Jetson Charger Cluster-Next Gen Hardware Manual

Table of Contents

Introduction

The Jetson Charger Cluster-Next Gen (JCC) is a hybrid team composed of Computer Engineering students: Jordan Thomas, Karson Knoll, William Bullard, Satyo Wasistho, Michael O'Dell and a Cybersecurity Engineer: Svetlana Freeman. The goal of this project is to create a relatively powerful and inexpensive high performance computer capable of parallel computing made from off the shelf components. The purpose of this computing machine is so that students who take CPE 412/512 have access to an up to date parallel computing machine that they can access on their own time, is inexpensive for the department to maintain, and allows students to experiment with node control and how to optimally run their programs in a parallel computing environment. Something they do not get with the current class setup of them just using the Alabama Supercomputer. It automizes the programs and their execution for them, cutting out an important part of their parallel computing education. Our solution, a Beowulf style cluster made with 5 Jetson 2GB development boards, a left over switch the school had in a storage room that was not in use, open source software, and an inexpensive 3D printed case. All of which by the end of this project will come in well below our possible competitors prices. We will also have a cybersecurity risk management/mitigation plan in effect to insure that we do not add any undue risk to the school and its resources.

Proposed solution/Scope

In consideration with the intended scope of this hardware manual we will cover the general problem proposed for the hardware team: Jordan Thomas, William Bullard, and Michael O'Dell; and the solution decided upon in order to solve the identified problem. The problem set forth for the hardware team was to set up the hardware components of a cluster computer in a beowulf configuration using boards provided by our sponsor and to create an enclosure for the switch box and boards to rest within for viewing purposes. Addressing this issue the hardware team decided to use five Jetson Nano 2GB Dev boards, defining one as a head node and the other four as computational nodes which will all communicate through a switch box provided to us. Our solution for creating an enclosure environment is setting the system in a server rack with the actual boards resting in a 3d printed rig that has fans affixed to each of the boards' rack positions or cooling and a usb power supply to power each of the boards. The switch box connecting all of the boards to the head will rest beneath the boards in the rack and will be connected out of the front of the enclosure.

Hardware Described in this manual

1. SanDisk 32 GB Micro SD
2. Jetson Nano 2GB Dev Boards
3. Ethernet Cables
4. Network Switch
5. Jetson Power Cable
6. Jetson interface Cables

Interfacing with the cluster

The method to interface with the hardware is to execute the karun command with the mpi program executable input. From there follow the onscreen prompts to configure how the code should execute on the cluster.

From Scratch Setup

1. The first step is to flash each of the boards with their respective operating system and perform the initial boot setup sequence
2. The next step is to connect each board to the internet so that the installation of each of the packets needed
3. The next step is to perform an apt-update that will then begin installing all of the things needed
4. The next step is to then install Slurm utilizing apt on each board
5. The next step is to then install OpenMpi and SSH on each board
6. The next step is to then select a board to be the “Head node”
    a. This will serve as the host for the DHCP server
7. The next step is to install DHCP onto the selected head node
8. From here you will need to create a copy of the DHCP config file
9. After creating the copy the next step is to go into the config file and begin configuring the DHCP server
    a. The first step of this is to open the config file using your preferred text editor
    b. Next, set the DHCP such that it is authoritative
    c. The next step is to set the server’s ip
    d. The next thing is to then set a range of the valid ips that the server can hand out
    e. After this the next step is to statically allocate the host “head” ip to the ethernet port on the “head” node
    f. Once this is completed a reboot is needed
    g. Once reboot is completed perform a restart of the dhcp server using the systemctl command
    h. Check the status of the server using the same command mentioned above
    i. If there issues please reference the troubleshooting portion of this manual.

10. After completing the DHCP server setup the next step is to perform a ping onto each of the boards on the network

Troubleshooting/Seen Issues

1. Issues arose when flashing the dev boards with their images as the image would consistently load on to the sd cards with errors. The solution to this problem was found through brute force testing as through repeated trial and error the image would load on to the sd cards which would allow us to boot the dev boards properly. It was also found that windows 10 systems were more likely to have no issues with the loaded image.
2. SLURM was experiencing issues with interfacing with OpenMP and how it manages its power draw. The issue currently still persists, but a solution was found by using OpenMPI directly without the use of SLURM. This makes things a little more challenging as far as scheduling workloads goes, but the project was under a strict timeline.
3. DHCP issues and troubleshooting
    a. If the server is having configuration issues, check the isc-dhcp.conf configuration file to make sure the configuration file is set up correctly.
    b. If the server is having connection issues make sure to reboot the system after configuring the server.
        i. This ensures that the network manager is rebooted
        ii. Another method is to reboot the network manager through the terminal
    c. If a node is having connection issues
        i. Check the network settings related to port eth0 and see if the ip configuration is set to Automatic (DHCP)
        ii. Execute the ifconfig command and seeing if port eth0 is running when connected to the switch
            1. Check connections
            2. Utilize the ifup command to power on the port or use the ubuntu network configuration screen and power on the port
        iii. Check to see if a static ip is set on the node and ensure the ipv4 address set is with in the range of the DHCP server

