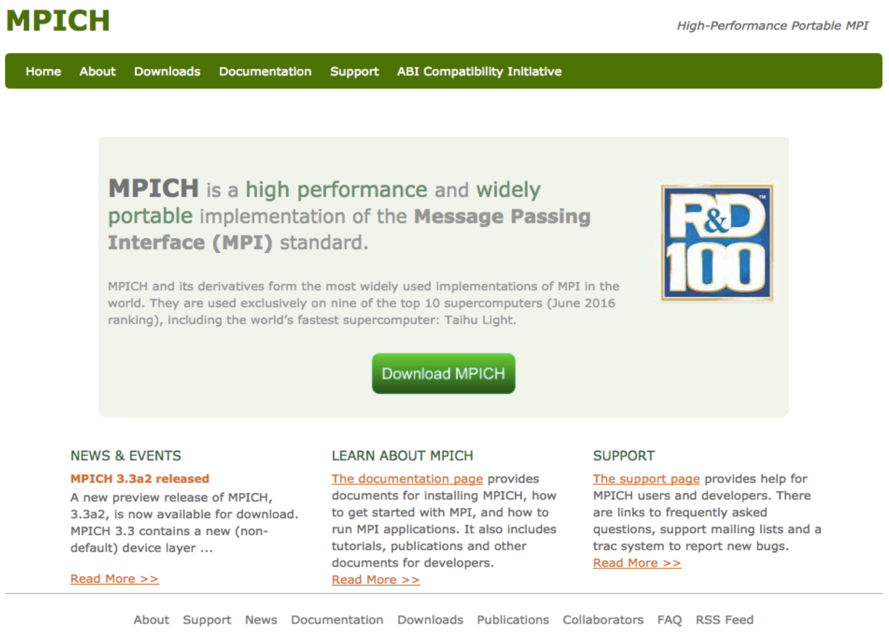
Lab 11
MPI Programming on Linux
- MPICH v2 -

C.H. Youn (Oct. 24, 2017)

MPICH

- MPICH is free software that implements the MPI standard.
- MPICH can run on a variety of platforms including Linux, Mac OS / X, Solaris, and Windows
- MPICH is one of the most popular MPI implementation programs, and many company message-parsing programs are derived from MPICH (for example, IBM MPI, Intel MPI, Cray MPI, and Microsoft MPI for Blue Gene).
- Download:
  https://www.mpich.org/


MPICH
High-Performance Portable MPI
Home About Downloads Documentation Support ABI Compatibility Initiative
R&D 100
Download MPICH
About Support News Documentation Downloads Publications Collaborators FAQ RSS Feed

MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard.

MPICH and its derivatives form the most widely used implementations of MPI in the world. They are used exclusively on nine of the top 10 supercomputers (June 2016 ranking), including the world's fastest supercomputer: Taihu Light.

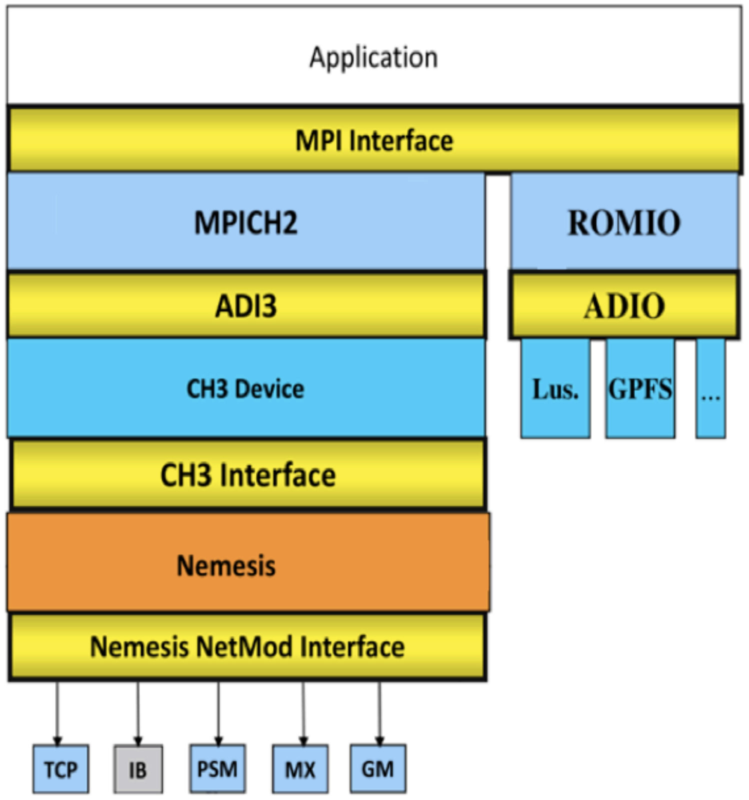
NEWS & EVENTS
MPICH 3.3a2 released
A new preview release of MPICH, 3.3a2, is now available for download. MPICH 3.3 contains a new (non-default) device layer ...
Read More >>

LEARN ABOUT MPICH
The documentation page provides documents for installing MPICH, how to get started with MPI, and how to run MPI applications. It also includes tutorials, publications and other documents for developers.
Read More >>

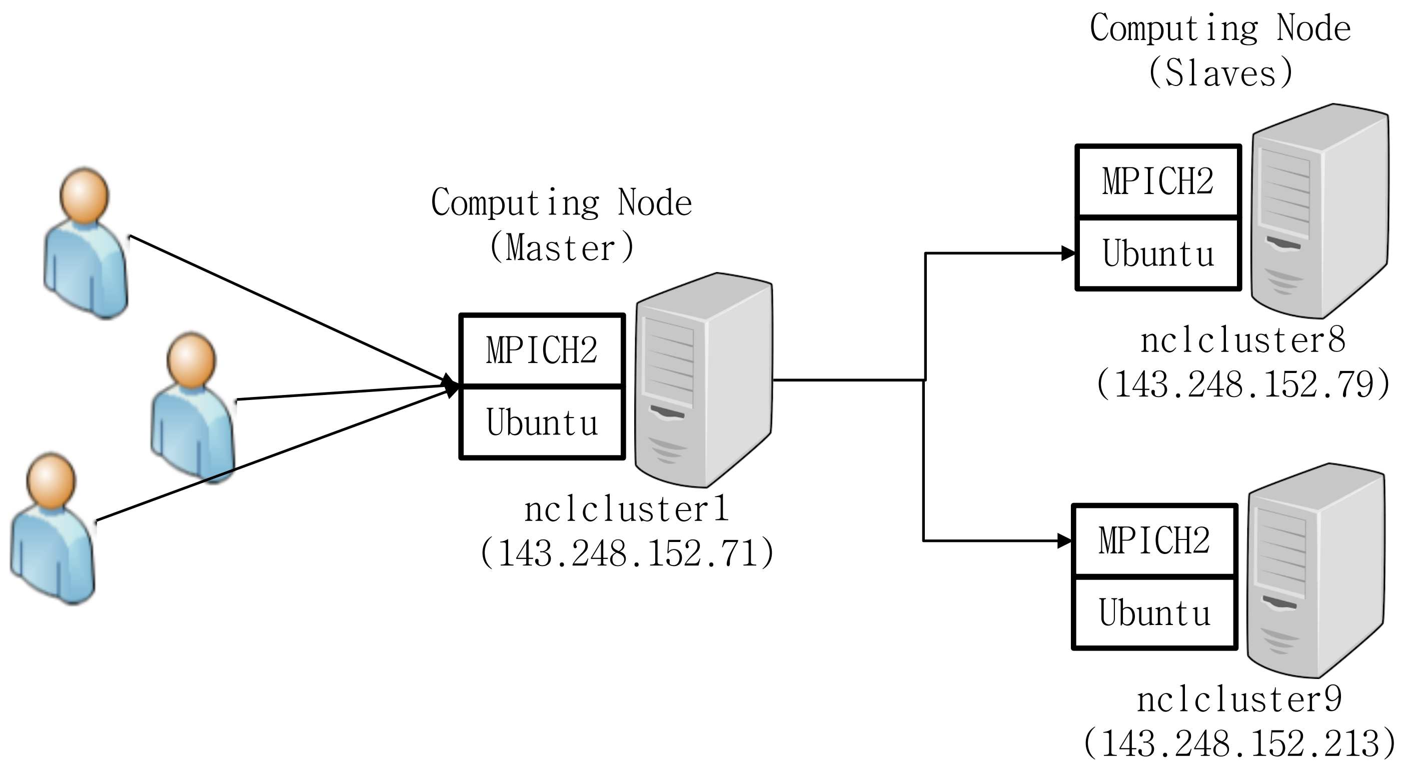
SUPPORT
The support page provides help for MPICH users and developers. There are links to frequently asked questions, support mailing lists and a trac system to report new bugs.
Read More >>

MPICH

- MPICH implements the latest standards of high quality standards MPI.
- There are various derived versions in the basics of the program to meet special needs
- Disadvantages of MPICH
  - Does not support InfiniBand
  - Other derived versions of MPICH (eg Intel MPI) support InfiniBand
  - MPICH currently supports InfiniBand using netmod but is not optimized as in other versions
- The advantage of MPICH is that it supports a wide range of hardware and platforms
- The mpd of the MPICH is often down and difficult to use.
  - Hydra process manager instead of using mpd, is rated as good as OpenMPI's ORTE.



MPI Testbed in NCL



MPI Testbed in NCL

Usage	IP	Hostname	MPICH
User Client Computing Node	143.248.152.71	nclcluster1	MPICH2 (v3.1.4)
Computing Node	143.248.152.79	nclcluster8	MPICH2 (v3.1.4)
Computing Node	143.248.152.213	nclcluster9	MPICH2 (v3.1.4)

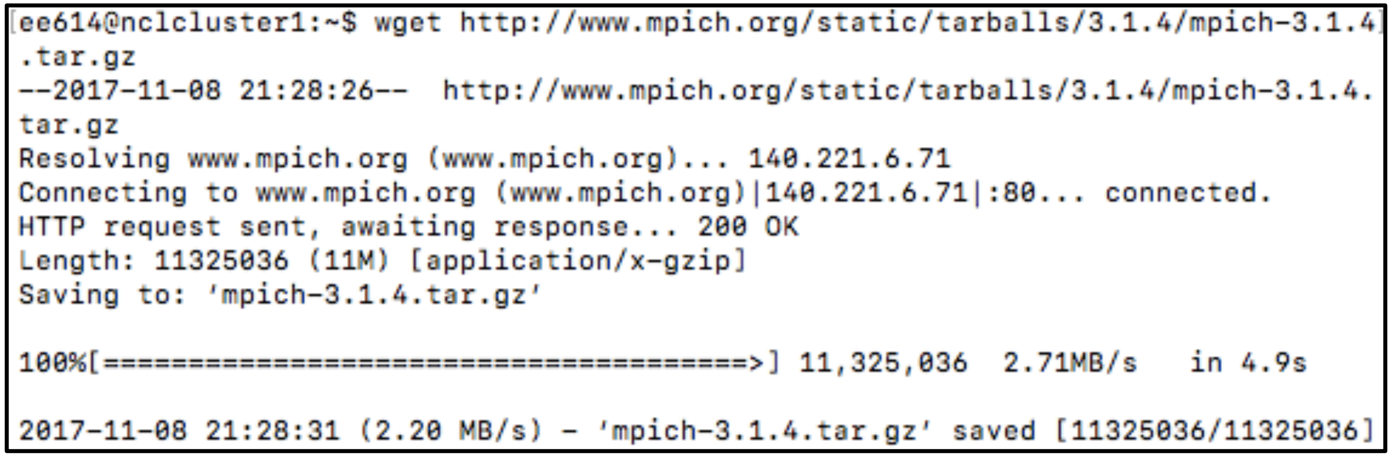
- User Client – your problem solving node - connect to the node as your account
- Computing Node – workers
  - MPICH2 (v3.1.4) is installed

Requirements for MPI

- Install MPICH2
  - Download MPICH2
  - Unpack the tar file
  - Configure MPICH, with specifying the installation directory
  - Build MPICH
  - Install the MPICH commands
  - Set environmental variable ‘PATH’
- Host Setting
  - SSH Authorization Setting (Pub/Sub Key)
  - Change host domain name for machine description

Install MPICH2 (Ubuntu)

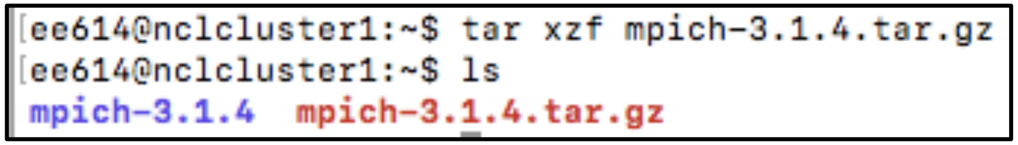
- Step 1-1) Download MPICH source from http://mpich.org
  - $ wget http://www.mpich.org/static/tarballs/3.1.4/mpich-3.1.4.tar.gz



```
ee614@nclcluster1:~$ wget http://www.mpich.org/static/tarballs/3.1.4/mpich-3.1.4
.tar.gz
--2017-11-08 21:28:26--  http://www.mpich.org/static/tarballs/3.1.4/mpich-3.1.4.
tar.gz
Resolving www.mpich.org (www.mpich.org)... 140.221.6.71
Connecting to www.mpich.org (www.mpich.org)|140.221.6.71|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11325036 (11M) [application/x-gzip]
Saving to: 'mpich-3.1.4.tar.gz'

100%[======================================>] 11,325,036  2.71MB/s   in 4.9s

2017-11-08 21:28:31 (2.20 MB/s) - 'mpich-3.1.4.tar.gz' saved [11325036/11325036]
```

