OpenMPI Cluster

Sameer Patil

August 2024

1 Introduction

In this document the procedure to create a openmpi cluster in ubuntu platform is explained. The cluster setup needs minimum two computers, you can implement this using virtual machines also.

1.1 What is cluster in distriubuted computing/cloud computing?

The group of computers communicating with each other using network to execute given task/program/job/operation as if they are single computer.

1.2 What is OpenMPI?

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.

1.2.1 Message Passing Interface

The MPI standard includes point-to-point message-passing, collective communications, group and communicator concepts, process topologies, environmental management, process creation and management, one-sided communications, extended collective operations, external interfaces, I/O, some miscellaneous topics, and multiple tool interfaces. Language bindings for C and Fortran are defined.

Historically, the evolution of the standard is:

- MPI-1.0 (May 5, 1994): Initial release.
- MPI-1.1 (June 12, 1995): Minor updates and bug fixes.

- MPI-1.2 (July 18, 1997): Several clarifications and additions.
- MPI-2.0 (July 18, 1997): New functionality and all the clarifications and additions from MPI-1.2.
- MPI-1.3 (May 30, 2008): For historical reasons, combining the MPI-1.1, MPI-1.2, and several errata documents into one combined document.
- MPI-2.1 (June 23, 2008): Combining the previous documents.
- MPI-2.2 (September 4, 2009): Additional clarifications and seven new routines.
- MPI-3.0 (September 21, 2012): Extension of MPI-2.2.
- MPI-3.1 (June 4, 2015): Clarifications and minor extensions to MPI-3.0.
- MPI-4.0 (June 9, 2021): Significant new features beyond MPI-3.1.
- MPI-4.1 (November 2, 2023): Clarifications and minor extensions to MPI-4.0.

1.2.2 Goals of MPI

MPI (Message-Passing Interface) is a message-passing library interface specification. All parts of this definition are significant. MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process. Extensions to the "classical" message-passing model are provided in collective operations, remote-memory access operations, dynamic process creation, and parallel I/O. MPI is a specification, not an implementation; there are multiple implementations of MPI. This specification is for a library interface; MPI is not a language, and all MPI operations are expressed as functions, subroutines, or methods, according to the appropriate language bindings that, for C and Fortran, are part of the MPI standard. The standard has been defined through an open process by a community of parallel computing vendors, computer scientists, and application developers. The next few sections provide an overview of the history of MPI's development. The main advantages of establishing a message passing standard are portability and ease of use. In a distributed memory communication environment in which the higher level routines and/or abstractions are built upon lower level message passing routines, the benefits of standardization are particularly apparent. Furthermore, the definition of a message passing standard, such as that proposed here, provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases for which they can provide hardware support, thereby enhancing scalability. The goal of the Message-Passing Interface, simply stated, is to develop a widely used standard for writing message-passing programs. As such the interface should establish a practical, portable, efficient, and flexible standard for message passing. A complete list of goals follows.

- Design an application programming interface (not necessarily for compilers or a system implementation library).
- Allow efficient communication: Avoid memory-to-memory copying, allow overlap of computation and communication, and offload to communication co-processors, where available.
- Allow for implementations that can be used in a heterogeneous environment.
- Allow convenient C and Fortran bindings for the interface.
- Assume a reliable communication interface: the user need not cope with communication failures. Such failures are dealt with by the underlying communication subsystem
- Define an interface that can be implemented on many vendor's platforms, with no significant changes in the underlying communication and system software.
- Semantics of the interface should be language independent.
- The interface should be designed to allow for thread safety.

