Participants: Hanku Lee, Joshua DeNio, Heecheon Park

Date: Thu Jun 20 21:45:09 CDT 2019

**Master Cluster Hardware Information:**

Architecture:    x86\_64

CPU op-mode(s):  32-bit, 64-bit

Byte Order:      Little Endian

CPU(s):          4

On-line CPU(s) list: 0-3

Thread(s) per core:  2

Core(s) per socket:  2

Socket(s):       1

NUMA node(s):    1

Vendor ID:       GenuineIntel

CPU family:      6

Model:           58

Model name:      Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz

Stepping:        9

CPU MHz:         1230.960

CPU max MHz:     3200.0000

CPU min MHz:     1200.0000

BogoMIPS:        5187.98

Virtualization:  VT-x

L1d cache:       32K

L1i cache:       32K

L2 cache:        256K

L3 cache:        3072K

NUMA node0 CPU(s):   0-3

Flags:           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant\_tsc arch\_perfmon pebs bts rep\_good nopl xtopology nonstop\_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds\_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4\_1 sse4\_2 x2apic popcnt tsc\_deadline\_timer xsave avx f16c rdrand lahf\_lm cpuid\_fault epb pti ssbd ibrs ibpb stibp tpr\_shadow vnmi flexpriority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts md\_clear flush\_l1d

**Basic Setup on Master cluster:**

1. Install mpich via “sudo apt-get install mpich”
2. Configure /etc/hosts file. (Master node must have all the list of ip addresses and identity of slave nodes and master node itself) // mpirun or mpiexec can load this file to execute a program in parallel.
3. Add a user “mpiuser” (All nodes having mpiuser is preferred to reduce possible confusion) // sudo adduser mpiuser
4. Make mpiuser a sudoer “sudo usermod -a -G sudo mpiuser”
5. Setup an ssh server so that the master and slave clusters can interchange messages. “sudo apt-get install openssh-server”
6. In order for mpirun / mpiexec to execute a program in parallel smoothly, a passwordless ssh environment must be set up. Setup steps are as follows:
   1. su - mpiuser
   2. ssh-keygen -t rsa \*  or -t dsa
   3. ssh-copy-id client \*\* ip-address may also be used.
   4. eval `ssh-agent`
   5. ssh-add ~/.ssh/id\_rsa
   6. If done correctly with all the steps above, the master should be able to ssh to the slave nodes without a password.
   7. *If it is not working, rm ~/.ssh/id\* and redo the steps above*

**step g might fix lots of potential problems.**

\*  you might have to use -t dsa if your system is outdated. Like linux before 2000.

\*\* client is defined in machine file or /etc/hosts

i.e. cat /etc/hosts

      192.168.x.x client

      127.0.0.1 localhost

1. Once a passwordless ssh channel is created, a shared directory (nfs in this case), where a message can be interchanged, needs to be created. Passwordless ssh must be available to proceed to this step. Steps are as follows:
   1. sudo apt-get install nfs-kernel-server
   2. create a folder name “cloud” under mpiuser’s home directory
   3. add an entry to /etc/exports with “home/mpiuser/cloud \*(rw,sync,no\_root\_squash,no\_subtree\_check)”
   4. Reapply /etc/exports by running “exportfs -a”
   5. sudo service nfs-kernel-server restart
2. A shared directory must be configured on each slave cluster as well. Steps are as follows:
   1. sudo apt-get install nfs-common
   2. create a folder name “cloud” under mpiuser’s home directory (slave node)
   3. sudo mount -t nfs master:/home/mpiuser/cloud ~/cloud
   4. Step c requires that slave cluster configure /etc/hosts with ip address and identity of the master cluster and itself.
   5. add the following entry to /etc/fstab so that the mounted shared folder does not get unmounted on reboot. “master:/home/mpiuser/cloud /home/mpiuser/cloud nfs”
3. Once the basic environments are setup, the mpi program can be executed. Methods are as follows:
   1. mpirun -np 1 (no. of cores) -hosts master,client (identities in /etc/hosts) ./mpi\_executable
   2. mpirun -np 1 (no. of cores) ./mpi\_executable # Running on the master cluster only.
   3. mpirun -np 1 (no. of cores) --hostfile /etc/hosts ./mpi\_executable
4. Following is current /etc/hosts file on the master cluster:

127.0.0.1    localhost

#127.0.1.1  master\_identity

192.168.0.31    master

192.168.0.32    client

192.168.0.33    client2

**Basic setup on the client side:**

**1.** Install build essentials for mpich dependencies: “sudo apt-get install build-essentials”

**2.**  Install mpich using apt-get. Using a version from another source can cause errors (see error 3): “sudo apt-get install mpich” This results in installing version 3.3a2 in contrast to version 3.3.1 available from other sources.

**3.** Create a user on the client side for mpich: “sudo addusser” followed by a username.

**4.** Create a shared folder for both master and client machines to use: “mkdir” followed by the folder name.

**5.** Install vim or another editor to edit files: “sudo apt-get install vim”.

**6.** Edit the /etc/hosts file to include the master and client IP addresses.

**7.** Install openssh on the client: “sudo apt-get install openssh-server”

**8.**  Install nfs: “sudo apt-get install nfs-common”

**9.** Mount the shared folder: “sudo mount -t nfs master:/home/mpiuser/cloud ~/cloud

**Common Errors and Fixes:**

1. When encountering the following messages,

**control\_cb (pm/pmiserv/pmiserv\_cb.c:200): assert (!closed) failed**

**HYDT\_dmxu\_poll\_wait\_for\_event (tools/demux/demux\_poll.c:76): callback returned error status**

**HYD\_pmci\_wait\_for\_completion (pm/pmiserv/pmiserv\_pmci.c:198): error waiting for event**

**main (ui/mpich/mpiexec.c:344): process manager error waiting for completion**

it is likely that a slave node is a virtual cluster. I have not been able to fix the problem; however, we can solve the problem by setting the slave node on a physical machine.

2. When encountering the following messages,

**HYD\_pmcd\_pmi\_alloc\_pg\_scratch (pm/pmiserv/pmiserv\_utils.c:527): failed to allocate -60 bytes**

**HYD\_pmci\_launch\_procs (pm/pmiserv/pmiserv\_pmci.c:108): error allocating pg scratch space**

**main (ui/mpich/mpiexec.c:336): process manager returned error launching processes**

it is likely that the command has been mistyped. For instance,

“mpirun -np 5 -hosts master,client2 ./mpi\_hello” will not generate the error message but

“mpirun -np -5 -hosts master,client2 ./mpi\_hello” will generate the error message.

3. When encountering the following messages,

**bash: /usr/bin/hydra\_pmi\_proxy: No such file or directory**

it is extremely likely that mpirun,mpiexec, and any mpi related programs are not located in an identical path.

For instance, a master’s mpich was installed via apt-get and a slave’s mpich was installed via mpich.org. If mpich is installed via the repository, all the files are installed and configured at “/usr/bin/”. On the other hand, if the mpich is installed via mpich.org, all the files will be installed at “/usr/local/bin/”. Since the programs are on different paths, the error message above will be generated. Source: https://codeday.me/ko/qa/20190607/732978.html

**Correct Output:**

mpiuser@heecheonpark-MB45II7:~/cloud$ mpirun -np 4 -hosts master,client2 ./mpi\_hello

Hello world from processor heecheonpark-MB45II7, rank 0 out of 4 processors

Hello world from processor heecheonpark-MB45II7, rank 2 out of 4 processors

Hello world from processor turtle, rank 3 out of 4 processors

Hello world from processor turtle, rank 1 out of 4 processors

**Date: Jul 2nd, 2019**

Various attempts were made to get hpjava working. Despite failure, this entry illustrates what has been done to run hpjava.

Although the best environment to compile and implement hpjava is linux distros prior to 2007, it is extremely cumbersome to image such old OS images on modern hardware. The alternative way would be virtualization but most of images before 2007 were separated into multiple iso files which requires sequential installation and therefore I was not able to virtualize with my current technical skill.

Here are few steps taken to get as close to get hpjava working.