Install MPICH2 (Ubuntu)

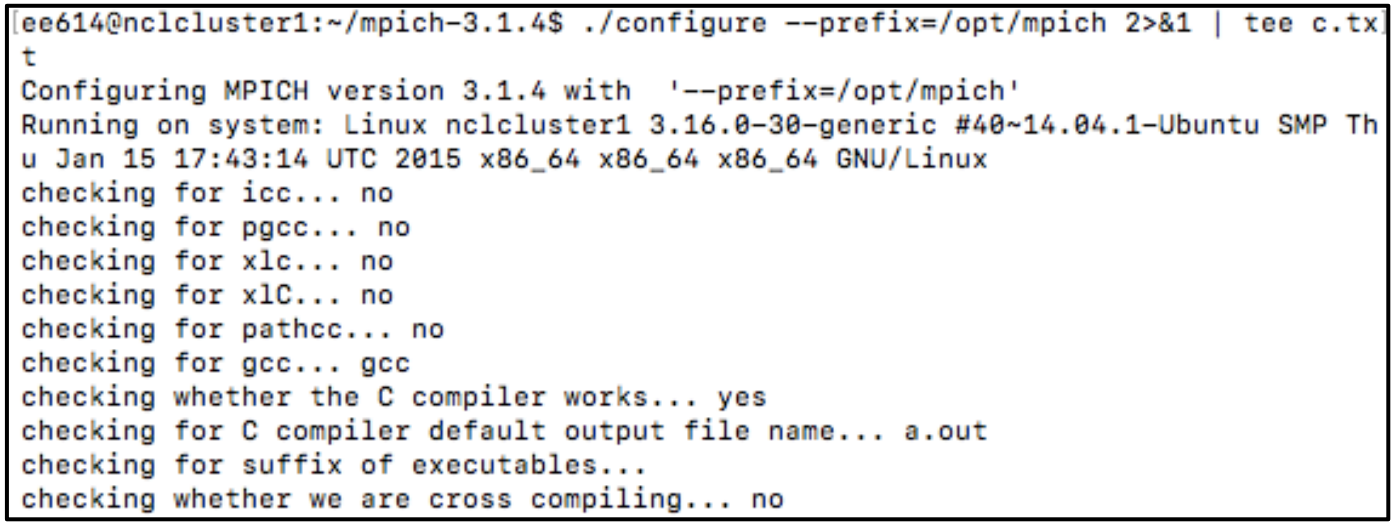
- Step 1-2) Unpack the tar file
  - $ tar xzf mpich-3.1.4.tar.gz
  - Let us assume that the directory where you do this is /home/ee614
  - It will now contain a subdirectory named mpich-3.1.4



```
ee614@nclcluster1:~$ tar xzf mpich-3.1.4.tar.gz
ee614@nclcluster1:~$ ls
mpich-3.1.4  mpich-3.1.4.tar.gz
```

Install MPICH2 (Ubuntu)

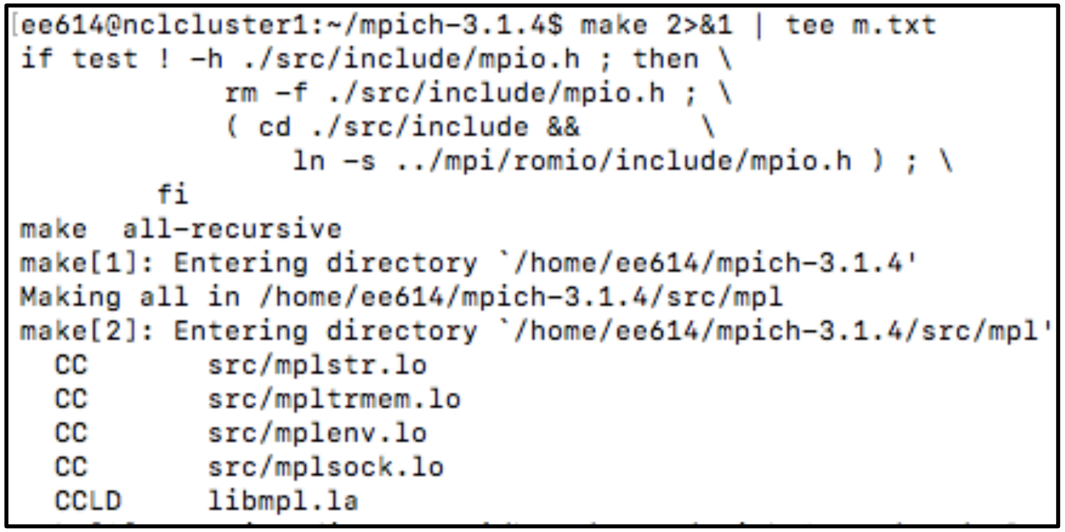
- Step 1-3) Configure MPICH, specifying the installation directory, and running the configure script in the source directory:
  - $ cd mpich-3.1.4
  - $ ./configure --prefix=[specific dir] 2>&1 | tee c.txt
- Other configure options are described in manual. Check the c.txt file to make sure everything went well. The file config.log is created by configure and contains a record of the tests that configure performed.



```
[ee614@nclcluster1:~/mpich-3.1.4$ ./configure --prefix=/opt/mpich 2>&1 | tee c.tx
t
Configuring MPICH version 3.1.4 with  '--prefix=/opt/mpich'
Running on system: Linux nclcluster1 3.16.0-30-generic #40~14.04.1-Ubuntu SMP Th
u Jan 15 17:43:14 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
checking for icc... no
checking for pgcc... no
checking for xlc... no
checking for xlC... no
checking for pathcc... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
```

Install MPICH2 (Ubuntu)

- Step 1-4) Build mpich
  - $ make 2>&1 | tee m.txt

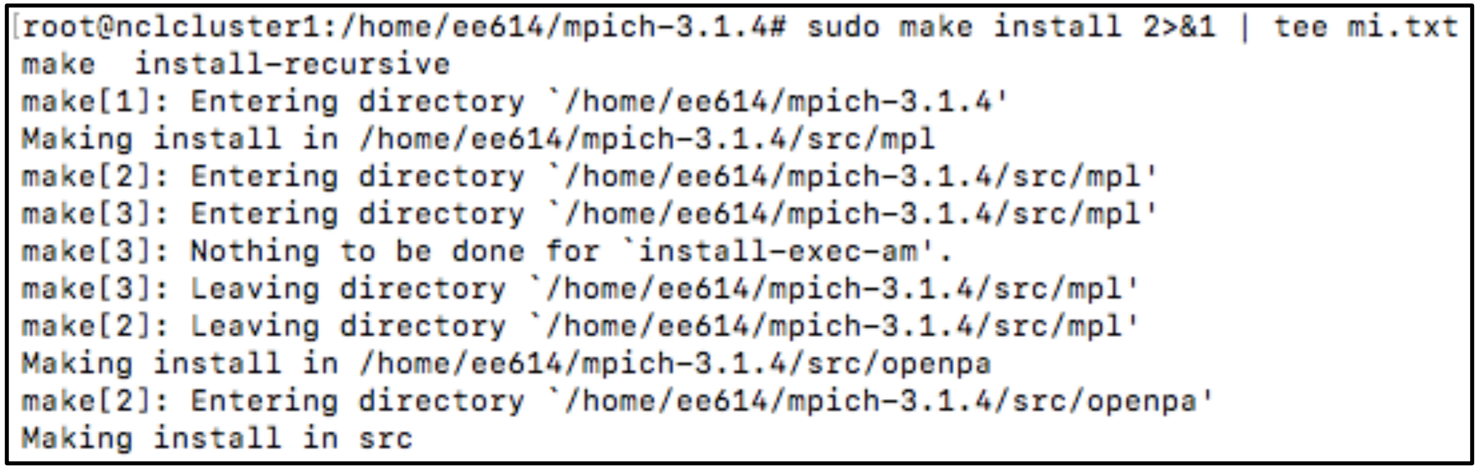


```
ee614@nclcluster1:~/mpich-3.1.4$ make 2>&1 | tee m.txt
if test ! -h ./src/include/mpio.h ; then \
            rm -f ./src/include/mpio.h ; \
            ( cd ./src/include &&       \
                ln -s ../mpi/romio/include/mpio.h ) ; \
        fi
make  all-recursive
make[1]: Entering directory `/home/ee614/mpich-3.1.4'
Making all in /home/ee614/mpich-3.1.4/src/mpl
make[2]: Entering directory `/home/ee614/mpich-3.1.4/src/mpl'
  CC       src/mplstr.lo
  CC       src/mpltrmem.lo
  CC       src/mplenv.lo
  CC       src/mplsock.lo
  CCLD     libmpl.la
```

  - This step should succeed if there were no problems with the preceding step. Check file m.txt.

Install MPICH2 (Ubuntu)

- Step 1-5) Install mpich command
  - $ sudo make install 2>&1 | tee mi.txt

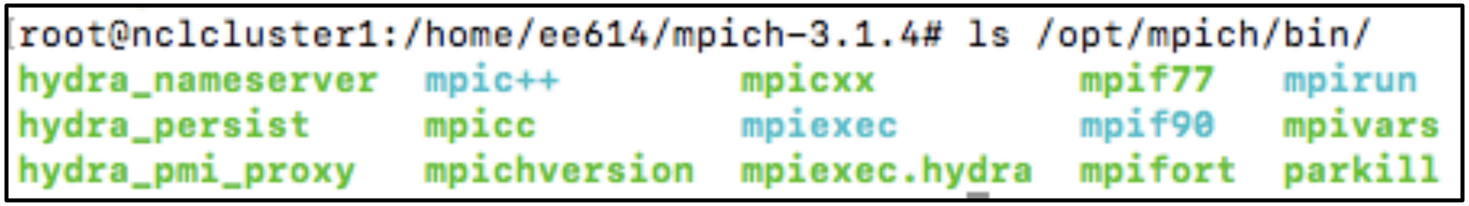


```
root@nclcluster1:/home/ee614/mpich-3.1.4# sudo make install 2>&1 | tee mi.txt
make  install-recursive
make[1]: Entering directory `/home/ee614/mpich-3.1.4'
Making install in /home/ee614/mpich-3.1.4/src/mpl
make[2]: Entering directory `/home/ee614/mpich-3.1.4/src/mpl'
make[3]: Entering directory `/home/ee614/mpich-3.1.4/src/mpl'
make[3]: Nothing to be done for `install-exec-am'.
make[3]: Leaving directory `/home/ee614/mpich-3.1.4/src/mpl'
make[2]: Leaving directory `/home/ee614/mpich-3.1.4/src/mpl'
Making install in /home/ee614/mpich-3.1.4/src/openpa
make[2]: Entering directory `/home/ee614/mpich-3.1.4/src/openpa'
Making install in src
```

  - This step collects all required executables and scripts in the bin subdirectory of the directory specified by the prefix argument to configure

Install MPICH2 (Ubuntu)