2 Cluster Setup

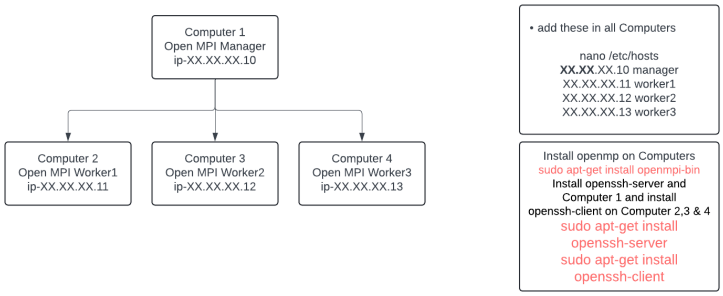

Figure 1: Cluster SetUp

The above diagram show the setup requirement for cluster. The diagram is showing the cluster of 4 computers. The computer 1 in the digram will work as a manager of the cluster and computers 2,3 and 4 will work as the workers in the cluster. You can add more number of workers by repeating the steps for adding worker. The minimum number of computers you shall use is two - one for manager and one for worker.
Open /etc/hosts file in all computers and add these lines at the end of the file

XX.XX.XX.10 manager # here xx represent the values of IP address
XX.XX.XX.11 worker1 # replace xx with appropriate values
XX.XX.XX.12 worker2
XX.XX.XX.13 worker3
example file
192.168.1.10 manager
192.168.1.11 worker1 and so on

2.1 Installing common packages for manager and worker

The setup is for Ubuntu platform - Ubuntu 18 or above. It is tested till Ubuntu 2024 - LTS.

Make sure you have updated repository information in the system. To confirm this use command.

sudo apt-get update

If the command get the error while executing, check your internet connection. If you are using proxy for internet connection. Then Setup /etc/apt/apt.conf file in ubuntu. Sample contents of apt.conf file are as below

$Acquire :: http\_proxy = http : //172.16.100.252 : 3128$
$Acquire :: https\_proxy = http : //172.16.100.252 : 3128$
$Acquire :: ftp\_proxy = http : //172.16.100.252 : 3128$

Once the update is successfully, You are ready to install the packages on your computers.
The next command will install openmpi packages on the computer.

sudo apt-get install openmpi-bin

Lets install openssh package on client and server. We have to perform ssh handshake between manager and workers. It is required to give the manager passwordless access to worker computers. This feature enables manager to access resource on the worker computer by login in to the worker computer without being prompted for password.

sudo apt-get install openssh-server # for manger - Computer 1 in Figure 1

sudo apt-get install openssh-client # for worker - Computer 2,3,4 in Figure 1

Make sure that you have same username and hostname in manager and worker computers. In this example the username and hostname both are set to ubuntu.

2.2 ssh key exchange

On the computer 1 (manager) :- In the home directory of user, find ".ssh" folder. Its a hidden folder. You can list hidden files in the folder by executing ls -a command. Once you locate the .ssh folder in the home directory, navigate inside the directory, and execute these commands.
To go inside .ssh directory command is:- (your current location is your home directory
cd /.ssh
Generate ssh keypair :-
ssh-keygen -t rsa
copy the key in authrized_key file :-
cat id_rsa.pub ¿¿ authorized_keys
copy the ssh-id on worker computers :-
ssh-copy-id worker1 or ssh-copy-id XX.XX.XX.11
ssh-copy-id worker2 or ssh-copy-id xx.xx.xx.12
ssh-copy-id worker3 or ssh-copy-id xx.xx.xx.13
These commands will ask for the password of worker computers. Once this process is complete try to login in to each worker from master computer's terminal.
ssh worker1
It should not ask for password. Same for worker2 and worker3. From worker computers, copy the id to manager computer.

2.3 NFS Server on manager and client on worker

The nfs server and client setup is required to send and receive the instructions and data between manager and worker. To achieve this install nfs server on manager system.
sudo apt-get install nfs-kernel-server
We need to create a folder that we will share across the network. In our case, we used "cloud". To export the cloud directory, we need to create an entry in /etc/exports
sudo nano /etc/exports
add
/home/ubuntu/cloud *(rw,sync,no_root_squash,no_subtree_check)
here * can be replaced with the IP address of the worker computers.
After an entry is made, run the following.
exportfs -a