

CS 553 Assignment 2

Ron Pyka & Jesi Merrick

Team Member Contributions

In this assignment, Ron provided his existing WordCount code from CS 451 last semester, and handled the Hadoop side of things, while Jesi worked with Swift.

Shared Memory Setup

For the single node for the shared memory program, we used a c3.large spot instance running the 64-bit Amazon Linux AMI (3.14 kernel.) Apart from installing gcc4.8 and dependencies for C++ support, there were no changes made to the initial image. The setup and running of the program for this test was very simple and did not take long. The times for running this version on 1-8 threads are included in the performance section below.

Hadoop Setup

Setting up Hadoop was fairly straightforward. I chose to go with the same 64-bit Amazon Linux AMI used in the shared memory setup (3.14 kernel) and used OpenJDK 1.7 for Java. I did not use ANT and chose to use Make instead, since that was more familiar. I also opted to use Hadoop 1.2.1 since I had worked with that last semester, and was at least a little familiar.

In Hadoop, the master acts as a dispatcher, assigning work to the various slave nodes, receiving their replies, and assigning more work until the task is completed. It also keeps track of mapping and reducing progress. Unique ports are set to various items to avoid conflicts in traffic and configuration files list out all the slaves for the master to know where to assign work, and tell the master and slaves what role each of them plays.

As for the configuration of Hadoop (for 16 nodes), on the master I changed the masters file from localhost to the master's public address and the slaves file from localhost to a list of all slaves public addresses. If I were to have less slaves, I would have put less into the slaves file. On each slave, I emptied the masters file, and changed the slaves file from localhost to the slave's own public address. On both the slaves and masters, I had to make the following changes:

conf/core-site.xml:

- fs.default.name - told all nodes where the namenode is for filesystem purposes, and the port to use (8020 assigned in this setup)
- hadoop.tmp.dir - told all nodes what the default directory would be for HDFS

conf/hdfs-site.xml:

- dfs.permissions - this was set to 'false' to indicate that any user could make changes to the HDFS. Would not be good for real-world use, but simplified this assignment.
- dfs.replication - I set this to 2 (one less than default) since I did not see a need for excessive replication in such a small and quick program. Could have been increased, though.

conf/mapred-site.xml:

- mapred.job.tracker - kept track of the jobtracker for mapreduce. In this case, it pointed to the single master.

Below is a screenshot of all 16 workers and the master up and running:

The screenshot displays the AWS Management Console interface for the EC2 service. The left sidebar shows navigation options like EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, Spot Requests, Reserved Instances, IMAGES, AMIs, Bundle Tasks, ELASTIC BLOCK STORE, Volumes, Snapshots, NETWORK & SECURITY, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces, AUTO SCALING, Launch Configurations, and Auto Scaling Groups. The main content area shows a list of 17 EC2 instances. The first instance, HadoopNameNode, is the master node, and the subsequent 16 instances, HadoopSlave1 through HadoopSlave16, are worker nodes. All instances are in the 'running' state, and their status checks are all passing (2/2 checks). The instances are located in the us-west-2 region, specifically in the us-west-2a availability zone. The HadoopNameNode instance has a public IP of 54.191.231.12, and the HadoopSlave instances have public IPs ranging from 54.191.231.38 to 54.191.231.19. The HadoopSlave16 instance is in a 'stopped' state. The bottom of the screenshot shows the details for the HadoopNameNode instance, including its description, status checks, monitoring, and tags.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time
HadoopNameNode	i-8406c88e	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-12.us-...	54.191.231.12	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave1	i-3303cd39	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-38.us-...	54.191.231.38	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave2	i-6602cc6c	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-24.us-...	54.191.231.24	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave3	i-3103cd3b	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-10.us-...	54.191.231.10	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave4	i-8b04ca61	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-76.us-...	54.191.231.76	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave5	i-6502cc6f	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-223-48.us-...	54.191.223.48	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave6	i-6904ca63	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-80.us-...	54.191.231.80	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave7	i-6802cc62	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-169-108.us-...	54.191.169.108	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave8	i-3203cd38	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-60.us-...	54.191.231.60	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave9	i-6404cafe	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-71.us-...	54.191.231.71	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave10	i-8506c88f	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-81.us-...	54.191.231.81	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave11	i-6902cc63	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-228-105.us-...	54.191.228.105	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave12	i-6a04ca60	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-31.us-...	54.191.231.31	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave13	i-6604ca6c	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-90.us-...	54.191.231.90	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave14	i-6504ca6f	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-62.us-...	54.191.231.62	hadoopWKey...	disabled	October 24, 2014 3:18:17 P...
HadoopSlave15	i-6704ca6d	c3.large	us-west-2a	running	2/2 checks...	None	ec2-54-191-231-19.us-...	54.191.231.19	hadoopWKey...	disabled	October 24, 2014 3:18:16 P...
HadoopSlave16	i-1f15ba85	c3.large	us-west-2a	stopped	2/2 checks...	None	ec2-54-68-194-94.us-w...	54.69.194.94	hadoopWKey...	disabled	October 24, 2014 4:16:53 P...
Steve	i-4502443f	t1.micro	us-west-2a	stopped	2/2 checks...	None	ec2-54-68-194-94.us-w...	54.69.194.94	hadoopWKey...	disabled	April 26, 2016 6:14:25 PM U...