- Check installation directory
  - $ ls /opt/mpich/bin

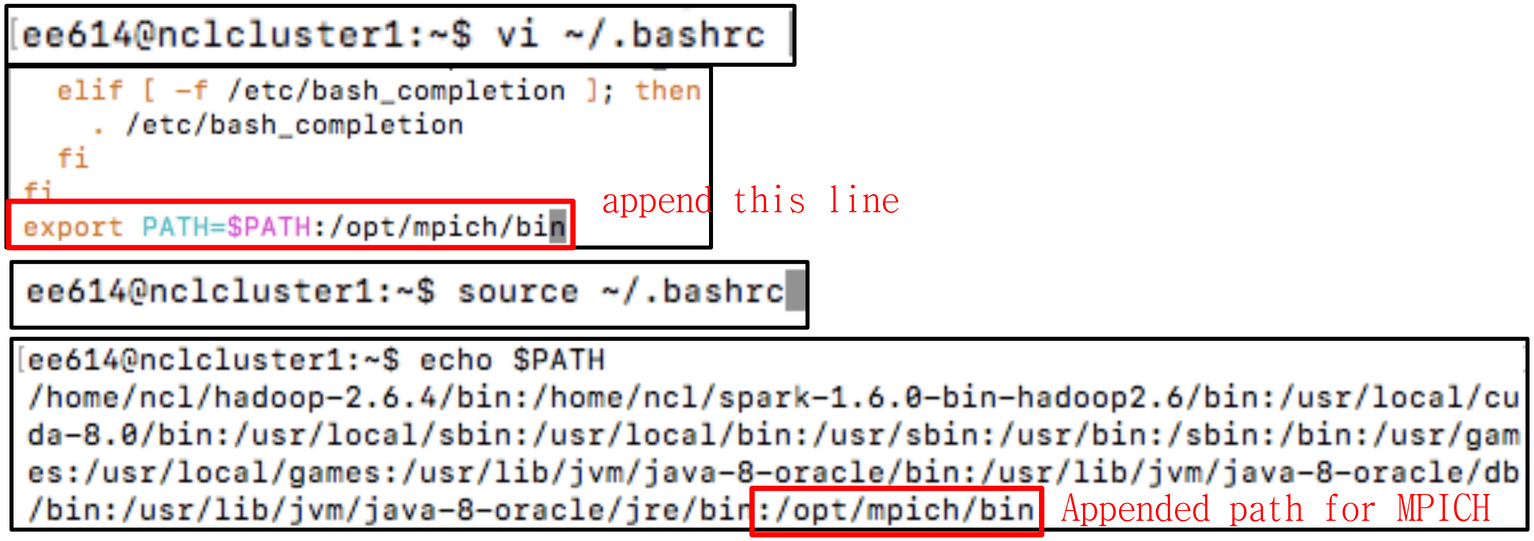


```
root@nclcluster1:/home/ee614/mpich-3.1.4# ls /opt/mpich/bin/
hydra_nameserver  mpic++        mpicxx        mpif77    mpirun
hydra_persist     mpicc         mpiexec       mpif90    mpivars
hydra_pmi_proxy   mpichversion  mpiexec.hydra mpifort   parkill
```

  - MPI Commands are installed
    : mpicc
    : mpiexec

Install MPICH (Ubuntu)

- Step 1-6) Add the bin subdirectory of the installation directory to your path
  - $ vi ~/.bashrc
  - append last line "export PATH=$PATH:/opt/mpich/bin"
  - $ source ~/.bashrc
  - $ echo $PATH



```
ee614@nclcluster1:~$ vi ~/.bashrc
```

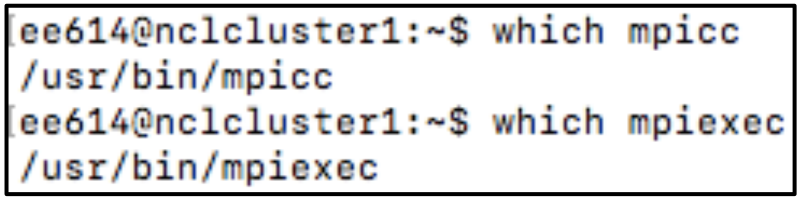
```
    elif [ -f /etc/bash_completion ]; then
      . /etc/bash_completion
    fi
fi
export PATH=$PATH:/opt/mpich/bin
```

```
ee614@nclcluster1:~$ source ~/.bashrc
```

```
ee614@nclcluster1:~$ echo $PATH
/home/ncl/hadoop-2.6.4/bin:/home/ncl/spark-1.6.0-bin-hadoop2.6/bin:/usr/local/cu
da-8.0/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/gam
es:/usr/local/games:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db
/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/opt/mpich/bin
```

Install MPICH2 (Ubuntu)

- Check completion of installation mpich



```
ee614@nclcluster1:~$ which mpicc
/usr/bin/mpicc
ee614@nclcluster1:~$ which mpiexec
/usr/bin/mpiexec
```

  – All should refer to the commands in the bin subdirectory of your install directory. It is at this point that you will need to duplicate this directory or install again on your other machines if it is not in a shared file system.

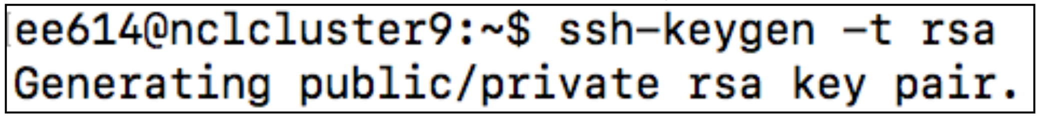
SSH authorized_key setting

- What is SSH key?
  - When you connect to the server, you submit a key instead of a password.
- How SSH Key Works?
  - The SSH key consists of a public key and a private key. Generating a key creates a public key and a private key. And, the private key must be located on the local machine, and the public key must be on the remote machine. (The local machine is the SSH client, and the remote machine is the computer where the SSH Server is installed.)
  - When an SSH connection is attempted, the SSH client compares the private key of the local machine with the private key of the remote machine to see if they match.

SSH authorized_key setting

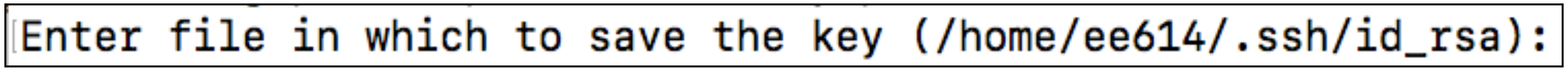
- Procedure for setting SSH key

1) Access to all nodes and type command '$ ssh-keygen -t rsa'



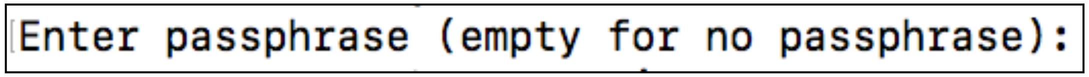
```
ee614@nclcluster9:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
```

2) Continuously input 'enter' until ssh-keygen command completed



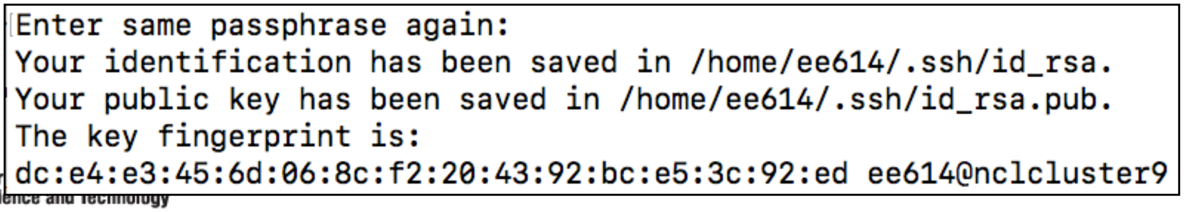
```
Enter file in which to save the key (/home/ee614/.ssh/id_rsa):
```

- Specifies the location to store the SSH key. When you press enter, it is saved in the default path. By default, the SSH Client tries to authenticate using the keys in this directory.



```
Enter passphrase (empty for no passphrase):
```

- Enter the passphrase. The passphrase encrypts the private key with a password. The recommended value is 10 to 30 characters and can be omitted. If omitted, be careful because this part can be a security hole. If you want automatic login, you should just press enter.



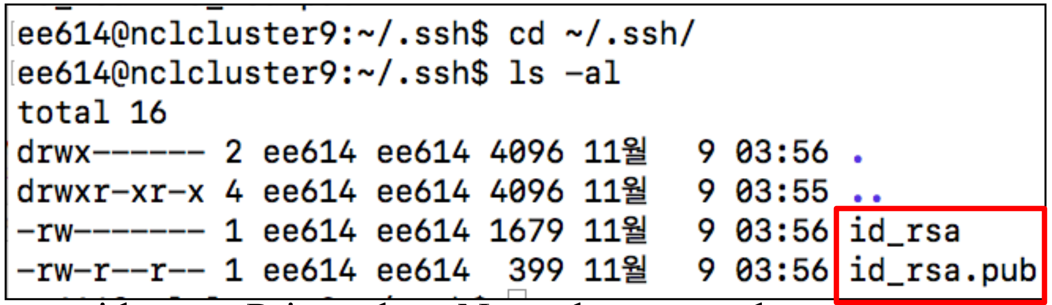
```
Enter same passphrase again:
Your identification has been saved in /home/ee614/.ssh/id_rsa.
Your public key has been saved in /home/ee614/.ssh/id_rsa.pub.
The key fingerprint is:
dc:e4:e3:45:6d:06:8c:f2:20:43:92:bc:e5:3c:92:ed ee614@nclcluster9
```

Done!!
Generating SSH key

SSH authorized_key setting

- Procedure for setting SSH key

3) Move to /.ssh folder and check related files

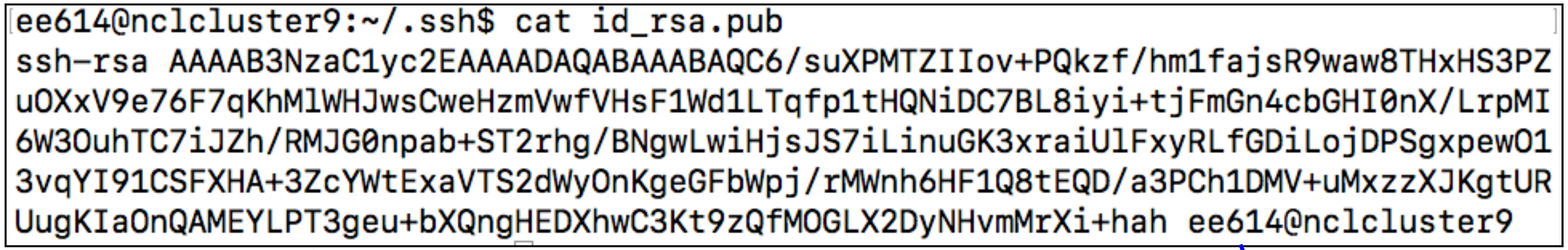


```
ee614@nclcluster9:~/.ssh$ cd ~/.ssh/
ee614@nclcluster9:~/.ssh$ ls -al
total 16
drwx------ 2 ee614 ee614 4096 11월   9 03:56 .
drwxr-xr-x 4 ee614 ee614 4096 11월   9 03:55 ..
-rw------- 1 ee614 ee614 1679 11월   9 03:56 id_rsa
-rw-r--r-- 1 ee614 ee614  399 11월   9 03:56 id_rsa.pub
```

- id_rsa = Private key, Never be exposed to others.
- id_rsa.pub = Public key, Enter the authorized_keys of the remote machine you want to connect to.
- authorized_keys = It is located under the .ssh directory of the remote machine and stores the value of the id_rsa.pub key.

SSH authorized_key setting

- Procedure for setting SSH key
  4) Make new file with vi editor named "authorized_keys" and append authorized_keys file with contents of id_rsa.pub file in each node like the following (include the accessed node)

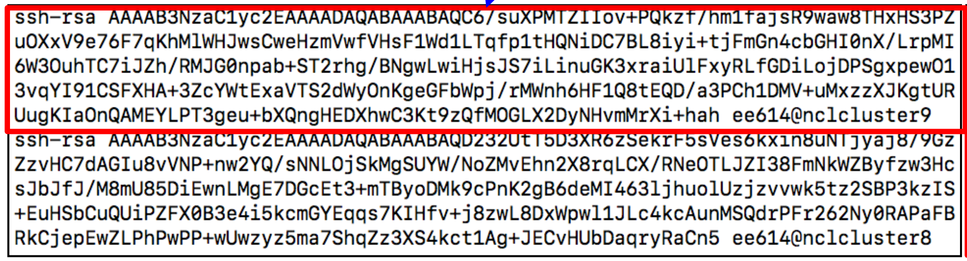


```
ee614@nclcluster9:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6/suXPMTZIIov+PQkzf/hm1fajsR9waw8THxHS3PZ
uOXxV9e76F7qKhMlWHJwsCweHzmVwfVHsF1Wd1LTqfp1tHQNiDC7BL8iyi+tjFmGn4cbGHI0nX/LrpMI
6W3OuhTC7iJZh/RMJG0npab+ST2rhg/BNgwLwiHjsJS7iLinuGK3xraiUlFxyRLfGDiLojDPSgxpew01
3vqYI91CSFXHA+3ZcYWtExaVTS2dWyOnKgeGFbWpj/rMWnh6HF1Q8tEQD/a3PCh1DMV+uMxzzXJKgtUR
UugKIaOnQAMEYLPT3geu+bXQngHEDXhwC3Kt9zQfMOGLX2DyNHvmMrXi+hah ee614@nclcluster9
```

