CSCI 5673 Distributed Systems
Summer 2016
Project Report
Brian McKean

Title: Examining performance scaling running Hadoop in AWS

Abstract:
There are many published articles examining performance of distributed applications like Spark and Hadoop in both non-virtualized and virtualized cloud environments. These articles primarily focus on overall benchmark performance and sometimes delve into cause of performance limitations for the overall benchmark. We look at AWS execution for Hadoop running a set of standard HiBench[i] micro-benchmarks. We tracked various system performance parameters at Hadoop cluster sizes with two, five and ten data nodes to understand how each parameter changes as the same workload is executed across several different cluster sizes. The parameters we examined are network bandwidth, disk io, CPU utilization and memory utilization. We will examine how the performance parameters change at different cluster sizes

## Introduction:

A Hadoop cluster can be used to perform map reduce function and can scale across multiple server nodes. For jobs that can be performed with map reduce the scaling provided by Hadoop can allow parallelization across a very large number of server. There have been much research published on Hadoop performance[1,2,3,4]. Some of this research also looks into detailed performance of the various aspects of the machines.[5] Some of the research looks at performance under failure conditions[6]. While we see performance scaling as node are increased we don't see how each node's cpu, disk, memory usage changes as the cluster size is increased. One reason that this information may be valuable is that the capacities used at various sizes for cloud based resources like AWS have different price points. The optimal price performance configuration for completing a job may not be same as the optimal performance configuration. By examining the individual node statistics, we may observe for instance that less memory is used on a larger cluster and therefore we can use a larger number of cheaper instances to perform the same job in perhaps better time.

## Hadoop Architecture:

A basic Hadoop systems consist of a name node and multiple data nodes as shown in Figure 1 Hadoop . Data node failure can be handled by the name node. Name nodes may use a

[i] https://github.com/intel-hadoop/HiBench/

master/slave arrangement to account for failure. For the purposes of this project we will use one name node and some number of data nodes for each run.