1. Get a linux working, best if the os was released between 2000~2007
2. Installed jdk 1.5.6, 8, 16, 20, 22 and 1.6.45 and none of the jdk were fully compatible with hpjava either because of the compatibilities with modern operating system or because of deprecated classes in java or hpjava.
3. All jdk can be installed at <https://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase5-419410.html>
4. In my case, I downloaded “linux-x64.bin” format files.
5. chmod +x manual\_linux-x64.bin && ./manual\_linux-x64.bin
6. Set default by adding installed java to “update-alternatives” entry by running
   1. sudo update-alternatives --install /usr/bin/java java /your/manual/path/java 1
   2. sudo update-alternatives --config java
   3. you can choose which java version to use.
7. For older version of java, you need javacc. Install via sudo apt-get install javacc.
8. Set variable paths. Mines were as follow:
   1. export HPJAVA\_HOME=/home/heecheonpark/hpjdk
   2. export JAVACC\_HOME=/home/heecheonpark/Downloads/javacc-master/target/
   3. export CLASSPATH=.:$HPJAVA\_HOME/classes:$HPJAVA\_HOME/classes/multithreaded
   4. PATH=$HPJAVA\_HOME/bin:$PATH
9. Removed most of the warnings but following error occurred at the last step to compile hpj file.
   1. heecheonpark@heecheonpark-MB45II7:~/hpjdk/examples/fft2d$ hpjavac Wolf.hpj
   2. Exception in thread "main" java.lang.UnsupportedClassVersionError: Bad version number in .class file
   3. at java.lang.ClassLoader.defineClass1(Native Method)
   4. at java.lang.ClassLoader.defineClass(ClassLoader.java:620)
   5. at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:124)
   6. at java.net.URLClassLoader.defineClass(URLClassLoader.java:260)
   7. at java.net.URLClassLoader.access$100(URLClassLoader.java:56)
   8. at java.net.URLClassLoader$1.run(URLClassLoader.java:195)
   9. at java.security.AccessController.doPrivileged(Native Method)
   10. at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
   11. at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
   12. at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:268)
   13. at java.lang.ClassLoader.loadClass(ClassLoader.java:251)
   14. at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:319)
10. I went through so many warnings and errors to resolve and get this far.
11. Please, if you are brilliant, try to compile and run hpjava. (It may require old hardware).

**Date: Wed Jul 17 21:53:09 CDT 2019**

Created a parallel merge sort program based on a skeleton file given by Dr. Lee.

Skeleton file implements blocked-communication strategies (MPI\_Send, MPI\_Recv)

I modified the skeleton so that arrays are initialized with random numbers between 0~100 and added merge and mergeSort functions.

For now, master splits the array and send to slave nodes.

Slave nodes merge sort received arrays. Slaves do not send the sorted arrays back to the master so it is not completed but almost done. Here is the sample result running the program.

heecheonpark@heecheonpark-MB45II7:~/MPICH$ mpirun -np 3 ./mat\_mergeSort

The number of row is 4.

The number of columns is 4.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 83.0 86.0 77.0 15.0

 93.0 35.0 86.0 92.0

 49.0 21.0 62.0 27.0

 90.0 59.0 63.0 26.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I am 2 of 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 15.0 35.0 77.0 83.0

 86.0 86.0 92.0 93.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I am 3 of 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 21.0 26.0 27.0 49.0

 59.0 62.0 63.0 90.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date: Sun Jul 21 19:59:16 CDT 2019**

I did some research and went through some tutorials about reduce. I initially had a wrong idea that the mpi\_reduce function should be explicitly called from each processor space but reading materials and tutorials gave me that it can be just called from the master once.

The prototype for MPI\_Reduce is as follow:

MPI\_Reduce(

    void\* send\_data,

    void\* recv\_data,

    int count,

    MPI\_Datatype datatype,

    MPI\_Op op,

    int root,

    MPI\_Comm communicator)

where MPI\_Op could be:

* MPI\_MAX - Returns the maximum element.
* MPI\_MIN - Returns the minimum element.
* MPI\_SUM - Sums the elements.
* MPI\_PROD - Multiplies all elements.
* MPI\_LAND - Performs a logical *and* across the elements.
* MPI\_LOR - Performs a logical *or* across the elements.
* MPI\_BAND - Performs a bitwise *and* across the bits of the elements.
* MPI\_BOR - Performs a bitwise *or* across the bits of the elements.
* MPI\_MAXLOC - Returns the maximum value and the rank of the process that owns it.
* MPI\_MINLOC - Returns the minimum value and the rank of the process that owns it.