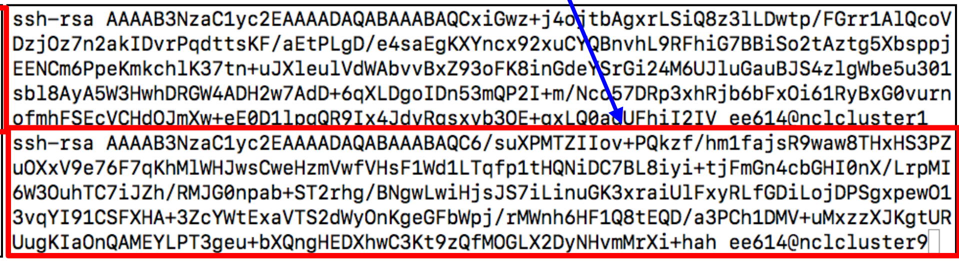
ee614@nclcluster9 → id_rsa.pub

append

append



```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6/suXPMTZIIov+PQkzf/hm1fajsR9waw8THxHS3PZ
uOXxV9e76F7qKhMlWHJwsCweHzmVwfVHsF1Wd1LTqfp1tHQNiDC7BL8iyi+tjFmGn4cbGHI0nX/LrpMI
6W3OuhTC7iJZh/RMJG0npab+ST2rhg/BNgwLwiHjsJS7iLinuGK3xraiUlFxyRLfGDiLojDPSgxpew01
3vqYI91CSFXHA+3ZcYWtExaVTS2dWyOnKgeGFbWpj/rMWnh6HF1Q8tEQD/a3PCh1DMV+uMxzzXJKgtUR
UugKIaOnQAMEYLPT3geu+bXQngHEDXhwC3Kt9zQfMOGLX2DyNHvmMrXi+hah ee614@nclcluster9
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD232UtT5D3XR6zSekrF5sVes6kxIn8uNTjyaj8/9Gz
ZzvHC7dAGIu8vVNP+nw2YQ/sNNLOjSkMgSUYW/NoZMvEhn2X8rqLCX/RNeOTLJZI38FmNkWZByfzw3Hc
sJbJfJ/M8mU85DiEwnLMgE7DGcEt3+mTByoDMk9cPnK2gB6deMI463ljhuolUzjzvvwk5tz2SBP3kzIS
+EuHSbCuQUiPZFX0B3e4i5kcmGYEqqs7KIHfv+j8zwL8DxWpwl1JLc4kcAunMSQdrPFr262Ny0RAPaFB
RkCjepEwZLPhPwPP+wUwzyz5ma7ShqZz3XS4kct1Ag+JECvHUbDaqryRaCn5 ee614@nclcluster8
```



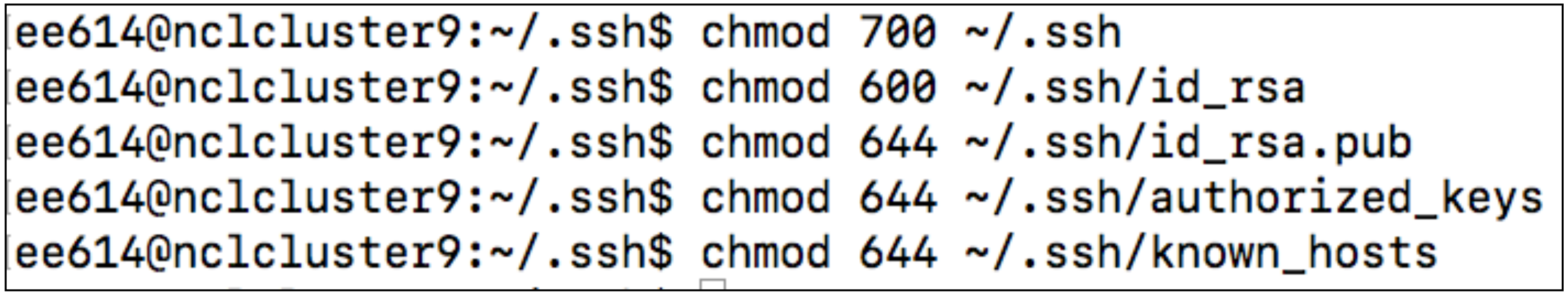
```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxiGwz+j4ojtbAgxrLSiQ8z3lLDwtp/FGrr1AlQcoV
DzjOz7n2akIDvrPqdttsKF/aEtPLgD/e4saEgKXYncx92xuCYQBnvhL9RFhiG7BBiSo2tAztg5Xbsppj
EENCm6PpeKmkchlK37tn+uJXleulVdWAbvvBxZ93oFK8inGdeYSrGi24M6UJluGauBJS4zlgWbe5u301
sbl8AyA5W3HwhDRGW4ADH2w7AdD+6qXLDgoIDn53mQP2I+m/Ncd57DRp3xhRjb6bFxOi61RyBxG0vurn
ofmhFSEcVCHdOJmXw+eE0D1lpqQR9Ix4JdvRqsxyb3OE+qxLQ0aUFhiI2IV ee614@nclcluster1
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6/suXPMTZIIov+PQkzf/hm1fajsR9waw8THxHS3PZ
uOXxV9e76F7qKhMlWHJwsCweHzmVwfVHsF1Wd1LTqfp1tHQNiDC7BL8iyi+tjFmGn4cbGHI0nX/LrpMI
6W3OuhTC7iJZh/RMJG0npab+ST2rhg/BNgwLwiHjsJS7iLinuGK3xraiUlFxyRLfGDiLojDPSgxpew01
3vqYI91CSFXHA+3ZcYWtExaVTS2dWyOnKgeGFbWpj/rMWnh6HF1Q8tEQD/a3PCh1DMV+uMxzzXJKgtUR
UugKIaOnQAMEYLPT3geu+bXQngHEDXhwC3Kt9zQfMOGLX2DyNHvmMrXi+hah ee614@nclcluster9
```

ee614@nclcluster1 → authorized_keys

ee614@nclcluster8 → authorized_keys

SSH authorized_key setting

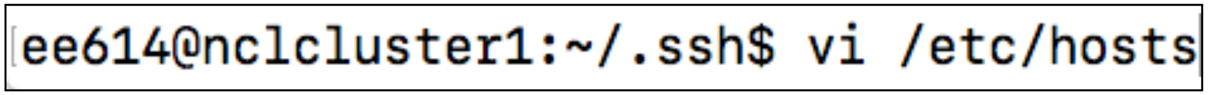
- Procedure for setting SSH key
  5) The .ssh directory is a directory of very important security information. Therefore, the following settings are recommended. Execute the following commands sequentially.



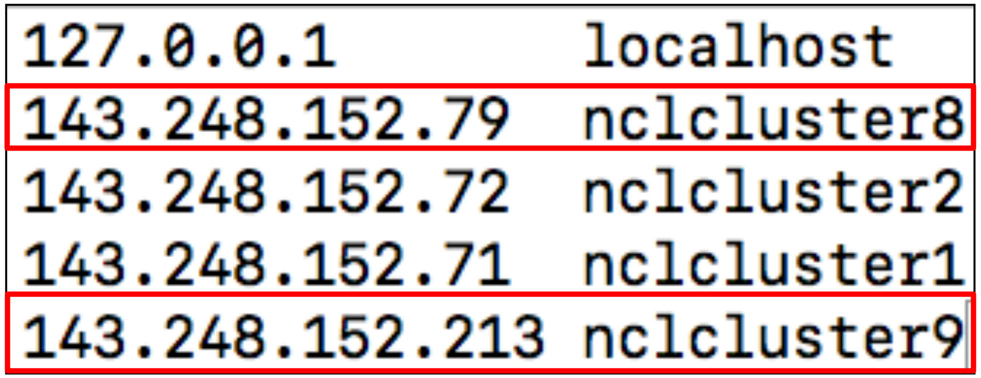
```
ee614@nclcluster9:~/.ssh$ chmod 700 ~/.ssh
ee614@nclcluster9:~/.ssh$ chmod 600 ~/.ssh/id_rsa
ee614@nclcluster9:~/.ssh$ chmod 644 ~/.ssh/id_rsa.pub
ee614@nclcluster9:~/.ssh$ chmod 644 ~/.ssh/authorized_keys
ee614@nclcluster9:~/.ssh$ chmod 644 ~/.ssh/known_hosts
```

Host name describes

- Each host describes each other's hostnames so that they can access each other's hostnames.
  - Edit "/etc/hosts" file with vi editor for describes each other's hostnames



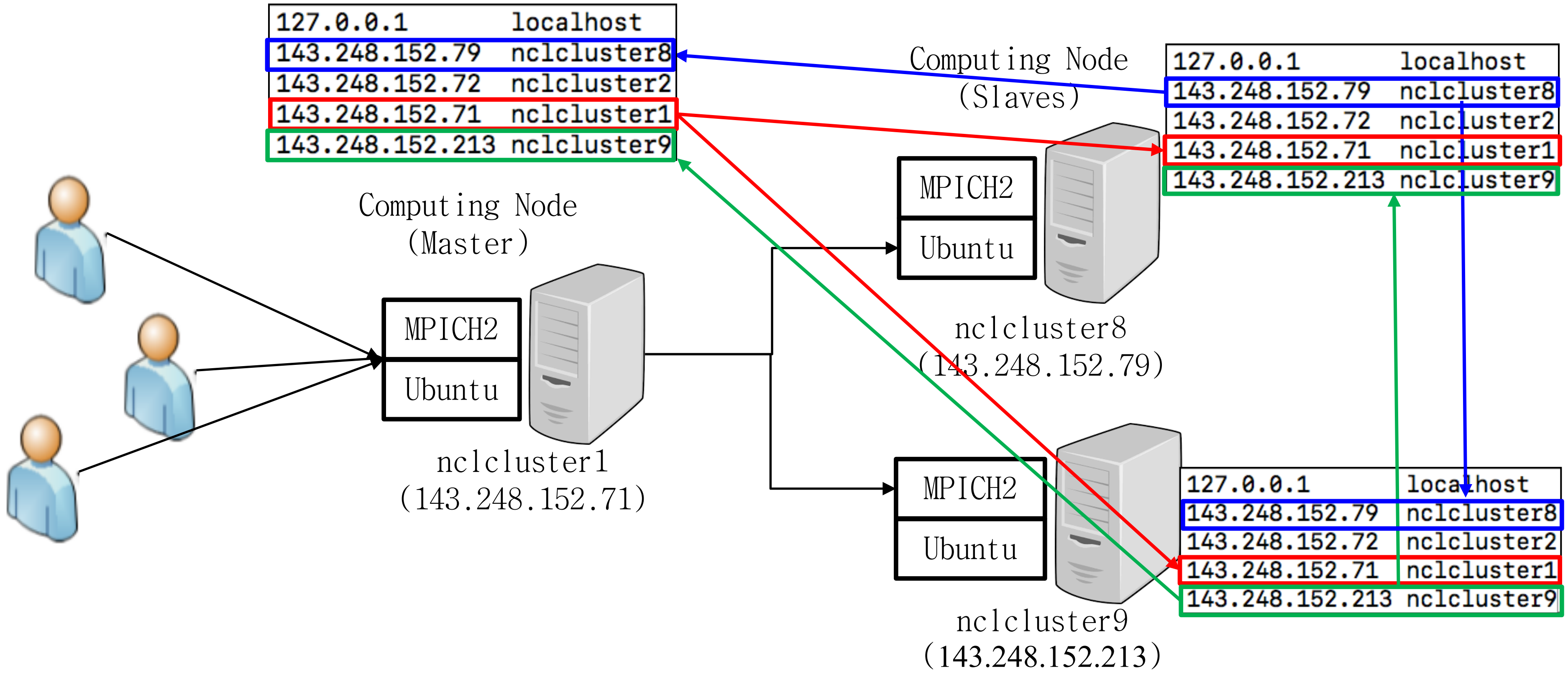
```
ee614@nclcluster1:~/.ssh$ vi /etc/hosts
```



```
127.0.0.1       localhost
143.248.152.79  nclcluster8
143.248.152.72  nclcluster2
143.248.152.71  nclcluster1
143.248.152.213 nclcluster9
```