Instance: **i-8406c88e (HadoopNameNode)** Public DNS: **ec2-54-191-231-12.us-west-2.compute.amazonaws.com**

Description	Status Checks	Monitoring	Tags
Instance ID	i-8406c88e		
Instance state	running		
Instance type	c3.large		
Private DNS	ip-172-31-45-67.us-west-2.compute.internal		
Private IPs	172.31.45.67		
Secondary private IPs			
Public DNS	ec2-54-191-231-12.us-west-2.compute.amazonaws.com		
Public IP	54.191.231.12		
Elastic IP	-		
Availability zone	us-west-2a		
Security groups	HadoopWC view rules		
Scheduled events	No scheduled events		

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

Also, here are screenshots of both my HDFS summary and list of active slaves:



NameNode 'ec2-54-191-231-12.us-west-2.compute.amazonaws.com:8020'

Started: Sat Oct 25 01:21:55 UTC 2014
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

33 files and directories, 189 blocks = 222 total. Heap Size is 214 MB / 889 MB (24%)

Configured Capacity	: 320.69 GB
DFS Used	: 20.06 GB
Non DFS Used	: 32.9 GB
DFS Remaining	: 267.73 GB
DFS Used%	: 6.26 %
DFS Remaining%	: 83.48 %
Live Nodes	: 17
Dead Nodes	: 0
Decommissioning Nodes	: 0
Number of Under-Replicated Blocks	: 0

NameNode Storage:

Storage Directory	Type	State
/home/ec2-user/hdfs/tmp/dfs/name	IMAGE_AND_EDITS	Active

This is Apache Hadoop release 1.2.1



ec2-54-191-231-12 Hadoop Machine List

Active Task Trackers

Task Trackers															
Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day	Total Tasks Last Hour	Succeeded Tasks Last Hour	Seconds since heartbeat
tracker_ec2-54-191-231-81.us-west-2.compute.amazonaws.com:localhost127.0.0.1:43191	ec2-54-191-231-81.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	63	62	63	62	0	0	572
tracker_ec2-54-191-231-24.us-west-2.compute.amazonaws.com:localhost127.0.0.1:42508	ec2-54-191-231-24.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	71	67	71	67	0	0	573
tracker_ec2-54-191-231-76.us-west-2.compute.amazonaws.com:localhost127.0.0.1:58817	ec2-54-191-231-76.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	70	65	70	65	0	0	572
tracker_ec2-54-191-231-80.us-west-2.compute.amazonaws.com:localhost127.0.0.1:53690	ec2-54-191-231-80.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	81	72	81	72	0	0	572
tracker_ec2-54-191-231-71.us-west-2.compute.amazonaws.com:localhost127.0.0.1:43750	ec2-54-191-231-71.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	71	64	71	64	0	0	572
tracker_ec2-54-69-194-94.us-west-2.compute.amazonaws.com:localhost127.0.0.1:45910	ec2-54-69-194-94.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	70	66	70	66	0	0	572
tracker_ec2-54-191-231-10.us-west-2.compute.amazonaws.com:localhost127.0.0.1:52419	ec2-54-191-231-10.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	65	62	65	62	0	0	572
tracker_ec2-54-191-231-52.us-west-2.compute.amazonaws.com:localhost127.0.0.1:51742	ec2-54-191-231-52.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	69	63	69	63	0	0	572
tracker_ec2-54-191-231-31.us-west-2.compute.amazonaws.com:localhost127.0.0.1:59252	ec2-54-191-231-31.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	68	68	68	68	0	0	572
tracker_ec2-54-191-231-19.us-west-2.compute.amazonaws.com:localhost127.0.0.1:49813	ec2-54-191-231-19.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	63	55	63	55	0	0	572
tracker_ec2-54-191-169-108.us-west-2.compute.amazonaws.com:localhost127.0.0.1:45814	ec2-54-191-169-108.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	71	63	71	63	0	0	572
tracker_ec2-54-191-231-50.us-west-2.compute.amazonaws.com:localhost127.0.0.1:48469	ec2-54-191-231-50.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	67	65	67	65	0	0	572
tracker_ec2-54-191-231-38.us-west-2.compute.amazonaws.com:localhost127.0.0.1:51251	ec2-54-191-231-38.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	0	0	0	0	0	0	64
tracker_ec2-54-186-223-48.us-west-2.compute.amazonaws.com:localhost127.0.0.1:51629	ec2-54-186-223-48.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	66	55	66	55	0	0	572
tracker_ec2-54-191-228-105.us-west-2.compute.amazonaws.com:localhost127.0.0.1:58402	ec2-54-191-228-105.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	73	72	73	72	0	0	572
tracker_ec2-54-191-231-90.us-west-2.compute.amazonaws.com:localhost127.0.0.1:57231	ec2-54-191-231-90.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	68	65	68	65	0	0	572