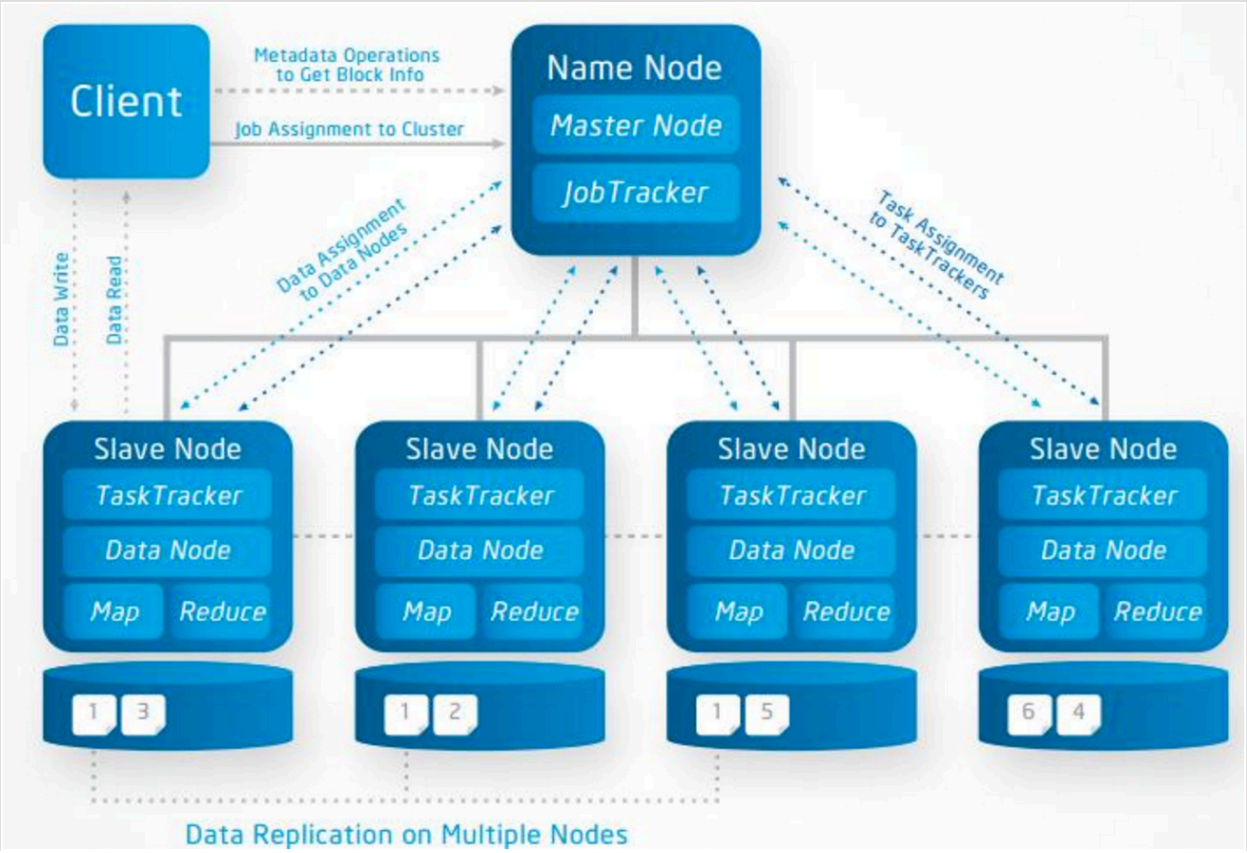


*Figure 1 Hadoop Architecture*

## Project Specific Configurations:

We used amazon AWS to set up and run the Hadoop cluster. We used one name node and performed runs with 2, 5 and 10 data nodes. We used an open source spark based set up script[ii] to set up the EC2 instances. The script is designed to set up spark. It sets up a specified number of data nodes and one name node with Hadoop and Spark running on top of Hadoop. The details of the setup in EC2 are shown in Table 1: AWS and Hadoop/Spark Setup.

*Table 1: AWS and Hadoop/Spark Setup*

| Parameter | Value |
|---|---|
| **AWS Instance Type** | M1.large |
| **AWS EBS Volume** | Standard 8 GiB Magnetic |
| **Number of Data Nodes** | 2, 5, 10 |
| **Number of Name Nodes** | 1 |
| **Number of Clients** | 1 |
| **Zone** | us-east-1 |
| **Hadoop Data Copies** | 3 |

[ii] https://github.com/amplab/spark-ec2

## Benchmarks Used:

The Intel Hi-Bench benchmark suite was very fragile and difficult to set up. For instance, the benchmark requires a larger instance to run. During the course project we spent several days getting HiBench set up to run in Hadoop mode. We were never able to run a Spark mode tests despite several days in attempting to work through issues. Therefore, we elected to focus on the Hadoop benchmarks. Part of the issues may have been the lack of explanation as to how to configure HiBench from a client node perspective. Also the build instructions were not provided for the instances setup by the Spark ec2 scripts so we had to find a way to install the Maven build tool[iii] on the AWS Linux images. The HiBench benchmark is a set of micro-benchmarks. In order to limit the scope of this project we chose three micro-benchmarks from the set provided by HiBench. The three micro-benchmark we chose are: sort, wordcount and pagerank. HiBench also provides streaming benchmarks but we did not have time to investigate those.

The data scale we chose to use with HiBench was 'large'. This was the configuration that the system faulted to. We did not have time to try other configurations. The possible options are `tiny`, `small`, `large`, `huge`, `gigantic`, and `bigdata`.