Describes two slave node' s hostname

Host name describes



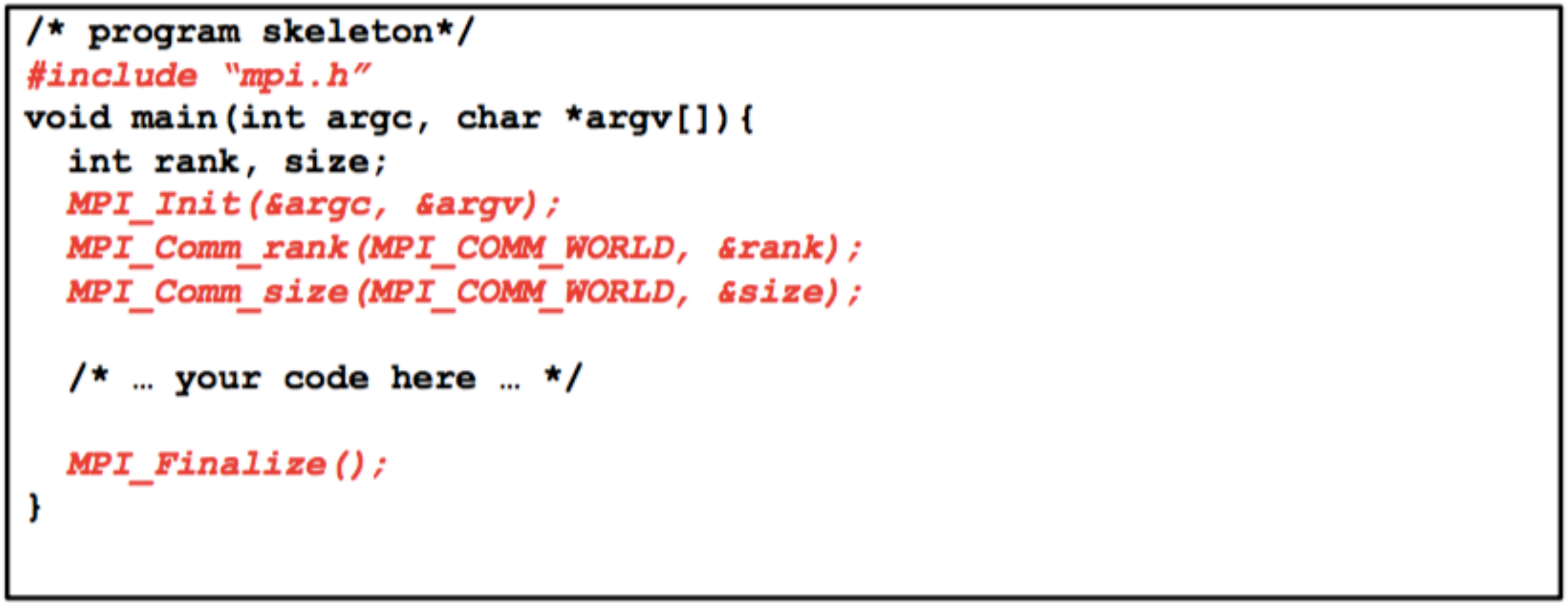
```
127.0.0.1        localhost
143.248.152.79   nclcluster8
143.248.152.72   nclcluster2
143.248.152.71   nclcluster1
143.248.152.213 nclcluster9
```

```
127.0.0.1        localhost
143.248.152.79   nclcluster8
143.248.152.72   nclcluster2
143.248.152.71   nclcluster1
143.248.152.213 nclcluster9
```

```
127.0.0.1        localhost
143.248.152.79   nclcluster8
143.248.152.72   nclcluster2
143.248.152.71   nclcluster1
143.248.152.213 nclcluster9
```

Running Program with MPICH

- Compile : mpicc –o [executable] [source]
- Execution: : mpiexec **–f [machine file]** –n [#processes] [executable]



```
/* program skeleton*/
#include "mpi.h"
void 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);

  /* … your code here … */

  MPI_Finalize();
}
```

➢ Use -f or -machinefile or -hostfile to specify cluster machines

MPD

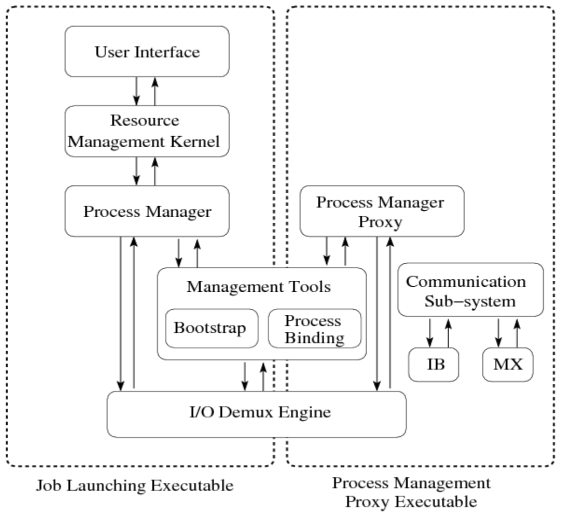
- MPICH started with an infrastructure called MPD (Multi-Purpose Daemon or something).
- It switched to the newer Hydra process manager.
- Password in .mpd.conf
  - MPI requires a file in your home directory called .mpd.conf (yes, there are two dots) which contains the line:

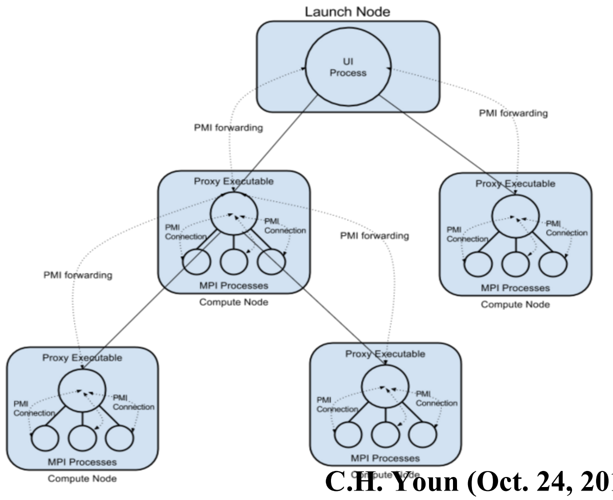
```
secretword=something_secret_password
```

Hydra Process Management Framework

- The Hydra framework has the following basic components:
  - User Interface (UI, e.g., mpiexec)
  - Resource Management Kernel (RMK)
  - Process manager
  - Bootstrap server (e.g., ssh, fork, pbs, slurm, sge)
  - Process Binding (e.g., plpa)
  - Communication Subsystem (e.g., IB, MX)
  - Process Management proxy
  - I/O demux engine

https://wiki.mpich.org/mpich/index.php/Hydra_Process_Management_Framework





Machinefile → MPD.hosts

- To be able to send the processes to other hosts on the network, create a file in your home directory called mpd.hosts which contain a list of the nodes to be used by MPI, one per line.
  - Example mpd.hosts

```
host1
host2:2
host3:4
host4:1
```

  - ’host1’, ’host2’, ’host3’ and ’host4’ are the hostnames of the machines you want to run the job on. The ’:2’, ’:4’, ’:1’ segments depict the number of processes you want to run on each node. If nothing is specified, ’:1’ is assumed.

More details on interacting with Hydra can be found at https://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager

Lab 1) Getting Started with "Hello World"

- Write a program that uses MPI and has each MPI process print
  - Hello world message from process i of n
    - The rank in MPI_COMM_WORLD for i
    - The size of MPI_COMM_WORLD for n..
    - Hostname of the machine

Lab 1) Getting Started with "Hello World"

- You may want to use these MPI routines in your solution:
  - **MPI_Init**: Initialize the MPI execution environment
  - **MPI_Comm_size**: Determines the size of the group associated with a communicator
  - **MPI_Comm_rank**: Determines the rank of the calling process in the communicator
  - **MPI_Get_processor_name**: Get hostname of the running machine
  - **MPI_Finalize**: Terminates MPI execution environment

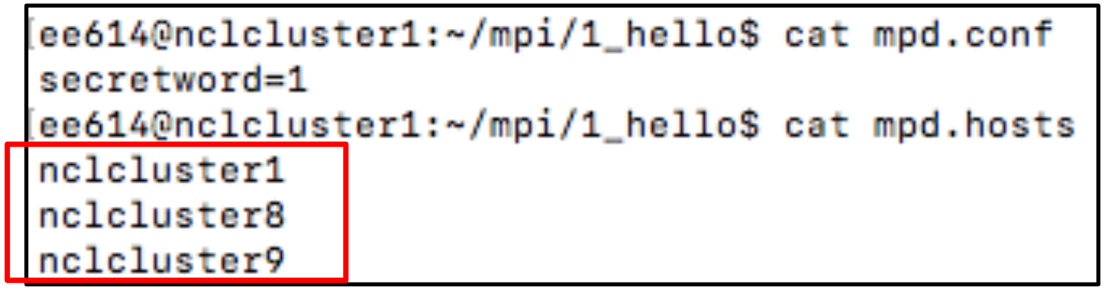
Lab 1) Getting Started with “Hello World”

Code at Lab Materials
 - Lab11/src/HelloWorld.c

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

int main(int argc, char** argv ) {
    int rank, size;
    int    namelen;
    char   hostname[50];

    MPI_Init(&argc, &argv);     //MPI Initialize
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);   //Get the rank of the
calling process in the communicator
    MPI_Comm_size(MPI_COMM_WORLD, &size);   //Get the size of the group
associated with a communicator
    MPI_Get_processor_name(hostname, &namelen);   //Get hostname of the
running machine
    printf( "Hello world from process  %d (%s) of %d\n", rank,
hostname,size);

    MPI_Finalize(); //Terminates MPI execution environment
    return 0;
}
```

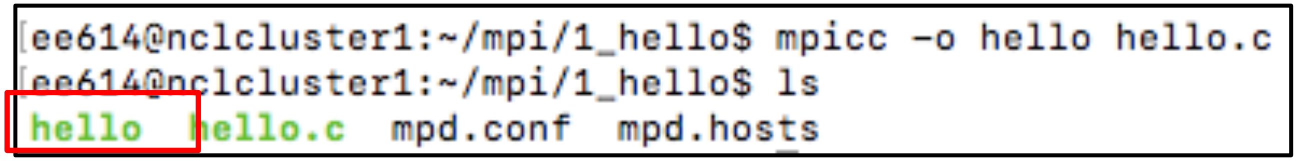
Lab 1) Getting Started with "Hello World"

1) Create mpd.conf and mpd.hosts



```
ee614@nclcluster1:~/mpi/1_hello$ cat mpd.conf
secretword=1
ee614@nclcluster1:~/mpi/1_hello$ cat mpd.hosts
nclcluster1
nclcluster8
nclcluster9
```

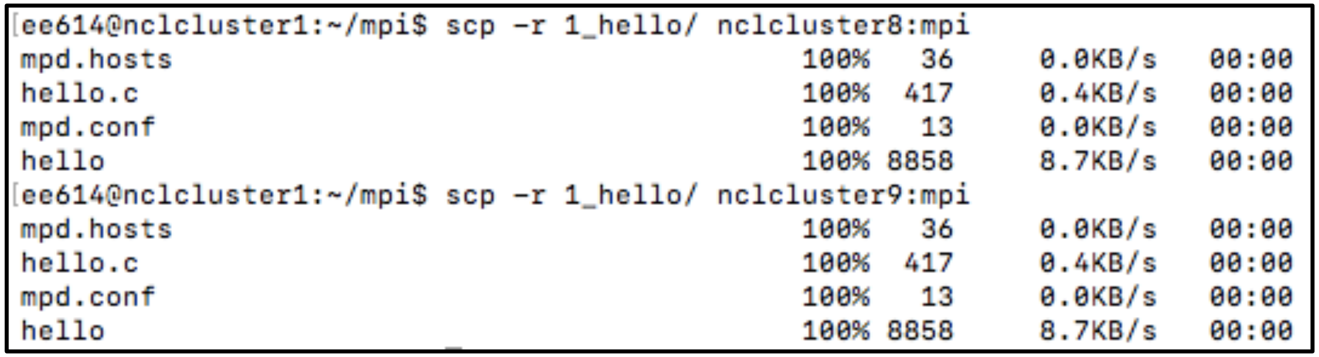
2) Compile src code with mpicc



```
ee614@nclcluster1:~/mpi/1_hello$ mpicc -o hello hello.c
ee614@nclcluster1:~/mpi/1_hello$ ls
hello  hello.c  mpd.conf  mpd.hosts
```

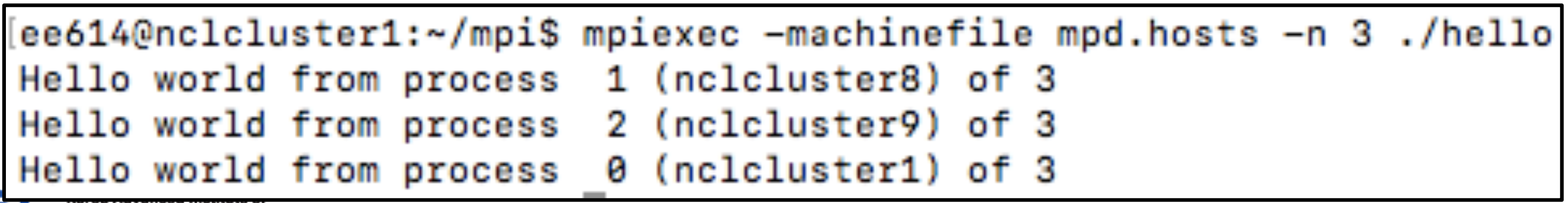
Lab 1) Getting Started with "Hello World"

3) All executables and mpd must be configured identically on each node
   - Utilize scp command



```
[ee614@nclcluster1:~/mpi$ scp -r 1_hello/ nclcluster8:mpi
 mpd.hosts                                    100%   36     0.0KB/s   00:00
 hello.c                                      100%  417     0.4KB/s   00:00
 mpd.conf                                     100%   13     0.0KB/s   00:00
 hello                                        100% 8858     8.7KB/s   00:00
[ee614@nclcluster1:~/mpi$ scp -r 1_hello/ nclcluster9:mpi
 mpd.hosts                                    100%   36     0.0KB/s   00:00
 hello.c                                      100%  417     0.4KB/s   00:00
 mpd.conf                                     100%   13     0.0KB/s   00:00
 hello                                        100% 8858     8.7KB/s   00:00
```

4) Running execuable with MPIEXEC