This is Apache Hadoop release 1.2.1

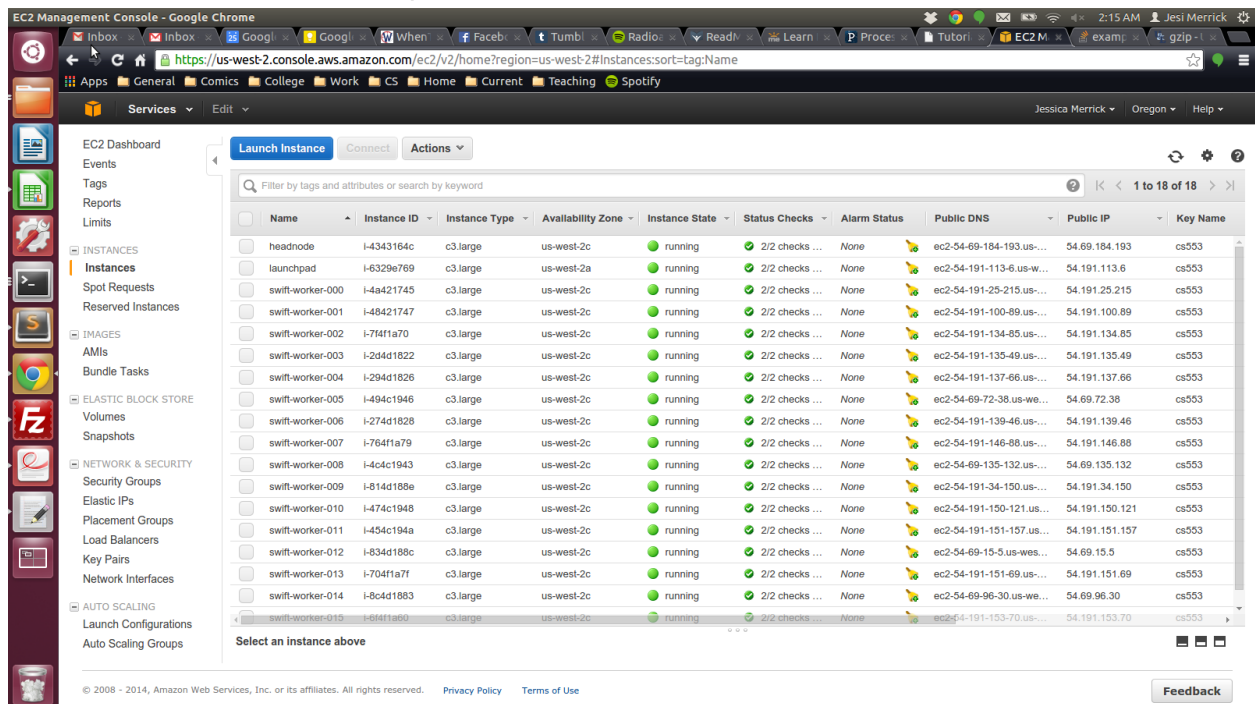
Swift Setup

The Swift setup was done by following [the tutorial](#) provided in the assignment description. The AMI used for all the nodes was Ubuntu Server 14.04 LTS. A launchpad instance is used, as in the tutorial, to determine the configurations of the head node and the workers node, and launch them by command. Additionally, an extra volume of 22GB was mounted to the head node (after it was created) in order to handle the larger data set. This step is not necessary for the workers, as the file is split before it is sent to them.