Enclosure Design:

We looked at a few different enclosure options for the Jetson Charger Cluster-Next Gen, but in the end, we chose to design our own 3D printed enclosure. This provided a number of benefits, but the main driving factor for us was the ability to customize the enclosure to be exactly what we wanted. We knew that the enclosure would need to mount in some way close to the large rack-mounted networking switch that we were provided for our cluster. Since the networking switch of the cluster was rack-mounted, we decided that designing our enclosure to be rack mounted would be the optimal solution in terms of protection, usability and ease of maintenance.

In the start of the design phase of the enclosure, we knew that we would need to house at least 5 Jetson Nano boards. The overall size of 5 jetson nano boards together was larger than the print volume of the 3D printer that we had access to, so we chose to design our cluster in separate modular cages. Each cage would house a single Jetson Nano board, and these cages would interlink together to form the overall enclosure. In the beginning, we also thought that we would need to accommodate two different boards: Jetson Nano boards, and ODroids. Due to the need for being able to swap between these boards, we decided that a separate sled to mount the boards to would be ideal. This sled would then be able to slide in and out of the enclosure cage to make switching between two different boards easier.

The first design we constructed was the sled that the Jetson Nano board would mount to. We ensured that the holes all lined up and that it fit the overall footprint of the board and was firm enough to safely hold the Jetson Nano board. From this, we designed the enclosure cage to fit the overall dimensions of both the Jetson Nano board and the ODroid board. Then the sliding interface between the board sled and enclosure was implemented. This required a good bit of tolerance testing and many test prints to achieve a good fit that held the board sled in place and didn't fall out, but wasn't too difficult to insert. Our initial sliding interface used a rectangular slot between the board sled and enclosure cage. After extensive testing, we realized that a rectangular sliding interface incurred too much friction on the 6 overall planes that were in contact between the board sled and enclosure cage. We later revised the sliding interface to be a triangular slot, which decreased our planes of contact to 4 and greatly reduced the friction between the board sled and enclosure cage.

Once we had a good working enclosure cage and sled for a single Jetson Nano board, we began to modify the enclosure cage design to allow for them to be mounted together with additional enclosure cages. We decided to mount them together using threaded rods through 4 holes in each enclosure cage. We also added tabs on one side of each enclosure cage to match slots on the other side of the enclosure cage to further strengthen the connection between each cage. Because our desired final complete enclosure would be rack-mounted, we decided to use 10 total enclosure cages to span the standard width of a server rack. The enclosure was also designed to fit within a standard 3U server space. For this, we designed brackets to mount the enclosure cages to the server rack.

Along the way, we learned quite a few lessons. We began printing using PLA as our material, but once we got a better understanding of the thermal characteristics of the Jetson Nano boards, we switched to PETG for our 3D printing material. Along with this change, we ran a few thermal simulations and decided to redesign our enclosure cages to include fans to provide forced airflow across the Jetson Nano heatsink. Although our simulations indicated that the Jetson Nano boards and enclosure would operate perfectly without forced airflow, we decided that fans would improve the long-term reliability of the Jetson Charger Cluster-Next Gen. We also decided to revise the board sleds to include handles on them in later revisions to make removal of the board sleds from the enclosure cages easier. Lastly, if we were to

improve on this design in the future, we would provide more internal support for each enclosure cage to ensure that each board sled operated as well as the others.

References

1. Spec sheets/User guide
    a. Jetson Spec Sheet
        i. https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide
    b. Switch User Guide
        i. https://dl.dell.com/manuals/all-products/esuprt_ser_stor_net/esuprt_powerconnect/powerconnect-5424_user's%20guide_en-us.pdf
2. DHCP setup guide
    a. https://www.blackmoreops.com/2015/03/26/setup-dhcp-or-static-ip-address-from-command-line-in-linux/
    b. https://www.linuxfordevices.com/tutorials/ubuntu/dhcp-server-on-ubuntu