```
[ee614@nclcluster1:~/mpi$ mpiexec -machinefile mpd.hosts -n 3 ./hello
 Hello world from process  1 (nclcluster8) of 3
 Hello world from process  2 (nclcluster9) of 3
 Hello world from process  0 (nclcluster1) of 3
```

Lab 2) Finding PI using MPI collective operations

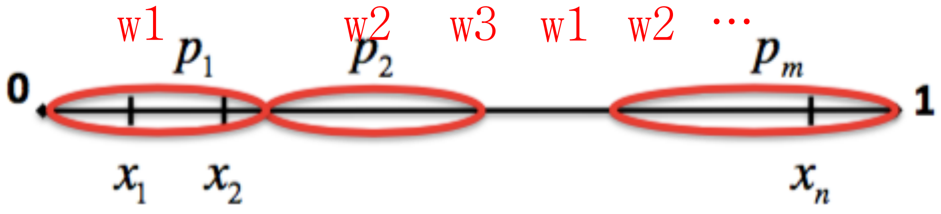
- Pi can be obtained by calculating the following value.
  - $\pi = \int_0^1 \frac{4}{1+x^2} dx$
- The integral is approximated by sum of n intervals
  - $\pi \approx \frac{1}{n}\sum_{i=0}^{n} \frac{4}{1+x_i^2}, \quad x_i = \frac{i-0.5}{n}$

https://en.wikipedia.org/wiki/Pi

Lab 2) Finding PI using MPI collective operations

- The integral is approximated by sum of n intervals
  - $\pi \approx \frac{1}{n}\sum_{i=0}^{n} \frac{4}{1+x_i^2}$, $x_i = \frac{i-0.5}{n}$
  - The summation is equally divided among the workers according to the comm_size as step size.
    - $x = \frac{rank}{n}, \frac{\text{rank+size}}{\text{n}}, \frac{\text{rank+}2\times size}{\text{n}}, \ldots$

```
for (i = myid + 1; i <= n; i += numprocs) {
        x = h * ((double)i - 0.5);
        sum += f(x);
}
```



** n is interval

Lab 2) Finding PI using MPI collective operations

- The approximation to the integral in each interval is $\frac{1}{n} \cdot \frac{4}{1+x_i^2}$
  1) The master process (rank 0) asks the user for the number of intervals
  2) The master should then broadcast this number to all of other processes
  3) Each process then adds up every n'th interval $(x = \frac{rank}{n}, \frac{\mathrm{rank+size}}{\mathrm{n}}, \frac{\mathrm{rank}+2\times size}{\mathrm{n}}, \ldots)$
  4) Finally, the sums computed by each process are added together using a reduction

Lab 2) Finding PI using MPI collective operations

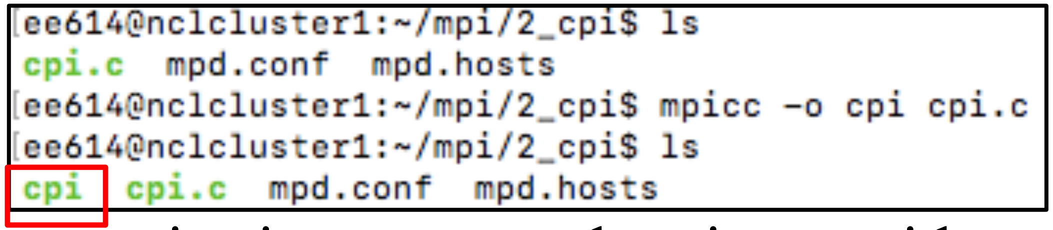
Code at Lab Materials
- Lab11/src/cpi.c

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

double f(double a) {
    return (4.0 / (1.0 + a*a));
}

int main(int argc,char *argv[]) {
    int    n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int    namelen;
    char   processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);

    fprintf(stdout,"Process %d of %d is on %s\n", myid, numprocs,
processor_name);
    fflush(stdout);

    n = 10000;                     /* default # of rectangles */
    if (myid == 0) {
        startwtime = MPI_Wtime();
        printf("Enter the number of intervals: (0 quits) ");
        fflush(stdout);
        scanf("%d",&n);
    }

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

```c
    h   = 1.0 / (double) n;
    sum = 0.0;
    for (i = myid + 1; i <= n; i += numprocs) {
        x = h * ((double)i - 0.5);
        sum += f(x);
    }
    mypi = h * sum;

    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    if (myid == 0) {
        endwtime = MPI_Wtime();
        printf("pi is approximately %.16f, Error is %.16f\n", pi,
fabs(pi - PI25DT));
        printf("wall clock time = %f\n", endwtime-startwtime);
        fflush(stdout);
    }

    MPI_Finalize();
    return 0;
}
```

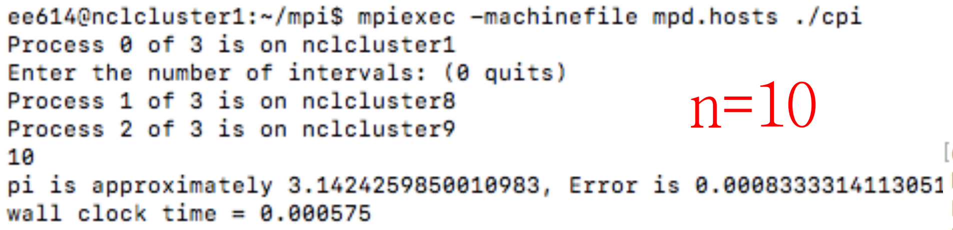
Lab 2) Finding PI using MPI collective operations

- Prepare executable and MPD files for all machines



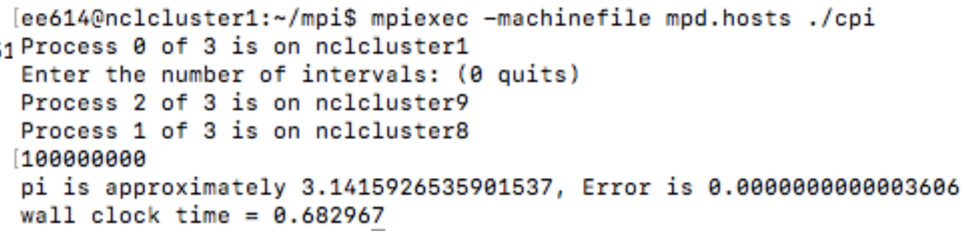
```
[ee614@nclcluster1:~/mpi/2_cpi$ ls
cpi.c  mpd.conf  mpd.hosts
[ee614@nclcluster1:~/mpi/2_cpi$ mpicc -o cpi cpi.c
[ee614@nclcluster1:~/mpi/2_cpi$ ls
cpi  cpi.c  mpd.conf  mpd.hosts
```

- Execute cpi using command mpiexec with various interval value



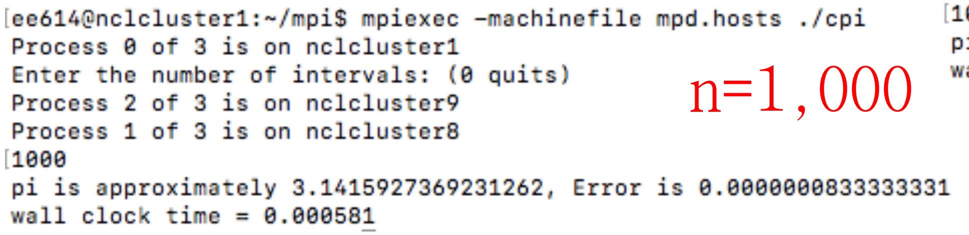
n=10

```
ee614@nclcluster1:~/mpi$ mpiexec -machinefile mpd.hosts ./cpi
Process 0 of 3 is on nclcluster1
Enter the number of intervals: (0 quits)
Process 1 of 3 is on nclcluster8
Process 2 of 3 is on nclcluster9
10
pi is approximately 3.1424259850010983, Error is 0.0008333314113051
wall clock time = 0.000575
```



n=100,000,000

```
[ee614@nclcluster1:~/mpi$ mpiexec -machinefile mpd.hosts ./cpi
Process 0 of 3 is on nclcluster1
Enter the number of intervals: (0 quits)
Process 2 of 3 is on nclcluster9
Process 1 of 3 is on nclcluster8
[100000000
pi is approximately 3.1415926535901537, Error is 0.0000000000003606
wall clock time = 0.682967
```

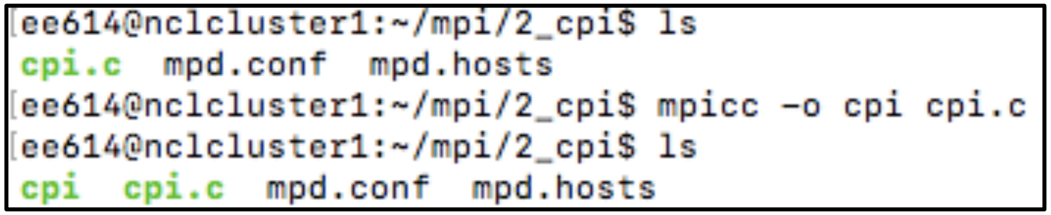


n=1,000

```
[ee614@nclcluster1:~/mpi$ mpiexec -machinefile mpd.hosts ./cpi
Process 0 of 3 is on nclcluster1
Enter the number of intervals: (0 quits)
Process 2 of 3 is on nclcluster9
Process 1 of 3 is on nclcluster8
[1000
pi is approximately 3.1415927369231262, Error is 0.0000000833333331
wall clock time = 0.000581
```

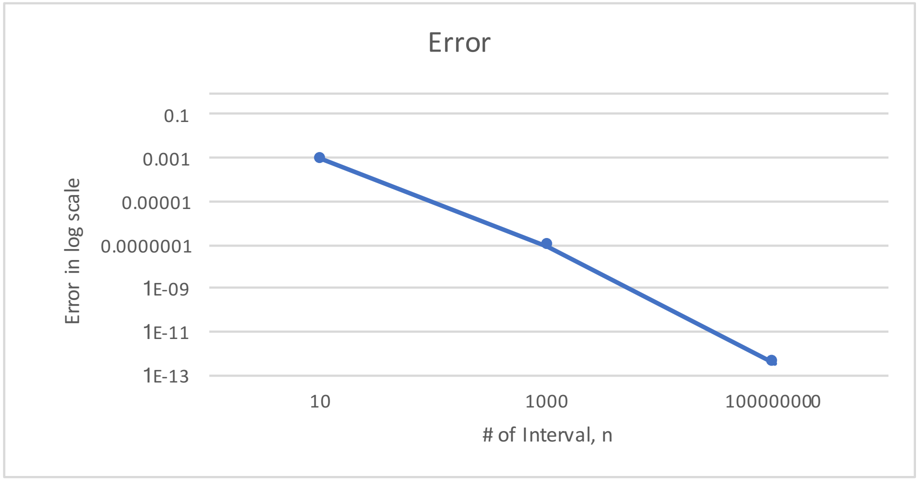
Lab 2) Finding PI using MPI collective operations

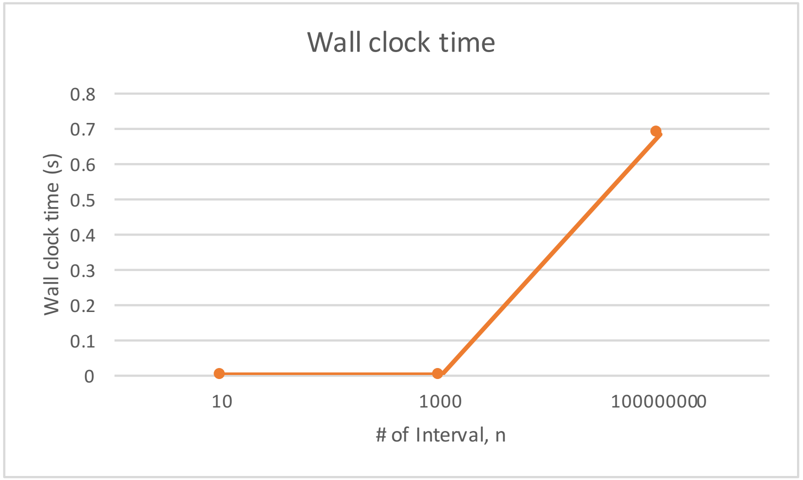
- Prepare executable and MPD files for all machines



