JCC Software Manual

Prepared by Satyo Wasistho

April 25th, 2023

Table of Contents

MPI Installation/Configuration Steps for JCC

In this document, we will go over how to set up MPI on the JCC.

To set up a cluster in the local environment, the same versions of the **OpenMPI** should be pre-installed in every system.

Prerequisites

- **Operating System:** The Operating System is Ubuntu 18.04.
- **MPI:** We could either use OpenMPI or MPICH. The JCC currently uses OpenMPI (version 2.1.1).

```
sudo apt-get install openmpi-bin        //OpenMPI Binaries (mpirun, mpiexec, etc.)

sudo apt-get install libopenmpi-dev     //OpenMPI Library + Compilers (mpicc,
                                        //mpic++, mpifort, etc.)
```

Install the binaries on the master node and every slave node. Install the library and compilers on only the master node.

Setting Up MPI for the JCC

The following steps will guide you through configuring your MPI installation for use on the JCC. Note that the JCC's hardware setup and proprietary software tools make this process slightly different (and simpler, we hope) from how you would set up most other clusters.

Step 1: Creating a user for MPI

In order to pass executables from the master node to every slave node, we need every node to have a Linux user with the same name. The JCC currently uses *jcccluster* as its shared user for MPI. Note that this step requires physical access to each node.

To add a new user:

```
sudo adduser jcccluster
```

Making mpiuser a sudoer :

```
sudo usermod -aG sudo jcccluster
```

Step 2: Setting up SSH

First, we must install SSH onto the system, as this is how machines are going to be talking over the network. Do this for all nodes, master and slave. Note that this step requires physical access to each node.

To install SSH in the system.

```
sudo apt-get install openssh-server
```

Beyond this point, we can access each node via SSH rather than plugging into them physically. The SSH setup needs to be done through the MPI user we just created. If you are on the MPI user on the master node, you can SSH into the same user on all the slave nodes without specifying a username.

Log in to the MPI user by

```
su - jcccluster
```

Next, we must determine the node IDs of each slave connected to the network, as the DHCP server allocates IP addresses to slave nodes dynamically on startup. Fortunately, JCC developer Satyo Wasistho has developed a software package specifically for node setup on the JCC to help automate this process.

The node setup package located in **/home/jcccluster/JCCdev/node_setup/** on the head node. You can copy this directory into any slave node using **scp** like so:

```
scp -r /home/jcccluster/JCCdev/node_setup <slave node id>:<desired directory>
```

The node setup tools can be used in any directory. Before proceeding onto the next part of the SSH setup, perform the following on each node.

Run the **gen_hosts.sh** script. This gives the node a record of all nodes that *could* exist on the cluster network.

```
sudo bash gen_hosts.sh
```

Next, run the **connect_to_hosts.sh** script. This tells the node which of the possible hosts are *actually* connected to the cluster network.

```
sudo bash connect_to_hosts.sh
```

For MPI to function across all nodes, the SSH connection must be password-less. We can use RSA encryption to set up a password-less SSH connection between the master node and each slave node. Do this on all nodes.

In the home directory:

```
ssh-keygen -t rsa
cd .ssh/
cat id_rsa.pub >> authorized_keys
```

If you are on the master node, run the **mass-copy-id.sh** script.

```
bash mass-copy-id.sh
```

If you are on a slave node, simply ssh copy the master node's ID.

```
ssh-copy-id master
```

With that, you should now have MPI properly set up for use on the JCC.

Running MPI Programs on the JCC

As stated previously, the JCC developer Satyo Wasistho has developed a number of software tools in order to simplify the development, maintenance, and use of the JCC. One of these tools is **KarSlurm**, the JCC's proprietary UI wrapper for MPI. **KarSlurm** is designed to be an easy-to-use, intuitive platform for running parallel programs on the JCC, offering users fine control over process allocation across the network of nodes while automating away the more repetitive parts of setting up MPI programs to run across a distributed memory system.

Running an MPI program using **KarSlurm** is quick and easy. Just pass your executable through the *karun* command, and the UI will guide you through configuring your parallel program for execution on the JCC. This works from any directory.

```
karun <mpi_executable>
```

Adding a Slave Node to the JCC

One of the greatest appeals of any cluster computer is its modularity. The ability to upscale the capabilities of a system over the course of its lifetime with relative ease is an invaluable resource and one that the JCC development team has placed at the core of our design philosophy.

As such, we have taken measures to greatly simplify the process of integrating additional nodes to the JCC's network. This guide will explain all the necessary steps to add a slave node to the JCC and set up its software environment for executing parallel code alongside the other nodes on the network. This is assuming you have already flashed the node with Ubuntu and physically connected it to the JCC node network (as described in the hardware manual).

Software Setup

For this process, if you are adding multiple nodes in one sitting, it is best to fully complete these steps for one node before moving on to the next.