## Performance Measurement Tools:

In addition to the overall elapsed time we measured with the HiBench tool using the Linux time command we used the sysstat[iv] tool installed on the name node and each of the data nodes to probe the details of system performance. The sysstat package has performance tools for a wide variety of parameters. We will use it to extract performance parameters or each of the test runs. The test environment will be as shown in Figure 3 Test Set Up. We developed scripts to run on the client computer, the name node and the data nodes. The client scripts execute in the following fashion:

1. Run HiBench test preparation to load data
2. Erase previous run data
3. Start sysstat on name node and data nodes
4. Run HiBench benchmark with Linux "time" command
5. Wait for completion of benchmark
6. Terminate sysstat on name node and data nodes
7. Gather logs of run to client machine

[iii] https://gist.github.com/sebsto/19b99f1fa1f32cae5d00

[iv] https://github.com/sysstat/sysstat

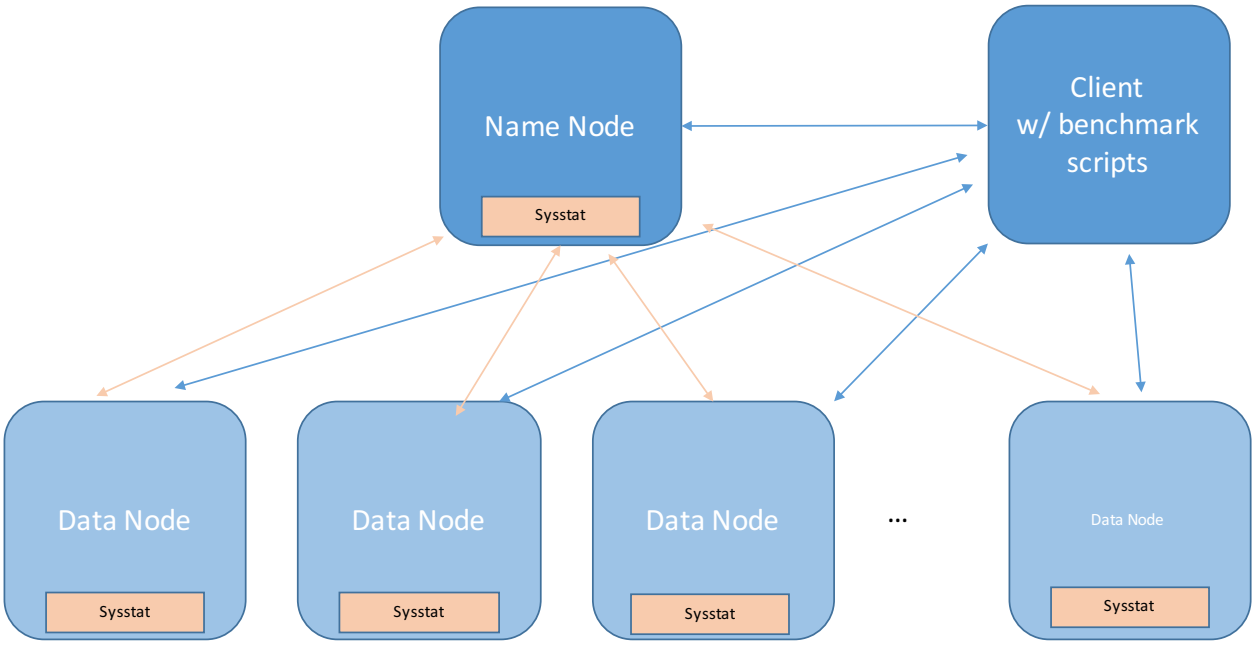


*Figure 2. Installation of the sysstat tool*

*Figure 3 Test Set Up*

Sysstat was set up to gather the set of information shown in Table 2: Sysstat Setup on Cluster Systems.