There were some difficulties in the swift setup. The first major block was with starting up the head node and the workers node as the tutorial indicated, but initially they could not connect. It was necessary to create a group in the IAM controls for the IAM user that is used in the tutorial. Additionally, the default configuration for the instances including only 8GB. This was not acceptable, as the large dataset is 10GB. As mentioned, this was resolved by mounting a new volume of 22GB to the head node, which was the only instance that had to handle the collective amount of the dataset.

There were a few smaller issues, such as the swift program never finishing. The only clear solution to this seemed to be to restart all the instances.

A screenshot of the the launchpad instance, the head node instance, and 16 worker instances:



The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, Spot Requests, Reserved Instances, IMAGES, AMIs, Bundle Tasks, ELASTIC BLOCK STORE, Volumes, Snapshots, NETWORK & SECURITY, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces, AUTO SCALING, Launch Configurations, and Auto Scaling Groups. The main content area displays a table of EC2 instances. The table has columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS, Public IP, and Key Name. The instances listed are: headnode, launchpad, swift-worker-000 through swift-worker-015. All instances are in the 'running' state. The table is filtered by tags and attributes, and the results are sorted by tag name. The footer of the console shows the copyright notice: © 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. and links to Privacy Policy and Terms of Use.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name
headnode	i-4343164c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-184-193.us...	54.69.184.193	cs553
launchpad	i-6329e769	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-113-6.us-w...	54.191.113.6	cs553
swift-worker-000	i-4a421745	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-25-215.us...	54.191.25.215	cs553
swift-worker-001	i-48421747	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-100-89.us...	54.191.100.89	cs553
swift-worker-002	i-7f4f1a70	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-134-85.us...	54.191.134.85	cs553
swift-worker-003	i-2d4d1822	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-135-49.us...	54.191.135.49	cs553
swift-worker-004	i-294d1826	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-137-66.us...	54.191.137.66	cs553
swift-worker-005	i-494c1946	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-72-38.us-we...	54.69.72.38	cs553
swift-worker-006	i-274d1828	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-139-46.us...	54.191.139.46	cs553
swift-worker-007	i-764f1a79	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-146-88.us...	54.191.146.88	cs553
swift-worker-008	i-4c4c1943	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-135-132.us...	54.69.135.132	cs553
swift-worker-009	i-814d180e	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-34-150.us...	54.191.34.150	cs553
swift-worker-010	i-474c1948	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-150-121.us...	54.191.150.121	cs553
swift-worker-011	i-454c194a	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-151-157.us...	54.191.151.157	cs553
swift-worker-012	i-834d188c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-15-5.us-we...	54.69.15.5	cs553
swift-worker-013	i-704f1a7f	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-151-69.us...	54.191.151.69	cs553
swift-worker-014	i-8c4d1883	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-96-30.us-we...	54.69.96.30	cs553
swift-worker-015	i-6f4f1a60	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-153-70.us...	54.191.153.70	cs553

Performance

Below are comparisons of performance for various numbers of threads for the shared memory program, as well as a comparison between all three on one node and hadoop and swift on 16 nodes.

