	
1	2	3	4	← sendbuf (before)
1	3	6	10	← recvbuf (after)

Collective Communication Routines

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
  MPI_Allreduce
  MPI_Redue_scatter
- MPI_Alltoall
- MPI_Scan

Synchronization

Data movement

Collective computation

Collective computation operation + data movement

Examples:

- Point-to-Point communication by using MPI_Send and MPI_Recv

- Collective Communications by using MPI_Scatter

How to Compile and Run a MPI Program

The table below lists OpenMPI compiler wrapper scripts for Linux clusters.

Language	Script Name	Underlying Compiler
C	**mpicc**	C compiler for loaded compiler package
C++	**mpiCC** **mpic++** **mpicxx**	C++ compiler for loaded compiler package
Fortran	**mpif77**	Fortran77 compiler for loaded compiler package. Points to mpifort.
	mpif90	Fortran90 compiler for loaded compiler package. Points to mpifort.
	mpifort	Fortran 77/90 compiler for loaded compiler package.

Resources

- For more information of MPI:

  Open MPI: Open Source High Performance Computing

  https://www.open-mpi.org

  https://www.open-mpi.org/doc/v4.0/

  MPI Forum

  https://www.mpi-forum.org

Highly Optimized Math Libraries

- Open Source
  - **BLAS**: Basic Linear Algebra Subprograms
  - LAPACK: Linear Algebra PACKage
  - ScaLAPACK: Scalable Linear Algebra PACKage
  - ....
- Commercial
  - Intel's MKL: Intel Math Kernel Library
  - IBM's ESSL: Engineering and Scientific Subroutine Library
  - AMD's AMCL: AMD Core Math Library
  - ....

- Stay safe!
- See you next class!

Next Lecture will Continue:

- Review the Quiz1
- Introduction to Discovery