

Team 1 - Project 3

Edmond Zalenski, George Karchenko, George Poulos, Jeff Kaleshi, Kevin Liu , Kunal Shah, Tim Choh

Versions

Name	Version
Hypriot	v1.6
Kubernetes	v1.8.3
Flannel	v0.9
Hadoop	v2.8.2

Individual Raspberry Pi

For this project we utilized the Raspberry Pi 3 (RPi3) which runs on a 64 bit ARM architecture and we paired it with a 32gb microSD storage solution. Additionally, a heat sink was required in order to provide adequate cooling to the Raspberry Pi when under heavy load. The RPi3 will have power provided from a 12 amp USB hub and wall sockets in order to ensure consistent current and power is provided to the pie. As for networking, the pie will be connected to the internet via an ethernet cable which connects to the router. Lastly, the RPi3 will have Hypriot OS and Kubernetes installed on the microSD card.

Octa-Pie Cluster

The cluster made for this project consists of 8 RPi3's. Each of which follows the specifications defined above. We 3D printed two racks where each rack holds four RPi3's. The racks are designed in such a way that there are four RPi3's stacked on top of each other with enough clearance for each RPi3 to not touch the one above or below it. Each RPi3 has one heat sink on the CPU to enable better airflow to cool down the RPi3's due to their close positioning on the rack. Their close positioning allows for us to be able to connect all the RPi3's to a power source, which consists of two USB hubs with eight USB ports at a total of 60W or 12A for each hub. To connect each RPi3 via ethernet connection, we are utilizing a 10/100 Mbps connection router. We chose not to go with a wireless connection due to the lack of reliability from experiences with Arduinos in previous classes.

Deploying Map/Reduce

In order to deploy our Map/Reduce (M/R) program onto the Octa-Pie cluster, there are a couple of steps we had to take involving various services. Firstly, after we had the Octa-Pie Cluster physically set up on the racks and connected via ethernet, we had to load an operating system onto the cluster. For each RPi3 we installed Hypriot OS and on top of Hypriot we installed Kubernetes as a container manager. Additionally, assigned each container an IP address using Flannel. Thus, the assigned Kubernetes master node will thereafter communicate

with the other nodes within the cluster in order to successfully run our M/R implementation with other nodes once we upload it. Lastly, Hadoop also had to be installed on our cluster in order for the cluster to successfully run our M/R implementation.

Descriptions of steps taken and commands required are provided below.

Commands to Init Cluster

[Source for initializing cluster](#)

```
# Initialize cluster on master, save the args for the kubeadm join command
outputted
$ sudo kubeadm init --pod-network-cidr 10.244.0.0/16 # cidr is for flannel
# Save kubernetes config file
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
$ echo "export KUBECONFIG=${HOME}/.kube/config" >> ~/.bashrc
$ source ~/.bashrc
# Add flannel related resources to cluster
$ curl -sSL
https://raw.githubusercontent.com/coreos/flannel/v0.9.0/Documentation/kube-
flannel.yml | sed "s/amd64/arm64/g" | kubectl create -f -
# On all nodes flannel related traffic will need to be forwarded, it
# is recommended to use something like netfilter-persistent to save
# the rules between sessions.
$ sudo iptables -P FORWARD ACCEPT
$ sudo iptables -t nat -A POSTROUTING -s 10.244.0.0/16 ! -d 10.244.0.0/16 -j
MASQUERADE
$ sudo iptables -I FORWARD 1 -i cni0 -j ACCEPT -m comment --comment "flannel
subnet"
$ sudo iptables -I FORWARD 1 -o cni0 -j ACCEPT -m comment --comment "flannel
subnet"
# Now on each of the other nodes run the kubeadm command saved earlier
$ sudo kubeadm join $ARGS
# Check to see them back on master node
$ kubectl get nodes
```

At this point all the nodes should have joined the cluster. The output should look something like this