Step 1: MPI Setup

The following steps will guide you through installing and configuring MPI on the new node to match the existing MPI setup on the JCC.

Install the binaries for OpenMPI (version 2.1.1) using the following command in your terminal.

```
sudo apt-get install openmpi-bin
```

In order to pass executables from the master node to every slave node, we need every node to have a Linux user with the same name. The JCC currently uses *jcccluster* as its shared user for MPI. Note that this step requires physical access to the new node.

To add a new user:

```
sudo adduser jcccluster
```

Making mpiuser a sudoer :

```
sudo usermod -aG sudo jcccluster
```

Step 2: SSH Setup

We now have MPI installed and usable on the new node. However, the node is still unable to communicate with the rest of the JCC through MPI. On distributed memory systems like the JCC, MPI transfers data between nodes through **SSH**.

First, we must install SSH onto the new node.

To install SSH in the system:

```
sudo apt-get install openssh-server
```

Beyond this point, we can access the new node via SSH rather than plugging into it physically. From this point forward, the SSH setup needs to be done through the MPI user we just created. If you are on the MPI user on the master node, you can SSH into the same user on all the slave nodes without specifying a username.

Log in to the MPI user by

```
su - jcccluster
```

Next, we must determine the node ID of the new slave that has been connected to the network, as the DHCP server allocates IP addresses to slave nodes dynamically on startup. Fortunately, JCC developer Satyo Wasistho has developed a software package specifically for node setup on the JCC to help automate this process.

The node setup package located in **/home/jcccluster/JCCdev/node_setup/** on the master node.

First, **cd** into the node setup directory and run the **connect_to_hosts.sh** script on the master node. This script gives the master node a record of all the slave nodes currently connected to the network and specifies the node ID of any newly detected nodes that have not been previously recorded to exist on the network.

```
bash connect_to_hosts.sh
```

connect_to_hosts.sh output

```
How many worker nodes are on the network (default 100)? 4
node found: slave5
node found: slave6
```

```
node found: slave7
node found: slave8 (new)
```

The hostname that is tagged as 'new' in the script output is the node ID of the new node. Use this node ID to copy the node setup package into the new node. You can do this using **scp** like so:

```
scp -r /home/jccluster/JCCdev/node_setup <slave node id>:<desired directory>
```

The node setup tools can be used in any directory, so long as the scripts have been copied to those directories. Perform these next steps on the new slave node.

Run the **gen_hosts.sh** script. This gives the node a record of every possible node ID it can ever have (as the DHCP server dynamically allocates IP addresses to each slave node upon a system startup).

```
sudo bash gen_hosts.sh
```

For MPI to function across all nodes, the SSH connection must be password-less. We can use RSA encryption to set up a password-less SSH connection between the master node and the new slave node. Do this on both nodes.

In the home directory:

```
ssh-keygen -t rsa
cd .ssh/
cat id_rsa.pub >> authorized_keys
```

If you are on the master node, ssh copy the slave node's ID.

```
ssh-copy-id <slave node id>
```

If you are on a slave node, ssh copy the master node's ID.

```
ssh-copy-id master
```

With that, your new slave node should be properly integrated into the JCC's network.

Adding/Removing Users on the JCC

The intended purpose of the JCC is to provide an in-house, high performance computing environment for UAH students, faculty, and researchers. As such, JCC administrators will regularly need to add a large number of users at one time and set up user environments for all of them in order to grant JCC access to a UAH course, UAH instructors, or a UAH research group, and the same can be said for removing a large number of users to revoke JCC access to those groups.

On other systems, this is a *very* tedious process. However, the JCC Development Team has provided a fast and easy way to add or remove several users via the **account management** package within the JCC's proprietary software suite (courtesy of JCC developer Satyo Wasistho).

The **account management** software package is located in the **/home/jcccluster/JCCdev/account_management** directory on the JCC's master node and is only accessible to JCC administrators. This package contains two scripts, **addusers.sh** and **removeusers.sh**.

Requirements

To set up or delete a large number of user accounts, you only need a list of usernames. Store this list in a text file. The file can be named anything. For the purposes of this guide, we will refer to the file as **usernames.txt**.

usernames.txt format

```
saw0037
sfr0009
jct0025
mdo0002
wpb0003
```

Adding Users

With this file in your current directory, you can now set up the listed users on the JCC. Simply pass the **usernames.txt** file into the **addusers.sh** script as shown below.

```
sudo bash addusers.sh usernames.txt
```

This will add the listed user accounts as **standard users**. Do this if the group you are giving JCC access to is of students taking a UAH course.