```
ee614@nclcluster1:~/mpi/2_cpi$ ls
cpi.c   mpd.conf   mpd.hosts
ee614@nclcluster1:~/mpi/2_cpi$ mpicc -o cpi cpi.c
ee614@nclcluster1:~/mpi/2_cpi$ ls
cpi   cpi.c   mpd.conf   mpd.hosts
```

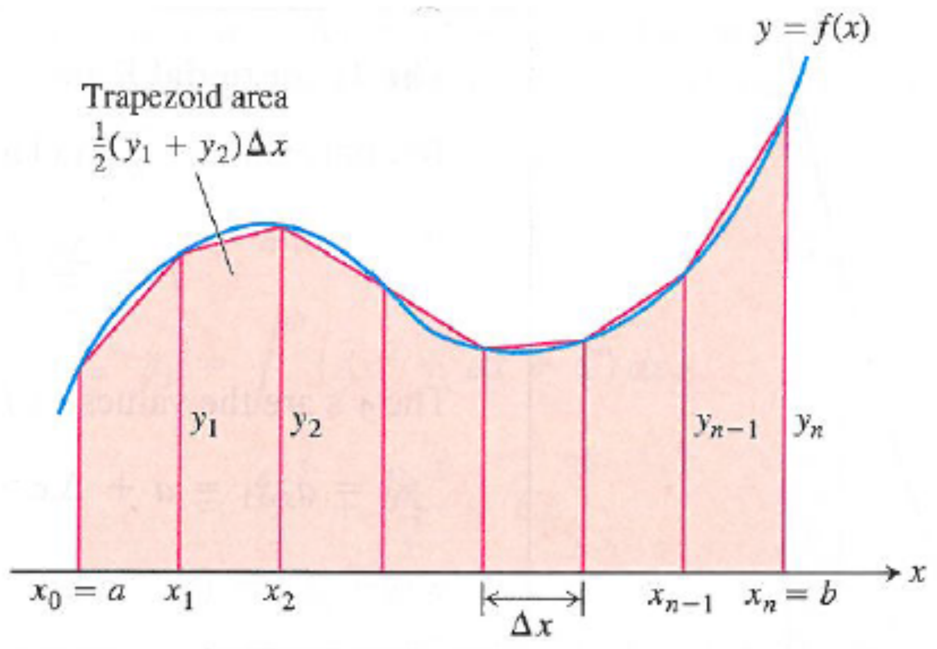
- Compare results in accuracy / wall clock time





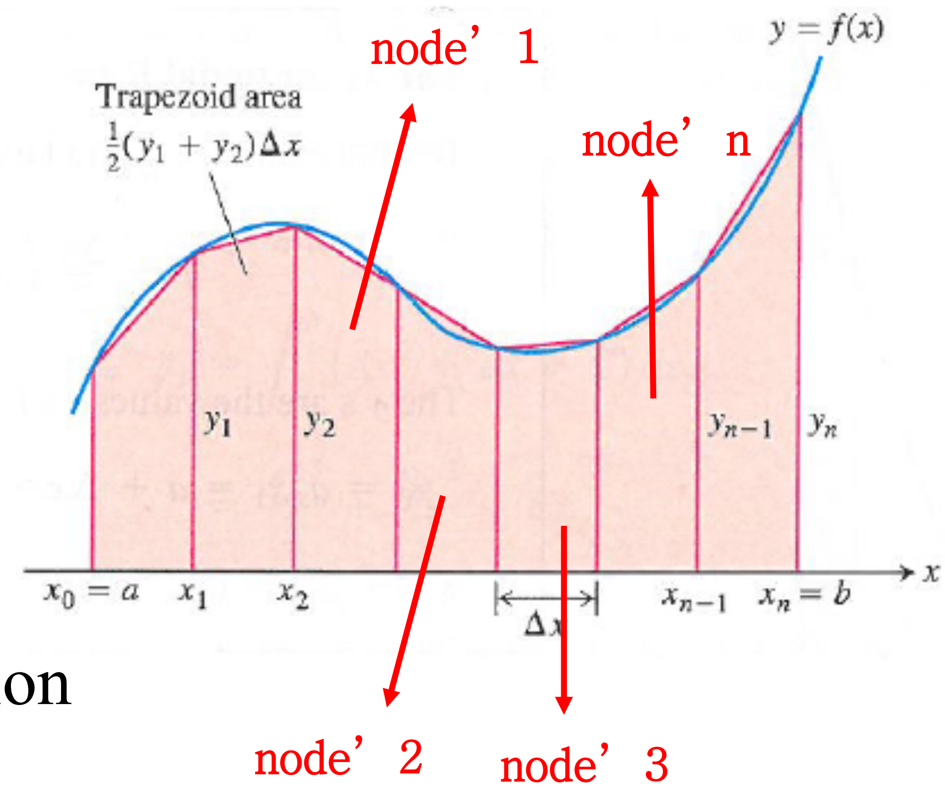
Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

- The trapezoidal rule is a method for finding an approximate value for a definite integral.
- Cut the section and draw a line connecting the value in the section and the value in the next section, and then taking the width of the range and adding it.



Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

- To compute the area of y using the trapezoidal rule, cutting more sections result the high accuracy.
- However, the calculation tim increases accordingly.
- Therefore, the area of each section is obtained through distributed processing to eacl node using MPI, the calculation time will be reduced.



Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

- Functions to obtain a local section's area.

```
double Trap(
    double  local_a   /* in */,
    double  local_b   /* in */,
    int     local_n   /* in */,
    double  h         /* in */) {
    double integral;   /* Store result in integral  */
    double x;
    int i;
    double f(double x); /* function we're integrating */

    integral = (f(local_a) + f(local_b))/2.0;
    x = local_a;
    for (i = 1; i <= local_n-1; i++) {
        x = x + h;
        integral = integral + f(x);
    }
    integral = integral*h;
    return integral;
}
```

$$\int_a^b f(x)dx$$

$$h = \frac{b-a}{n}$$

```
double f(double x) {
    double return_val;
    int i;

 /* Calculate f(x). */
 /* Store calculation in return_val. */
    return_val = 5*x*x*x*x;
        for (i=1;i<100000;i++);
    return return_val;
}
```

Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

- To distributed processing to each node using MPI, we need MPI_Send and MPI_Recv function.
  - MPI_Send(Blocking send)
    - int MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)
    - This routine may block until the message is received by the destination process.
  - MPI_Recv(Blocking receive for a message)
    - int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)
    - The count argument indicates the maximum length of a message; the actual length of the message can be determined with MPI_Get_count.

Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

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

main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    start = MPI_Wtime();
    h = (b-a)/n;    /* h is the same for all processes */
    local_n = n/p;  /* So is the number of trapezoids */

    local_a = a + my_rank*local_n*h;
    local_b = local_a + local_n*h;
    integral = Trap(local_a, local_b, local_n, h);

        /** (1) INSERT RECV & SEND OPERATIONS HERE **/
        if(my_rank == 0){
                for(source = 1; source < p; ++source){
                        MPI_Recv(&integral, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);
                        total += integral;
                }
        }
        else{
                MPI_Send(&integral, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
        }
    finish = MPI_Wtime();
    /* Print the result */
    MPI_Finalize();
}
```

MPI Initialize

Get the rank of the process in the communicator

Get the size of the group with a communicator

Code at Lab Materials
- Lab11/src/trap.c

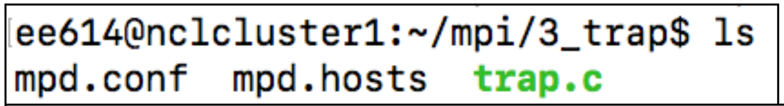
MPI_Recv (receive area of section)

MPI_Send (send area of section)

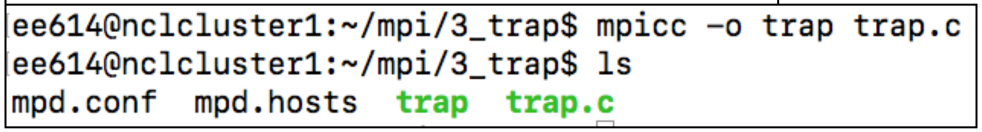
Terminates MPI execution environment

Lab 3) Solving trapezoidal rule using MPI_Send / MPI_Recv

- Prepare executable and MPD files for all machines



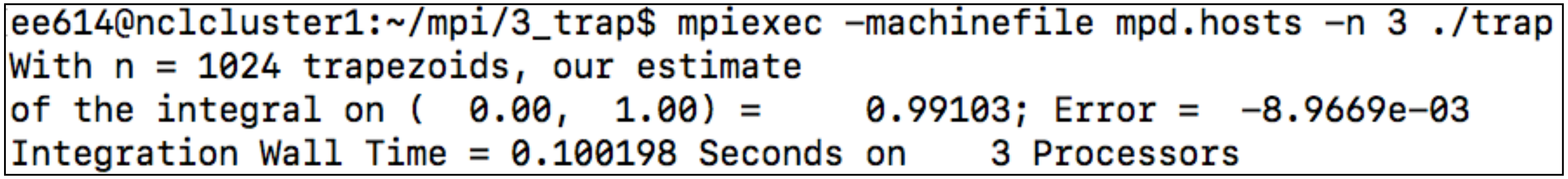
```
ee614@nclcluster1:~/mpi/3_trap$ ls
mpd.conf   mpd.hosts   trap.c
```



```
ee614@nclcluster1:~/mpi/3_trap$ mpicc -o trap trap.c
ee614@nclcluster1:~/mpi/3_trap$ ls
mpd.conf   mpd.hosts   trap   trap.c
```

- Execute cpi using command mpiexec with various interval value

n = 1024 (default)

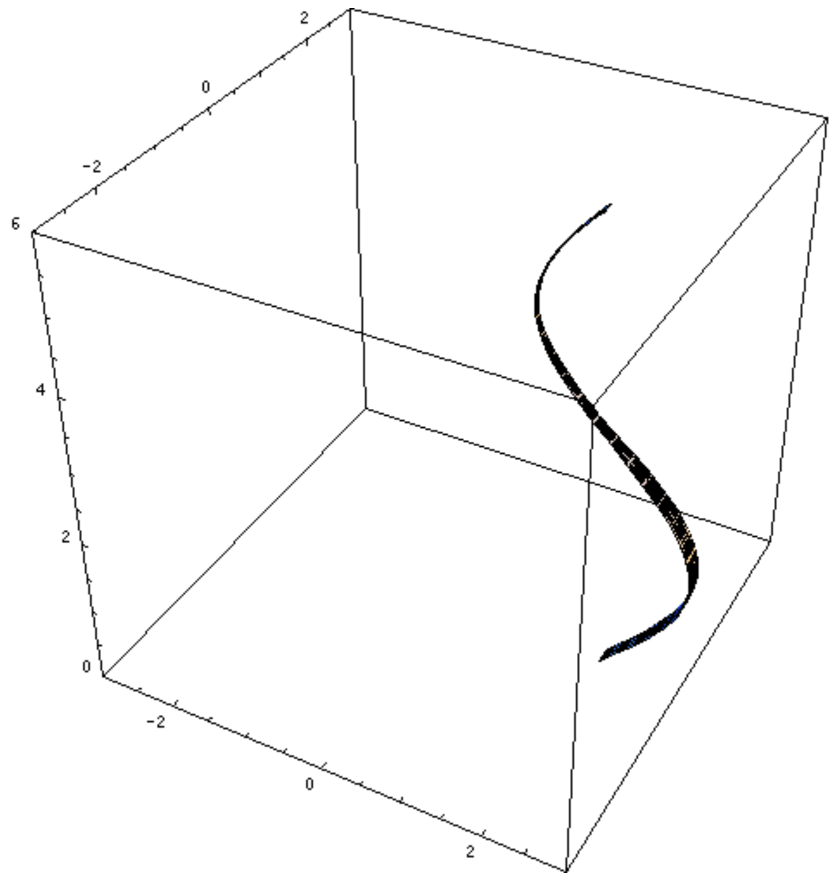


```
ee614@nclcluster1:~/mpi/3_trap$ mpiexec -machinefile mpd.hosts -n 3 ./trap
With n = 1024 trapezoids, our estimate
of the integral on (  0.00,   1.00) =      0.99103; Error =  -8.9669e-03
Integration Wall Time = 0.100198 Seconds on     3 Processors
```

- (Lab) Use different parameters to analyze execution time and accuracy.

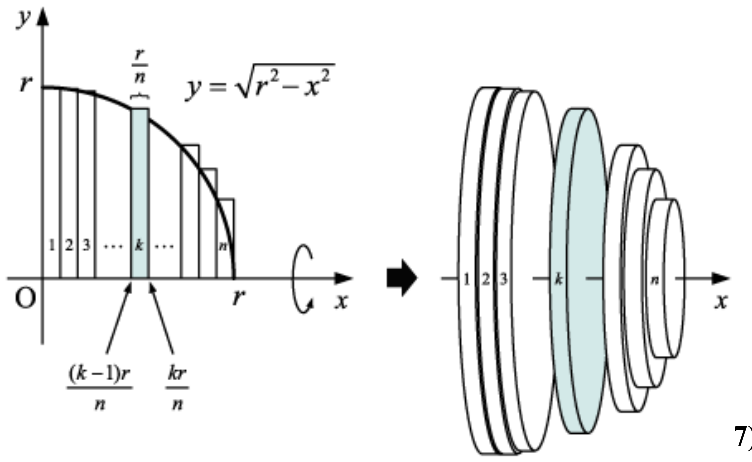
Lab 4) Volume of the Solid of revolution

- In mathematics, a solid of revolution is a solid figure obtained by rotating a plane curve around some straight line that lies on the same plane.