Name	Status	Roles
master	Ready	master
slave1	Ready	<none>
slave2	Ready	<none>
slave3	Ready	<none>
slave4	Ready	<none>

slave5	Ready	<none>
slave6	Ready	<none>

If the nodes aren't all ready yet, don't panic. It takes a few minutes for their pods to come online. You can check pod status with `kubectl get pods --all-namespaces`

```
# Now we'll start the hadoop services on the master
$ git clone https://github.com/georgek42/rpi-hadoop-wip.git
$ cd rpi-hadoop-wip
$ for file in $( ls | grep service ); do kubectl create -f $file; done
$ for file in $( ls | grep controller ); do kubectl create -f $file; done
# Wait until all the pods are up
$ kubectl get pods
```

Name	Ready	Status
hadoop-master-controller-XXXXX	1/1	Running
hadoop-slave1-controller-XXXXX	1/1	Running
hadoop-slave2-controller-XXXXX	1/1	Running
hadoop-slave3-controller-XXXXX	1/1	Running
hadoop-slave4-controller-XXXXX	1/1	Running

Now we are ready for the next step.

Running a Map Reduce Job

```
# Enter the master pod
$ kubectl exec -it $(kubectl get po | grep master | awk '{ print $1 }') -- /bin/bash

# Add slave ips to /etc/hosts
root@hadoop-master:~$ for slave in $( cat $HADOOP_HOME/etc/hadoop/slaves ); do
echo "$(nslookup $slave | grep -m2 Address | tail -n1 | awk '{ print $2 }')
$slave" >> /etc/hosts; done;

# restart sshd and start hadoop
root@hadoop-master:~$ service ssh restart
root@hadoop-master:~$ ./start-hadoop.sh
# Run map reduce job
root@hadoop-master:~$ hadoop jar $PATH_TO_JAR $ARGS
# success!
root@hadoop-master:~$ hdfs dfs -cat output/part-00000
```

```

17/11/17 22:48:35 INFO Input.FileInputFormat: Total input files to process : 2
17/11/17 22:48:36 INFO reduce.JobSubmitter: number of splits:2
17/11/17 22:48:36 INFO reduce.JobSubmitter: Submitting tokens for map: job-1518958888686_0001
17/11/17 22:48:42 INFO Input.VarLenIntInput: Submitted application application-1518958888686_0001
17/11/17 22:48:42 INFO mapreduce.Job: Job url: track the job file://adoop-master18831:8021/job/application-1518958888686_0001/
17/11/17 22:48:42 INFO mapreduce.Job: Running job: job-1518958888686_0001
17/11/17 22:48:42 INFO mapreduce.Job: Job url: track the job file://adoop-master18831:8021/job/application-1518958888686_0001/
17/11/17 22:49:17 INFO mapreduce.Job: map 0% reduce 0%
17/11/17 22:49:24 INFO mapreduce.Job: map 100% reduce 0%
17/11/17 22:49:24 INFO mapreduce.Job: map 100% reduce 100%
17/11/17 22:49:24 INFO mapreduce.Job: Job: job-1518958888686_0001 completed successfully
17/11/17 22:58:54 INFO mapreduce.Job: Counters: 58
File System Counters
FILE: Number of bytes read=0
FILE: Number of bytes written=179565
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=26
HDFS: Number of bytes written=26
HDFS: Number of read operations=0
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counter:
Launched map tasks=2
Launched reduce tasks=1
Data-local map tasks=1
Map-Local map tasks=1
Map-Local map tasks=1
Total time spent by all maps in occupied slots (ms)=73879
Total time spent by all reduce in occupied slots (ms)=24789
Total time spent by all map tasks (ms)=73879
Total time spent by all reduce tasks (ms)=24789
Total map-milliseconds taken by all map tasks=73879
Total reduce-milliseconds taken by all reduce tasks=24789
Total megabyte-milliseconds taken by all map tasks=73874726
Total megabyte-milliseconds taken by all reduce tasks=25182816
Map-Reduce Framework
Map input records=2
Map output records=4
Map output bytes=42
Map output materialized bytes=42
Input split bytes=332
Combine input records=4
Combine output records=4
Reduce input groups=2
Reduce input bytes=42
Reduce input records=4
Reduce output records=3
Spilled Records=2
Shuffled Maps=2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=2869
CPU time spent (ms)=6246
Physical memory (bytes) snapshot=3988221332
Virtual memory (bytes) snapshot=1826668736
Total committed map usage (bytes)=25526888
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=26
File Input Format Counters
Bytes Written=26
root@adoop-master:~#