*Table 2: Sysstat Setup on Cluster Systems*

| Performance Parameter Probed | Systtat command | Sysstat reported results gathered | Key parameter extracted for use in this repost |
|---|---|---|---|
| **Disk IOPs** | sar –b 10 | tps rtps wtps bread/, bwrtn/s | tps – transactions per second |
| **CPU Utilization** | sar –u 10 | CPU %user %nice %system %iowait %steal %idle | %user – percent of CPU used by user |
| **Memory Utilization** | sar –m 10 | kbmemfree kbmemused %memused kbbuffers kbcached kbcommit %commit | %memused |
| **Network KB/s tx and rx** | sar –n 10 | IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/sr | rxkB/s, rxKB/s – data rates for send and receive – both on eth0 and localhost |

Amazon AWS provides Cloudwatch for a high level overview of various performance aspects of the system. We had intended to compare the results to Amazon Cloudwatch, however, AWS CLoudWatch requires an installation and setup process that we did not have time to execute during the time allotted for the project.  We save that for future work

## Test Methodology:

We ran the benchmarks for the three configurations and track how the overall system performance changes as we increase the number of nodes. We will show a portion of the data collected to examine how the system setup and benchmark chosen affect the individual system performance. We collected three runs for each benchmark on each size cluster.

## Overall Results:

Our overall results for the three test runs are shown in Figure 4: Time to complete test runs. There is high consistency in the times for completion of each of the three runs. With the sort benchmark there is not much change in runt time as cluster size increases. Page rank show a clear advantage for cluster sizes of 5 and 10 data nodes over a cluster with 2 data nodes. However, there is no noticeable improvement when moving from 5 data nodes to 10 data nodes. The wordcount benchmark shows a similar pattern. It is possible that a larger benchmark would have demonstrated an advantage for the 10 node cluster over the 5 node cluster but we did not have time to examine this possibility. We may draw conclusions from the fact that the overall results is that some aspect of the jobs allowed 5 nodes to perform better than two nodes and about the same as 10 nodes. Since disk, memory and CPU are proportionally larger as node size increases it is likely that one of these is the limiting factor with two nodes. Since 5 nodes and 10 nodes perform about the same we can speculate that either the job is not divisible enough or that network performance is the limiting factor in those cases.

Another observation about the results is that the benchmarks clearly show a difference in their time durations to run. Sort runs the fastest, followed by wordcount with pagerank taking significantly longer that the rest. Even the small two node system finishes the sort benchmark in less than little more than a minute. It would be interesting to perform additional experiments to see what the effects of larger size data sets would be on wort and wordcount benchmarks.