Lab 4) Volume of the Solid of revolution

- **Disk method**
  - The disk method is used when the slice that was drawn is perpendicular to the axis of revolution The volume of the solid formed by rotating the area between the curves of f(x) and g(x) and the lines x = a and x = b about the x-axis is given by
  - $V = \pi \int_a^b |f(x)^2 - g(x)^2| dx$
  - If g(x) = 0 (e.g. revolving an area between the curve and the x-axis), this reduces to:
  - $V = \pi \int_a^b f(x)^2 dx$

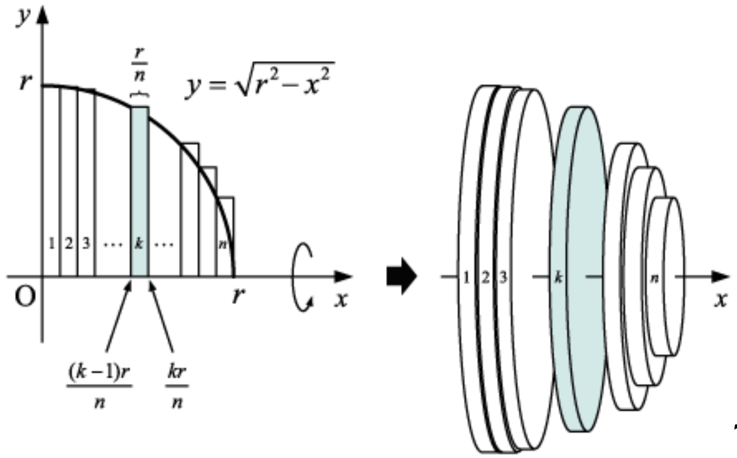


7)

KAIST Korea Advanced Institute of Science and Technology

Lab 4) Volume of the Solid of revolution

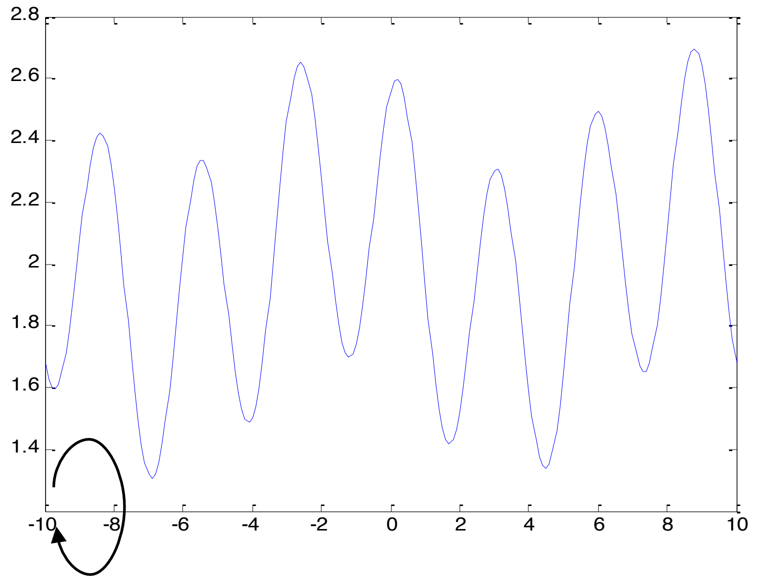
- **Disk method (Mensuration by parts)**
  - For a complex type of solid of revolution volume, it can be approximated by summing up the volume of each cylinder.
  - $V = \pi \sum f(x + n \cdot \Delta x)^2 \times (n \cdot \Delta x)$
  - The input is the starting and ending points of the integral, and the precision scale dx. The volume is integrated in dx units to obtain the volume.



7)

Lab 4) Volume of the Solid of revolution

- **Complex solid of revolution (example)**
  - $f(x) = 2 + 0.2 \cdot sin(0.2 \cdot \pi \cdot x + 2.5) + 0.5 \cdot cos(0.7 \cdot \pi \cdot x - 0.5)$
  - How can we get the volume?
    - Disk method (mensuration by parts)



Lab 4) Volume of the Solid of revolution

Code at Lab Materials
- Lab11/src/volume.c

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

#define PI 3.14159265

inline double f(double x);

int main(int argc, char* argv[]) {
        int size, rank;
        double from, to, dx, result;
        double start_time, end_time;

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

        printf("rank %d size %d\n", rank, size);
        fflush(stdout);
        MPI_Barrier(MPI_COMM_WORLD);
        if(rank == 0) {
                start_time = MPI_Wtime();
                printf("from: ");
                fflush(stdout);
                scanf("%lf", &from);
                printf("to: ");
                fflush(stdout);
                scanf("%lf", &to);
                printf("dx: ");
                fflush(stdout);
                scanf("%lf", &dx);
                printf("%lf %lf %lf\n", from, to, dx);
                fflush(stdout);
        }
        MPI_Bcast(&from, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
        MPI_Bcast(&to, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
        MPI_Bcast(&dx, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Broadcast parameter

Mensuration by parts

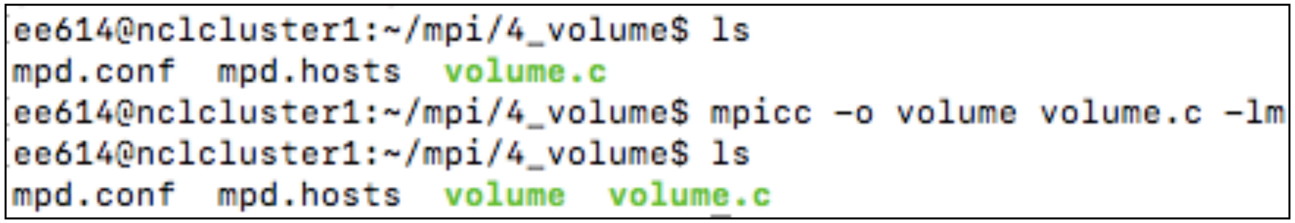
```c
        double portion = (to-from)/size;
        double start = from+portion*rank;
        double end = from+portion*(rank+1);
        double localResult = 0;

        double x;
        for(x = start; x < end; x += dx)
                localResult += ((f(x)>0)?f(x)*f(x):0);
        localResult *= PI*dx;

        MPI_Reduce(&localResult, &result, 1, MPI_DOUBLE, MPI_SUM,  0,
MPI_COMM_WORLD);
```

Reduce to summing up results

```c
        if(rank == 0) {
                end_time = MPI_Wtime();
                printf("from %lf to %lf dx %lf volume %lf\n", from, to,
dx, result);
                fflush(stdout);
                printf("process time %lf\n", end_time-start_time);
                fflush(stdout);
        }

        MPI_Finalize();
        return 0;
}

inline double f(double x) {
        return 2+0.2*sin(0.2*PI*x+2.5)+0.5*cos(0.7*PI*x-0.5);
}
```

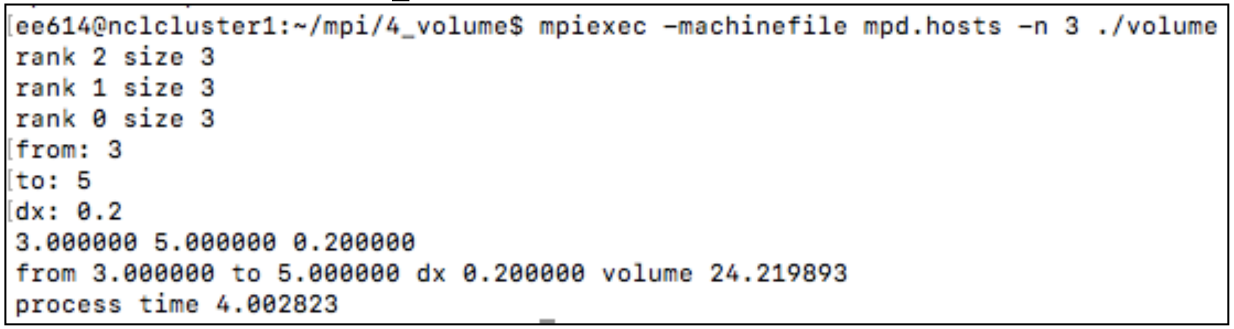
Lab 4) Volume of the Solid of revolution

- Prepare executable and MPD files for all machines



```
ee614@nclcluster1:~/mpi/4_volume$ ls
mpd.conf  mpd.hosts  volume.c
ee614@nclcluster1:~/mpi/4_volume$ mpicc -o volume volume.c -lm
ee614@nclcluster1:~/mpi/4_volume$ ls
mpd.conf  mpd.hosts  volume  volume.c
```

- Execute cpi using command mpiexec with various from, to and dx parameter



```
ee614@nclcluster1:~/mpi/4_volume$ mpiexec -machinefile mpd.hosts -n 3 ./volume
rank 2 size 3
rank 1 size 3
rank 0 size 3
from: 3
to: 5
dx: 0.2
3.000000 5.000000 0.200000
from 3.000000 to 5.000000 dx 0.200000 volume 24.219893
process time 4.002823
```

- (Lab) Use different parameters and function shape to analyze execution time and accuracy.

Lab 11. MPI Programming on Linux

- Read text materials and test practices
- Use your notebook PC to examine context and produce the source code when you finish the successful practice.
- Report installation procedure on your PC environment
- Objectives
  - Understand execution process of MPICH and machine settings
  - Understand MPI's basic API and how to run it through example exercises.
  - Compare the error and execution time while changing the precision.

Appendix. DFT

Code at Lab Materials
 - Lab11/src/dft.c

- Discrete Fourier Transform(DFT)
- DFT는 Discrete한 특성을 가지는 시간 축 신호를 Discrete한 주파수 대역의 신호로 변환해주는 기능을 한다. 이를 통해 얻은 주파수 대역의 신호는 시간 대역 신호보다 분석하기에 용이하고 Sparse한 특성을 가지고 있기에 양자화 및 소스코딩을 할 시 거의 전처리로 수행되는 기법이다.

Appendix. DFT

- Fourier Transform의 기본 원리는 Orthogonal에 있다. 1, sin, cos 그리고 이를 확장시킨 e와 같은 경우에 모두가 서로 직교성을 가지고 있고 주파수가 다를 경우에도 서로 직교성을 유지하고 있다. 이 경우에 각 함수는 좌표축을 이루게 되고 이를 베이스라고 한다. 따라서 시간 대역의 신호는 좌표공간에서 각 좌표성분의 선형 조합으로 이루어져 있는 것이고 따라서 이를 다시 좌표성분으로 추출할 수 있다. 이는 각각 inverse fourier transform, fourier transform이라고 부른다. fourier transform은 신호의 형태에 따라 여러가지 형태를 가진다. 물론 위의 기본적인 원리를 기반으로 하고 있다. Continuous영역에서 사용되는 일반적인 fourier transform의 경우에는 다음과 같은 모습을 가진다.
- $X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi tf}dx$
- $x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi tf}df$
-

Appendix. DFT

- 하지만 컴퓨터 영역에서 사용하는 경우에 아날로그 신호는 존재하지 못하므로 디지털 신호로 모두 변환된다. 이 과정이 샘플링과 양자화이다. 이 과정을 거치면 신호는 Discrete한 특징을 가지므로 조작이 쉬워진다. 이번 HW에서 구현한 fourier transform은 Discrete Fourier Transform으로 이산 푸리에 변환이라고 부른다. 우선 식은 다음과 같다.
- $\mathrm{X_k} = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}$
- $\mathrm{x_n} = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn}$
- 자세히 보면 이 식은 구현할 수 없다는 것을 알 수 있다. 바로 e의 지수부분에 복소수가 있어서 문제가 되는 것이다. 이를 해결하기 위해 $\mathrm{e^{ix}} = \cos(x) + i\sin(x)$ 수식을 이용하여 전개한다. 추가적인 고려사항까지 생각하여 최종적으로 만들어 낸 수식은 다음과 같다.
- $\mathrm{X[k]} = \frac{1}{\sqrt{n}}\left(\sum_{i=0}^{n-1} x[i]\cos(2\pi ik/n) - j\sum_{i=0}^{n-1} x[i]\sin(2\pi ik/n)\right)$

Appendix. DFT

- 위와 같은 형식에서도 문제점이 발생한다. sin함수쪽의 허수 형태인데 이는 실수부와 허수부를 따로 저장하고 처리하는 방식으로 구현한다. 일반적으로 DFT를 빠르게 사용하기 위해 FFT를 사용한다. 하지만 여기서는 계산물리학 등 많은 연산량에 강한 분산컴퓨팅을 테스트해보는 기회이기 때문에 위의 DFT 식을 이용한다.
- 실제 구현은 다음과 같다. 시간대역 입력 신호로는 $x[i] = 0.5e^{j4\pi i/n} + 1.3\sin(6\pi i/n)$로 임의의 신호를 입력하였다. 여기에 Fourier Matrix상의 n개의 coefficient들이 곱해져서 더해진다. 이는 X_k로 되어 연산이 완료된다. 연산의 분배는 열(row)를 기준으로 자른다. 모든노드들은 자신에게 할당 된 행렬의 곱만 진행하여 결과값을 마스터 노드에 리턴한다. 마스터 모드는 이 정보들을 모아 완성된 정보를 가지게 된다.