**********Draft**********

20 Core RPi4B, RPi3B, and RPi3B+
MPICH Cluster
04/09/20

**********Draft**********

The software used ***"mpich-3.3.2.tar.gz"*** was downloaded from "http://www.mpich.org/downloads/". This was compiled and installed **"/home/devel/mpi/local"** on a RPi4B.

With rsync was copied to the other 4 nodes.

Step1.
devel@mypi3-15:~/mpich-3.3.2 $ tar xvf mpich-3.3.2.tar.gz

Step2.
cd mpich-3.3.2
./configure –prefix=/home/devel/mpi/local

Step3.
make
make install

Step4.
Install ssh keys on all nodes. This will allow login without having to provide a pass word.
The commands used were "**ssh-keygen -t rsa"** & "**ssh-copy-id mypi3-xx"**.

Step5.
This that the software was built.
mpiexec --version

mpiexec (OpenRTE) 3.1.3

Step6.

rsync -avl –delete mpi mypi3-xx:~/

rsync -avl –delete mpich-3.3.2 mypi3-xx:~/

Step7.
Need provide a script that will provide a PATH to where the mpi software was installed.

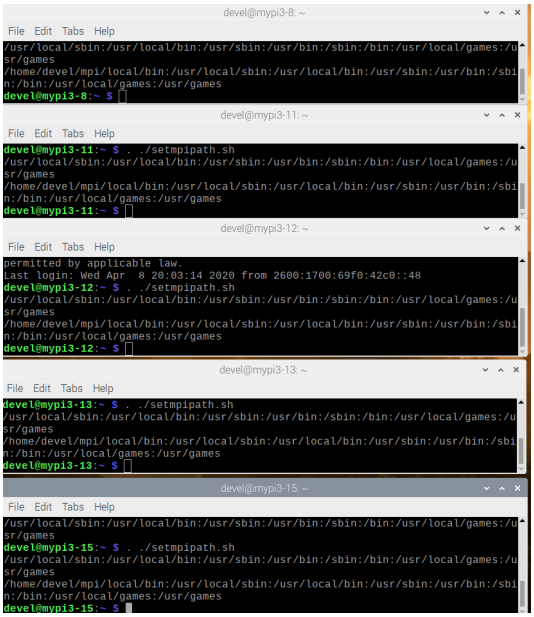
The file **"setmpipath.sh"**

```
#!/bin/bash
echo $PATH
export PATH=/home/devel/mpi/local/bin:$PATH
echo $PATH
```

The file **“machinefile” is used to ID the systems in the cluster and the number of jobs.**

mypi3-15:4
mypi3-13:4
mypi3-8:4
mypi3-11:4
mypi3-12:4

The image below is initizing all the nodes.



The image below is testing hellow all the nodes.

```
File  Edit  Tabs  Help
user    0m0.546s
sys     0m0.259s
devel@mypi3-15:~ $ time mpiexec -f machinefile -n 20 mpich-3.3.2/examples/hellow
Hello world from process 12 of 20
Hello world from process 15 of 20
Hello world from process 13 of 20
Hello world from process 14 of 20
Hello world from process 16 of 20
Hello world from process 19 of 20
Hello world from process 17 of 20
Hello world from process 18 of 20
Hello world from process 8 of 20
Hello world from process 9 of 20
Hello world from process 10 of 20
Hello world from process 11 of 20
Hello world from process 5 of 20
Hello world from process 0 of 20
Hello world from process 6 of 20
Hello world from process 3 of 20
Hello world from process 1 of 20
Hello world from process 4 of 20
Hello world from process 7 of 20
Hello world from process 2 of 20

real    0m1.524s
user    0m0.620s
sys     0m0.270s
devel@mypi3-15:~ $
```

In the example below the program cpi uses 10000 steps to compute the value of Pi.

```
File  Edit  Tabs  Help
devel@mypi3-15:~ $ time mpiexec -f machinefile -n 20 mpich-3.3.2/examples/cpi
Process 16 of 20 is on mypi3-12
Process 17 of 20 is on mypi3-12
Process 18 of 20 is on mypi3-12
Process 19 of 20 is on mypi3-12
Process 4 of 20 is on mypi3-13
Process 5 of 20 is on mypi3-13
Process 6 of 20 is on mypi3-13
Process 7 of 20 is on mypi3-13
Process 1 of 20 is on mypi3-15
Process 3 of 20 is on mypi3-15
Process 8 of 20 is on mypi3-8
Process 2 of 20 is on mypi3-15
Process 9 of 20 is on mypi3-8
Process 10 of 20 is on mypi3-8
Process 11 of 20 is on mypi3-8
Process 0 of 20 is on mypi3-15
Process 12 of 20 is on mypi3-11
Process 14 of 20 is on mypi3-11
Process 15 of 20 is on mypi3-11
Process 13 of 20 is on mypi3-11
pi is approximately 3.1415926544231279, Error is 0.0000000008333347
wall clock time = 0.024480

real    0m0.909s
user    0m0.559s
sys     0m0.327s
devel@mypi3-15:~ $
```

Appendix B. Source of hellow.c

```
/* -*- Mode: C; c-basic-offset:4 ; indent-tabs-mode:nil ; -*- */
/*
 *  (C) 2001 by Argonne National Laboratory.
 *      See COPYRIGHT in top-level directory.
 */

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

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

    MPI_Init(0, 0);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("Hello world from process %d of %d\n", rank, size);
    MPI_Finalize();
    return 0;
}
```

Appendix B. Source of cpi.c

```
/* -*- Mode: C; c-basic-offset:4 ; indent-tabs-mode:nil ; -*- */
/*
 *  (C) 2001 by Argonne National Laboratory.
 *      See COPYRIGHT in top-level directory.
 */

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

double f(double);

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

int main(int argc, char *argv[])
{
    int n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264 3;
    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();

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

    h = 1.0 / (double) n;
    sum = 0.0;
    /* A slightly better approach starts from large i and works back */
    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;
}
```