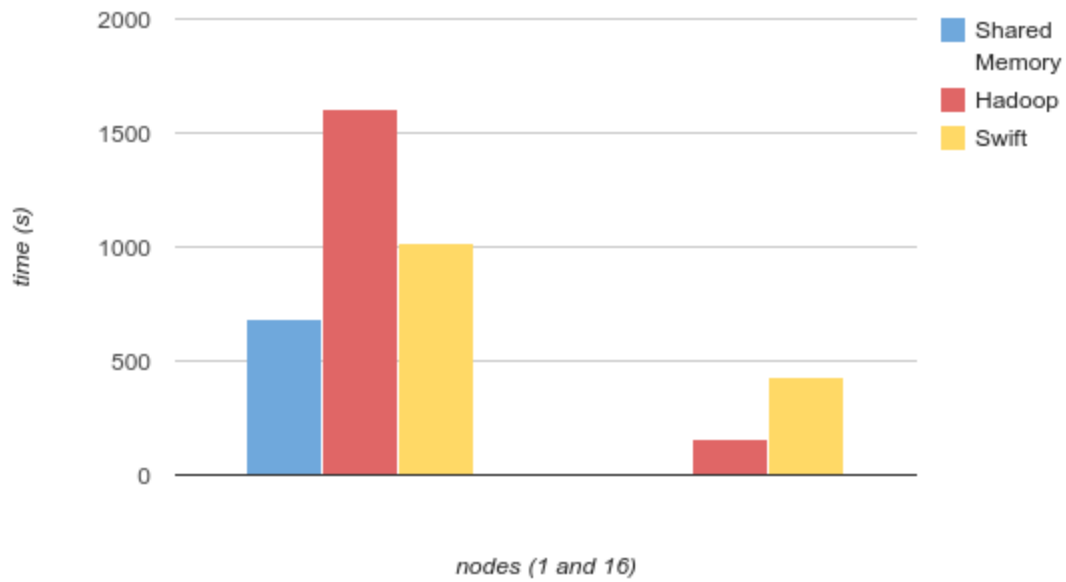
Shared Memory Performance

Threads	Time (in seconds)	Speedup
1	821.1 s	1
2	680.7 s	1.206
4	685.7 s	1.197
8	691.7 s	1.187

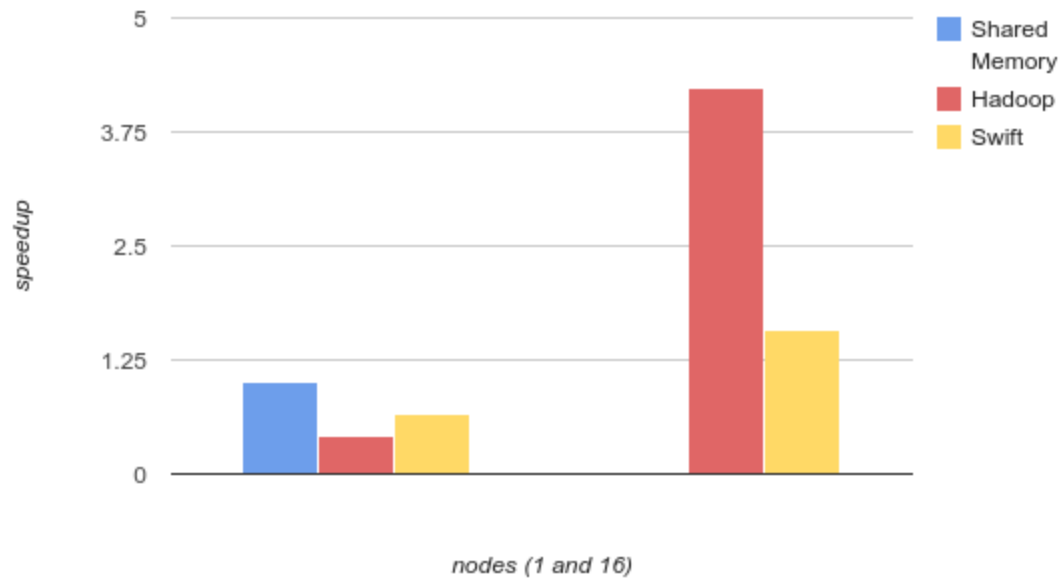
Comparison of all approaches

	Time (in seconds)	Speedup
Shared Memory	680.7 s	1
Hadoop (1 node)	1604.2 s	0.424
Swift (1 node)	1020.4 s	0.667
Hadoop (16 nodes)	160.8 s	4.233
Swift (16 nodes)	429.35 s	1.585

Execution Time



Speedup



When only one node is used, shared memory count has the best performance over Hadoop and Swift. This makes sense, since the benefit of Hadoop and Swift comes from how they distribute the work over multiple nodes. At one node, Hadoop and Swift show their overhead, which is normally acceptable due to improved performance with multiple nodes. Here, shared memory count does not have the same overhead, and so performs better.

If we compare Hadoop and Swift at one nodes to sixteen nodes, we see a significant increase in performance. Both Hadoop and Swift split up the work of word count in their own way, which obviously would improve over running word count sequentially (one node). Hadoop is better suited for the word count task however, and shows this in its performance over Swift.

With a continued increase in the number of nodes, we can expect Hadoop to continue to perform the best.