A little program I made is that it takes a program argument, a number of random numbers, and populate an array that takes the program argument as an index size. As random number is based on seed value from time() function, each processor reduces with MPI\_SUM with different random numbers.

Here is the output for running my program:

heecheonpark@heecheonpark-MB45II7:~/MPICH$ mpirun -np 4 ./mpi\_reduce 5

Local sum for process 0 - 359.00, avg = 71.80

Local sum for process 1 - 166.00, avg = 33.20

Local sum for process 3 - 315.00, avg = 63.00

Local sum for process 2 - 185.00, avg = 37.00

Total sum 1025.00, avg = 51.25

**Wed Sat 27 20:27:20 CDT 2019**

Worked on presentation about APIs for MPI including mpiJava, MPJ, MPJ Express, and mpi4py.

Installed mpi4py on a personal computer.

mpi4py requires proper MPICH environment. If a machine has MPICH environment, following can be done to get working mpi4py on a Linux machine.

sudo apt-get install libopenmpi-dev

sudo python3 -m pip install mpi4py

or

sudo pip3 install mpi4py

pip3 should be installed by default but there are some cases in which it is not installed by default. If that is the case, issue the following command.

sudo apt-get install python3-pip.

**Wed Jul 31 22:24:49 CDT 2019**

Worked on basic point 2 point communication with mpi4py. I was expecting that it was going to be very easy to make a simple program but wasn’t expecting that figuring out numpy would take some time. There are a multitude of declaring arrays in numpy and here are few keywords that a lot of numpy functions use.

size: length of an array

dtype: data type of an array. i.e. numpy.int8 or numpy.float64 and so forth

shape: (m, n)

for example:

>>> np.zeros((5,), dtype=int)

array([0, 0, 0, 0, 0])

>>> np.zeros((5,3), dtype=float)

array([0.0, 0.0, 0.0, 0.0, 0.0],

          [0.0, 0.0, 0.0, 0.0, 0.0],

          [0.0, 0.0, 0.0, 0.0, 0.0])

Hence, I am not used mpi4py yet and I think it has a lot of constraints compared to MPICH.

For example, Send and Recv functions do not take a size of data.

Consider an array, arr[] = {1,2,3,4,5}. If I would like to send the first 3 elements in the array,

MPICH = MPI\_Send(arr, 3, MPI\_INT, dest, tag);

mpi4py = COMM\_Send([arr[:3], MPI\_INT], dest, tag)

As seen above, the array must be sliced by the amount of data to send.

Furthermore, the fact that Python is an OO language makes a few distinctions compared to C.

COMM\_send is used if the data to send is Python object. (Python list is an object)

COMM\_Send is used if the data is a memory buffer. (Numpy array is a memory buffer)

**Sat Aug  3 11:03:59 CDT 2019**

I have tried a few tweaks to see if “recv” can be used for memory buffer and found that Python interpreter complains with following message:

When “Send and recv” are used,

*test\_mpi4py.py:45: UserWarning: the 'buf' argument is deprecated*

*COMM.recv([receivedMat2, MPI.INT], source=MASTER, tag=1)*

When “send and Recv” and “send and recv” are used,

*TypeError: can't pickle mpi4py.MPI.Datatype objects*

Likewise, I have tried send, recv, Send, and Recv on Python list object.

When “send and Recv” is used,

*TypeError: Recv() takes at least 1 positional argument (0 given)*

As seen above, “send” was successful, but due to Recv takes different parameters, it cannot be used.