For non-student users, you can specify a Linux group to add the user accounts to. The JCC already has common groups set up for this purpose. The groups are as follows:

Group	Sudo?
Users	No
Admin	Yes
Staff	Yes
Research	No

To add user accounts under one of these groups, simply add a group parameter when running the **addusers.sh** script as shown below.

```
sudo bash addusers.sh <groupname> usernames.txt
```

All user accounts are given the default password, "WelcomeToJCC". Users can change their password at any time using the **passwd** command.

Removing Users

To remove multiple users from the JCC, simply pass the **usernames.txt** file into the **removeusers.sh** script as shown below.

```
sudo bash removeusers.sh usernames.txt
```

With that, you now know how to easily add or remove several users on the JCC.

How to Run MPI Programs on the JCC using Karslurm

In this guide, you will learn how to use **Karslurm**, the JCC's proprietary job scheduling software (courtesy of Satyo Wasistho). **Karslurm** provides an accessible and intuitive mechanism for executing and monitoring parallel jobs on the JCC. Currently, **Karslurm** is comprised of three main tools, **karun**, **karqueue**, and **karkill**. This guide will explain each of these tools from the viewpoint of a standard user.

Karun

Users can execute their parallel jobs on the JCC using the **karun** command. To do so, simply compile your parallel code and pass the executable through **karun** as shown below.

```
<mpi compiler> <source file name> -o <executable name>
karun <executable name>
```

mpi compiler: the name of the mpi compiler for the language of your choice. (mpicc, mpic++, mpif77, etc.)
source file name: the name of the file containing your source code (the code that you want to compile and run)
executable name: the name of the executable file you want to generate

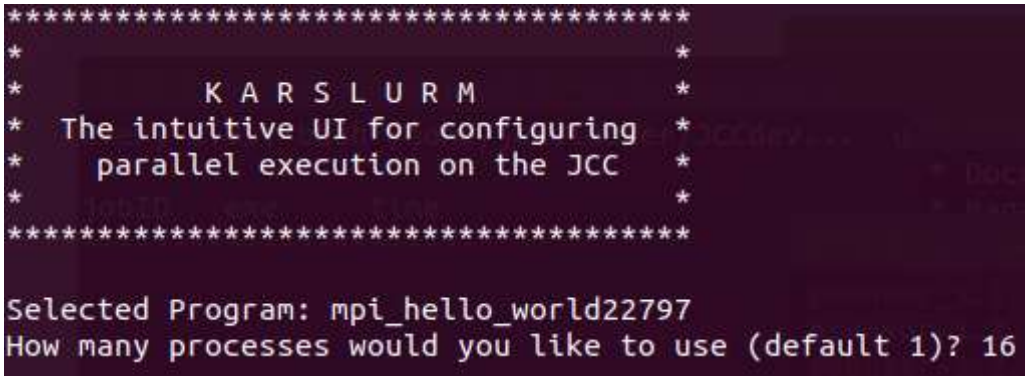
Here is an example using a simple *n*-process "hello world" MPI program.

```
mpicc mpi_hello_world.c -o mpi_hello_world
karun mpi_hello_world
```

If your parallel code makes use of command line arguments, you can pass those into **karun** as well.

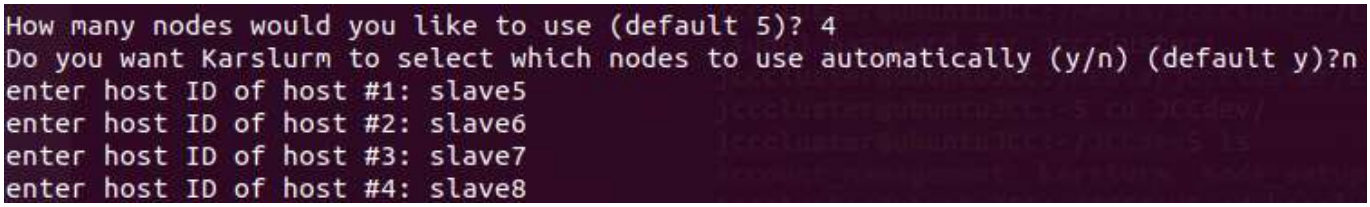
```
karun <executable name> <arg1 arg2 ... argX>
```

From here, you will customize your parallel job through the **Karslurm UI** that appears on your terminal. First, it will ask you the number of processes you would like to run. Without an input, the job configuration defaults to one process.