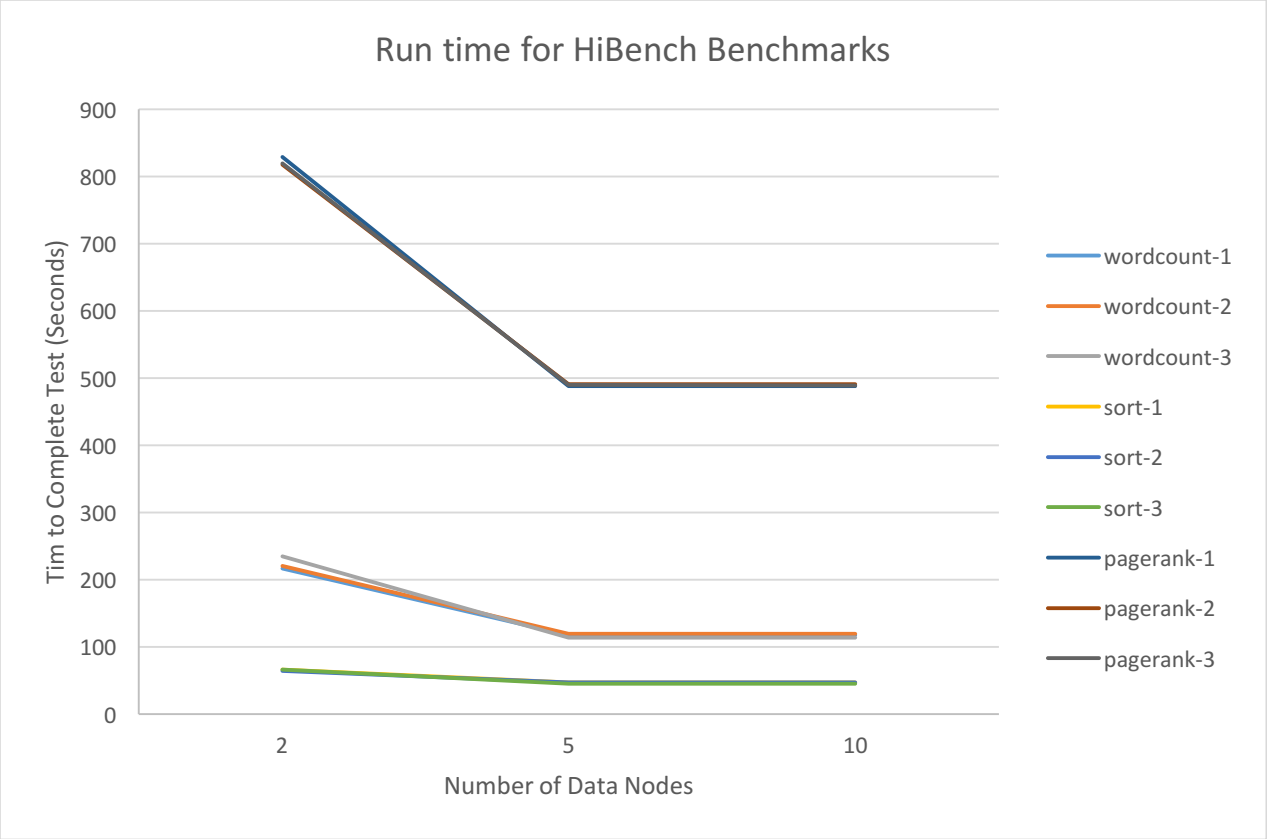


*Figure 4: Time to complete test runs*

## Results Details:

We ran the benchmarks for a variety of configurations previously specified and we tracked how the overall system performance changes as we increase the number of nodes. With the individual system performance monitored by sysstat we can examine what happens for each run. We averaged the system performance across all nodes in the systems and across all three runs to get an aggregate of the detailed performance.

Table 3 shows the results of the detailed system performance for the sort benchmark. We will later show the details of the run as the run progresses but this char shows an average across all the time duration of the benchmark. Figure 5 and Figure 6 show normalized performance across all runs, all nodes and all time slices of the test duration. Disk transactions per second scales down as nodes scale up so this is not a limiting factor in any of the configuration. It may be a limiting factor for the 2 Node configuration as magnetic disks have a maximum of 500 IOPs per volumes in EC2. The limit depends heavily on transfer lengths of the operations so the 127 tps for the 2 node cluster may be approaching a maximum CPU utilization is low. Memory utilization is surprisingly high for the 5 node and 10 node configuration and may be the limiting factor in this test for those cluster sizes. Network performance is puzzling since it is highest for the five node cluster. Perhaps the workload does not partition well for this size cluster.

*Table 3. Sort Average Across All Systems and All Runs*

| | 2 Node | 5 Node | 10 Node |
|---|---|---|---|
| **disk tps** | 127.55 | 94.96 | 48.73 |
| **cpu util %** | 13.98 | 9.40 | 5.26 |
| **mem util %** | 16.41 | 49.06 | 50.92 |
| **net KB/s tx** | 645.68 | 2384.69 | 1468.70 |
| **net KB/s rx** | 665.44 | 2449.95 | 1509.22 |
| **local KB/s tx** | 1105.17 | 1571.28 | 688.13 |
| **local KB/s rx** | 1105.17 | 1571.28 | 688.13 |