Prototypes:

COMM.Send([data[position], datatype], destination, tag)

COMM.send(data, destination, tag)

COMM.Recv([data[position], datatype], source, tag)

COMM.recv(source, tag) <- no argument to receive data

mpi4py can automatically detect datatype when using Send and Recv like:

COMM.Send(data, destination, tag)

COMM.Recv(data, destination, tag)

Conclusion:

Send and Recv are more compliant to the standard MPI while send and recv are additional mpi4py feature to communicate with Python objects.

**Sat Aug  4 16:04:44 CDT 2019**

Today, I worked on Bcast and bcast from mpi4py. Just like “Send and Recv” and “send and recv”, correct functions should be used depending on whether the data to broadcast is a memory buffer or Python pickleable object.

Just like MPI\_Bcast, the Bcast function must be used on all processor. For example, following use of Bcast is incorrect.

COMM = MPI.COMM\_WORLD

RANK = COMM.Get\_rank()

SIZE = COMM.Get\_size()

if RANK == MASTER:

     mat = np.arange(1,17,dtype='i').reshape(4,4) #numpy array

else:

     mat = np.zeros((4,4), dtype='i') #numpy array

if RANK == MASTER:

     COMM.Bcast(mat, MASTER)

Instead, following is the correct use,

COMM = MPI.COMM\_WORLD

RANK = COMM.Get\_rank()

SIZE = COMM.Get\_size()

if RANK == MASTER:

     mat = np.arange(1,17,dtype='i').reshape(4,4) #numpy array

else:

     mat = np.zeros((4,4), dtype='i') #numpy array

     COMM.Bcast(mat, MASTER)

Here is a sample output of my program.

Initial matrix at processor at: 0

[[ 1  2 3 4]

 [ 5  6 7 8]

 [ 9 10 11 12]

 [13 14 15 16]]

Broadcasting the matrix to all processor from 0

Initial matrix at processor at: 1

[[0 0 0 0]

 [0 0 0 0]

 [0 0 0 0]

 [0 0 0 0]]

Broadcasting the matrix to all processor from 0

Initial matrix at processor at: 2

[[0 0 0 0]

 [0 0 0 0]

 [0 0 0 0]

 [0 0 0 0]]

Broadcasting the matrix to all processor from 0

Final matrix after the broadcast from processor: 1

[[ 1  2 3 4]

 [ 5  6 7 8]

 [ 9 10 11 12]

 [13 14 15 16]]

Final matrix after the broadcast from processor: 2

[[ 1  2 3 4]

 [ 5  6 7 8]

 [ 9 10 11 12]

 [13 14 15 16]]

**Mon Aug  5 15:45:00 CDT 2019**

Going to log about scatter and gather.

**Sun Aug 11 18:52:39 CDT 2019**

Had some problems in calculating execution time of 1000 x 1000 matrix multiplication on numpy array with my matrix multiplication function. I have always noticed that accessing numpy array takes more time that list which is somewhat counterintuitive. What I have found is that accessing numpy array takes much more complex instruction as the way Python accessing list is fundamentally different from accessing numpy array since numpy array is wrapped in C.

Two other reasons why a[100][100] (getting from the list) is quicker than b[100,100] (getting from the numpy array) are that:

* The bytecode opcode BINARY\_SUBSCR is executed when indexing both lists and arrays, but it is optimised for the case of Python lists.
* The internal C function handling integer indexing for Python lists is very short and simple. On the other hand, NumPy indexing is much more complicated and a significant amount of code is executed to determine the type of indexing being used so that the correct value can be returned.

Additionally, I have managed to make Joshua’s cpp benchmark program to work with 1000x1000 matrix. His program works with 100x100 but when running the program with 1000x1000, we kept having segmentation fault. After a little bit of research, we have found that C++ cannot hold large array on stack. If we statically allocate memory on an array, we are allocating the memory on stack and to alleviate the situation, dynamic allocation is necessary.

One of ways to allocate 2d arrays in C++ is as follow:

long long int \*\* list\_mat;

list\_mat = new long long int\*[1000];

for (int i = 0; i < 1000; i++)

{

     list\_mat[i] = new long long int[1000];

}

// **Very IMPORTANT**.

// Make sure to **DEALLOCATE ALL**.

for (int i = 0; i < 1000; i++)

{

     delete [] list\_mat[i];

}

delete [] list\_mat;