```

Limitations of Our Implementation


One significant limitation we faced earlier on in the construction of our cluster concerned the versions of Hypriot and Flannel we were using. We found that that nodes were occasionally dropping out and there were large start-up times for the cluster. We eventually found this issue to be rooted in the Hypriot v1.4 and Flannel v0.7. Thus, we were limited to using Hypriot v1.6 and Flannel v0.9.

Another limitations of the implementation was the issue we faced regarding the port assigned to each of our slave nodes being chosen from a pool of 30,000 possible port numbers. The port which would get assigned to the slave node would have to be opened on the node itself. Thus, each time we had to go through and open a new port on each of our nodes. Thus, our initial fix was to open all ports on each of the nodes. However, that would take up resources that our RPi3's required for M/R. Thus, instead we decided to go to the source of the issue which was the node manager port and the yarn child port we assigned to port 3000 and the range of ports 3000-3008 respectively.

Additionally, we had a memory limitation with Map Reduce. Yarn was assigning 8GB of memory to each node. However, each node had only 1GB of memory available. Therefore, the cluster was experiencing large amounts of memory swapping due to the its limited memory. Thus, we had to limit the node and resource manager's memory requirements to be 1GB per node and 512MB per Map Reduce process.

Lastly, when we initially set up our cluster, we had the IPs being assigned to our cluster dynamically. Thus, when moved our cluster to a network which did not have the same IP address space, originally 10.0.0.1, the cluster could not connect to the internet. Thus, we had to have a static IP resolution for our cluster. Thereafter, when we moved our cluster to a new network, we rectified it by having our cluster connected to our own router, and our own router connected to the network.

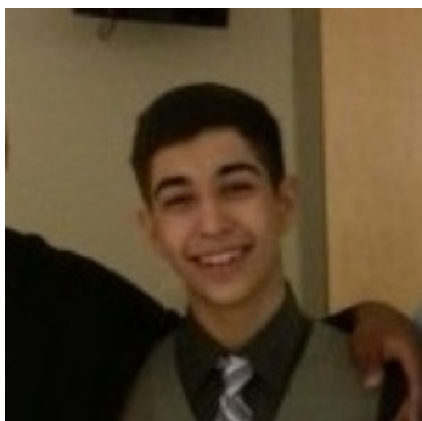
Team Members

Name	Image
Edmond Zalenski	

George Kharchenko



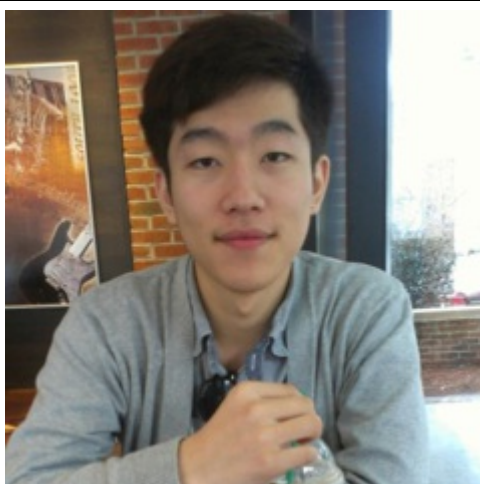
George Poulos



Jeff Kaleshi



Kevin Liu



Kunal Shah



Tim Choh