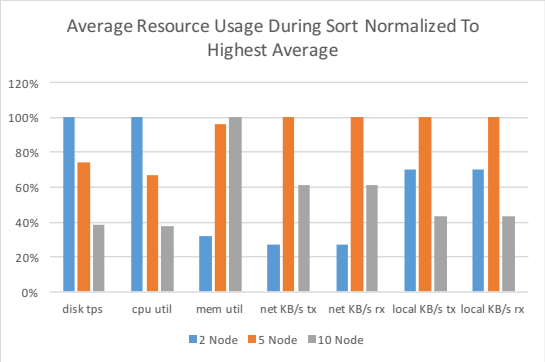


*Figure 5. Sort Average Resource Use Normalized to Highest Average*

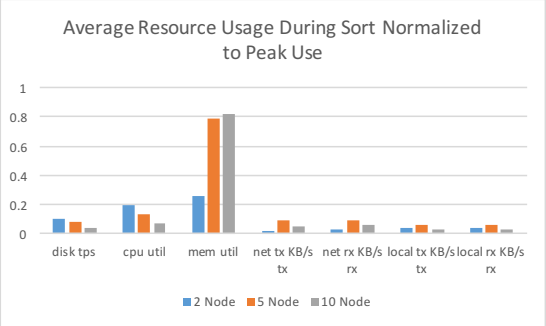


*Figure 6. Sort Average Resource Use Normalized to Peak Resource Use*

The results for the wordcount are shown in Table 4, Figure 7 and Figure 8. Wordcount shows very low disk transactions per second as compared to the other two benchmarks. This workload does not ask the disk to do much. In this test the 2 node memory utilization is much higher relative to the other configurations as compared to the sort workload. We speculate that this workload requires more memory to run and the sort uses more memory for coordination of work across the nodes. The limiting factor in the 2 and 5 node cluster appears to be the CPU utilization and for 5 and 10 node cluster memory utilization. There is significant network stack use directed at the local host in the test that we did not see in the host test. Perhaps this test requires significantly more coordination between the various components of Hadoop.

*Table 4. WordCount Average Across All Systems and All Runs*

| | 2 Node | 5 Node | 10 Node |
|---|---|---|---|
| **disk tps** | 5.03 | 5.14 | 3.96 |
| **cpu util %** | 42.91 | 41.95 | 28.81 |
| **mem util %** | 32.63 | 63.89 | 58.45 |
| **net KB/s tx** | 3.82 | 155.59 | 467.70 |
| **net KB/s rx** | 3.85 | 160.35 | 482.90 |
| **local KB/s tx** | 2833.91 | 2638.06 | 1524.57 |
| **local KB/s rx** | 2833.91 | 2638.06 | 1524.57 |

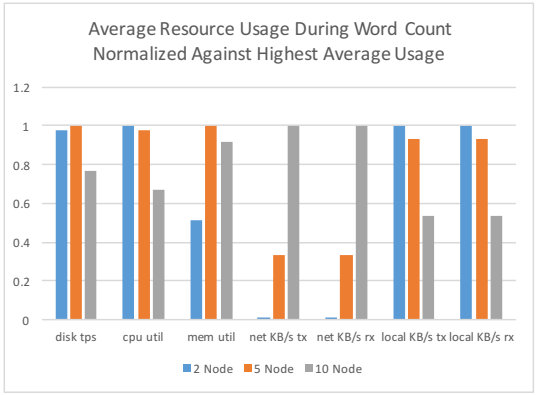


*Figure 7. Wordcount Average Resource Use Normalized to Highest Average*

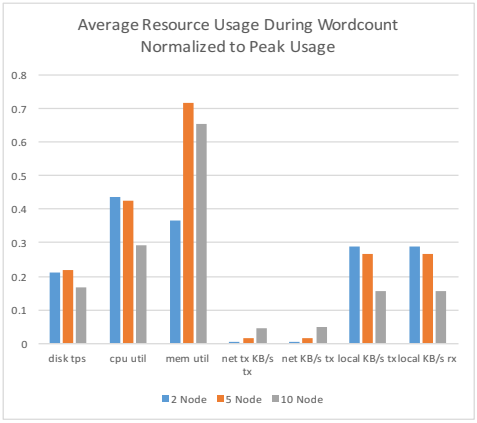


*Figure 8. Wordcount Average Resource Use Normalized to Peak Resource Use*

The pagerank benchmark is shown in Table 5, Figure 9 and Figure 10. Disk usage is again high in this benchmark as in the sort benchmark. CPU utilization is high for the 2 node and the 5 node and not as high for the 10 node. This suggests that the individual processing computations are more CPU intensive that the computations for exchange of information. Memory utilization is again high for the 5 and 10 node system. External networking is highest for this workload and may be a limiting factor in the execution time of the workload.