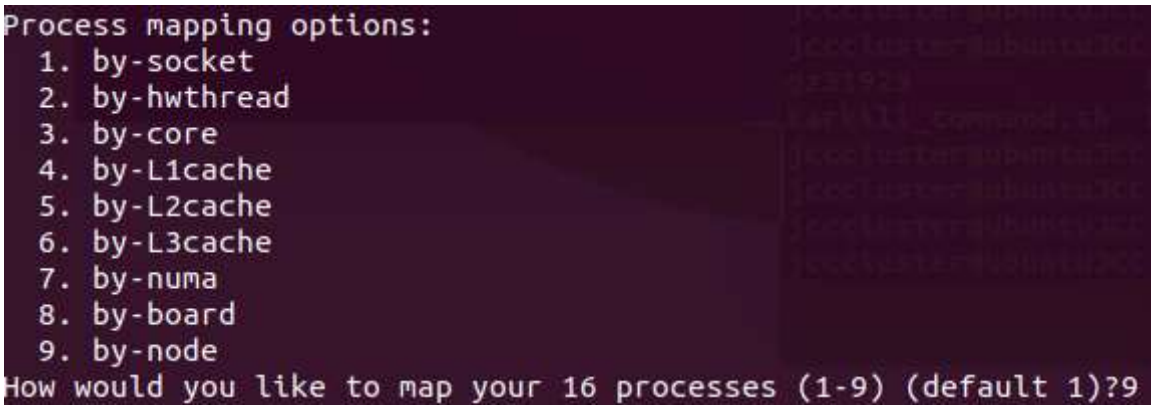


Next, you will be prompted for how many nodes you would like to run your job on. Without an input, the job configuration defaults to either the number of processes ran or the total number of nodes connected to the network (whichever is lower).

In cases where it is meaningful, the **Karslurm UI** will ask whether you want to select which nodes are used or have them be autoselected by **Karslurm**.



After this, you will get to select the mapping option for your parallel job. This affects how processes assigned to nodes (which nodes run which processes).



This completes the job customization. Now, **Karslurm** will send out the executable to all nodes set to run the job and execute the job on the set nodes and processes.

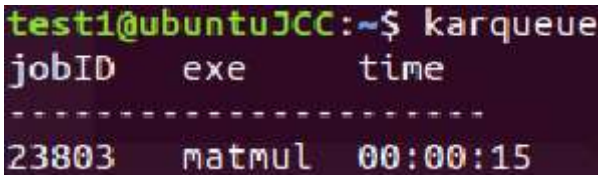
```
sending executable mpi_hello_world22797 to external nodes.
        executable mpi_hello_world22797 sent to slave5
        executable mpi_hello_world22797 sent to slave6
        executable mpi_hello_world22797 sent to slave7
        executable mpi_hello_world22797 sent to slave8
executing 'mpi_hello_world22797' with 16 process(es) on 4 node(s)...
```

Upon completion of your parallel job, **Karslurm** will send the output to your working directory. View the generated output file to see the node-process mapping and the job output.

```
test1@ubuntuJCC:~$ cat mpi_hello_world8961.kout
 Data for JOB [5362,1] offset 0

 ========================   JOB MAP   ========================

 Data for node: slave5  Num slots: 4    Max slots: 0    Num procs: 4
        Process OMPI jobid: [5362,1] App: 0 Process rank: 0 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 4 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 8 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 12 Bound: UNBOUND

 Data for node: slave6  Num slots: 4    Max slots: 0    Num procs: 4
        Process OMPI jobid: [5362,1] App: 0 Process rank: 1 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 5 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 9 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 13 Bound: UNBOUND

 Data for node: slave7  Num slots: 4    Max slots: 0    Num procs: 4
        Process OMPI jobid: [5362,1] App: 0 Process rank: 2 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 6 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 10 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 14 Bound: UNBOUND

 Data for node: slave8  Num slots: 4    Max slots: 0    Num procs: 4
        Process OMPI jobid: [5362,1] App: 0 Process rank: 3 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 7 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 11 Bound: UNBOUND
        Process OMPI jobid: [5362,1] App: 0 Process rank: 15 Bound: UNBOUND

 =============================================================
Hello world from process 6 of 16
Hello world from process 3 of 16
Hello world from process 12 of 16
Hello world from process 14 of 16
Hello world from process 7 of 16
Hello world from process 8 of 16
Hello world from process 1 of 16
Hello world from process 10 of 16
Hello world from process 11 of 16
Hello world from process 0 of 16
Hello world from process 5 of 16
Hello world from process 2 of 16
Hello world from process 15 of 16
Hello world from process 4 of 16
Hello world from process 13 of 16
Hello world from process 9 of 16
```

Karqueue

For longer jobs, you may want to monitor the progress of your job. You can do this using the **karqueue** command. This command shows every active karslurm job running on the JCC, detailing its job ID, executable name, and current execution time.



Additionally, running **karqueue** with the **real-time monitoring** flag (-rt) shows the job queue in real time, allowing you to see the exact moment your job starts and ends without having to rerun the command multiple times.

Karkill

There may be times when your parallel job takes longer to execute than you expect, and you would like to terminate the job. This can be done using the **karkill** command.

As shown previously, **Karslurm** jobs are assigned a job ID when they are run. You can view the ID of your job through **karqueue**. To kill a job, simply pass the job ID through the **karkill** command as shown below.

```
karkill <job ID>
```