The disk performance is clearly higher that the wordcount routine and may be a limiting factor in the 2 and 5 node runs. Disk transactions are much lower for the 10 node run. For the pagerank benchmark it looks like the limiting factor is disk performance for small cluster sizes and memory space for larger configurations.

*Table 5. PageRank Average Across All Systems and All Runs*

| | 2 Node | 5 Node | 10 Node |
|---|---|---|---|
| **disk tps** | 120.61 | 100.37 | 63.62 |
| **cpu util %** | 43.50 | 31.58 | 19.91 |
| **mem util %** | 23.37 | 55.77 | 56.16 |
| **net KB/s tx** | 1524.96 | 1911.63 | 1320.22 |
| **net KB/s rx** | 1573.31 | 1967.63 | 1360.49 |
| **local KB/s tx** | 2513.56 | 1458.69 | 818.21 |
| **local KB/s rx** | 2513.56 | 1458.69 | 818.21 |

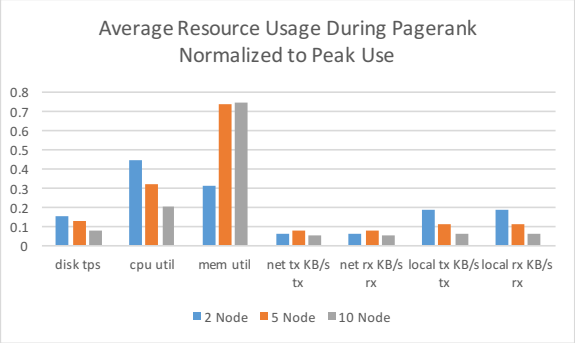


*Figure 10. Pagerank Average Resource Use Normalized to Peak Resource Use*

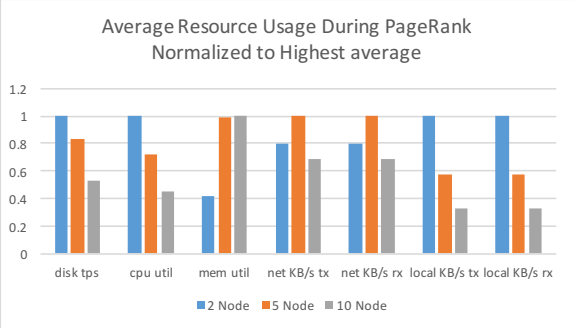


*Figure 9. Pagerank Average Resource Use Normalized to Highest Average*

## More details of pagerank runs:

We tracked each run at 10 second intervals to examine how performance changes over time. In this section we'll look at disk performance for all three tests and look at detailed performance for pagerank – which showed the most variation between cluster sizes.

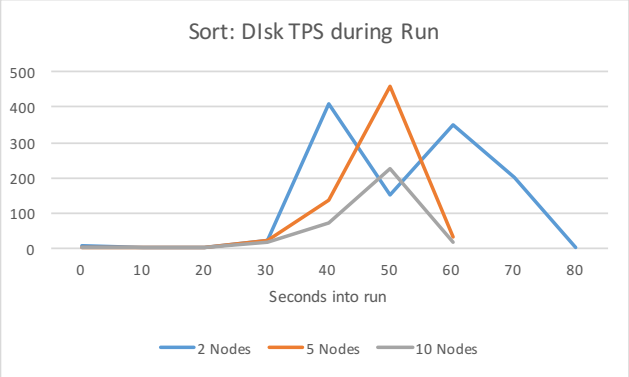


*Figure 11. Sort: Disk tps over run*

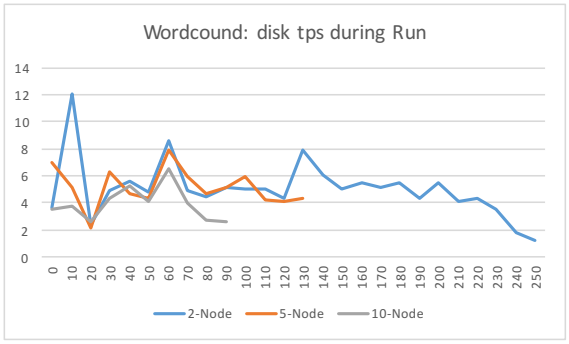


*Figure 12. Wordcount: Disk TPS during run*

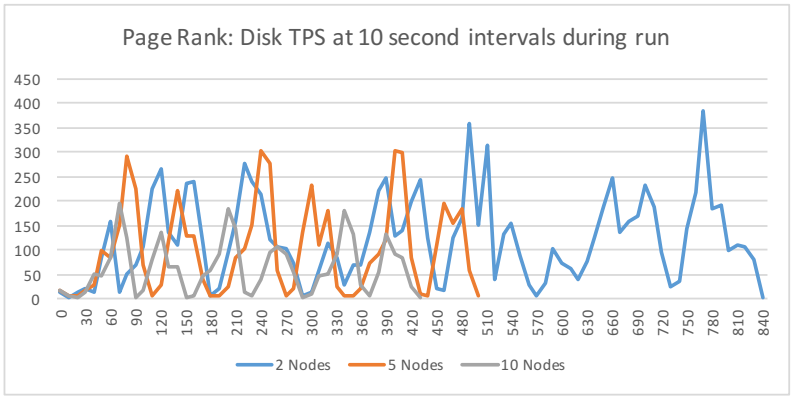


*Figure 13. Pagerank: disk tps during run*

For the comparison of disk transactions per second for the runs the details show that the pageerank benchmark is clearly bound by disk throughput with 2 nodes and 5 nodes. The maximum IOPs for an EBS HDD volume is 500 iops (input/output operations per second) (equivalent to 500 tps). Both the 2 node and the 5 node run show many peaks above 250 IOPs. The wordcount benchmark clearly has no disk limitation with peak disk IOPs never reaching 20 for any of the benchmarks. The sort benchmark also appears to have a disk limited performance for 2 and 5 nodes configurations. The collected performance data exceed peaks of 400 IOPs. In none of the cases does the 10 node cluster configuration appear to be limited by disk IOPs. Therefore, for these types of workloads an SSD configuration probably won't help reduce duration of the test. An SSD configuration will likely reduce the execution time for the 2 node and the 5 node configuration for the sort and the pagerank benchmarks.

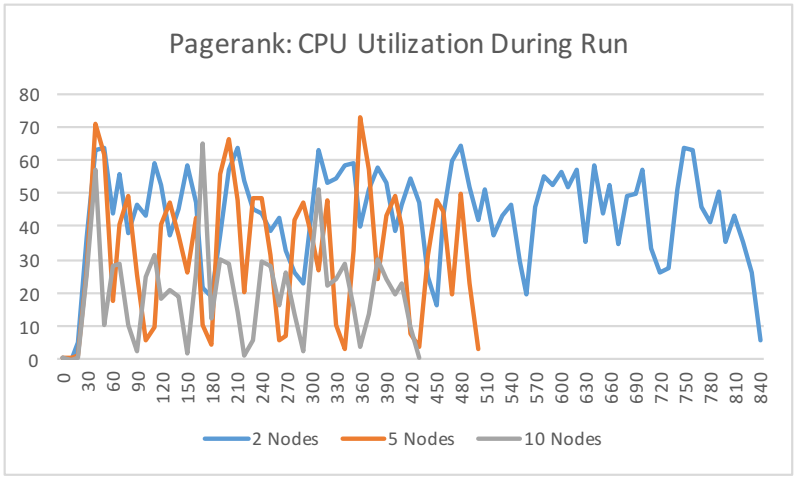


*Figure 14. Pagerank: CPU Utilization during the run*

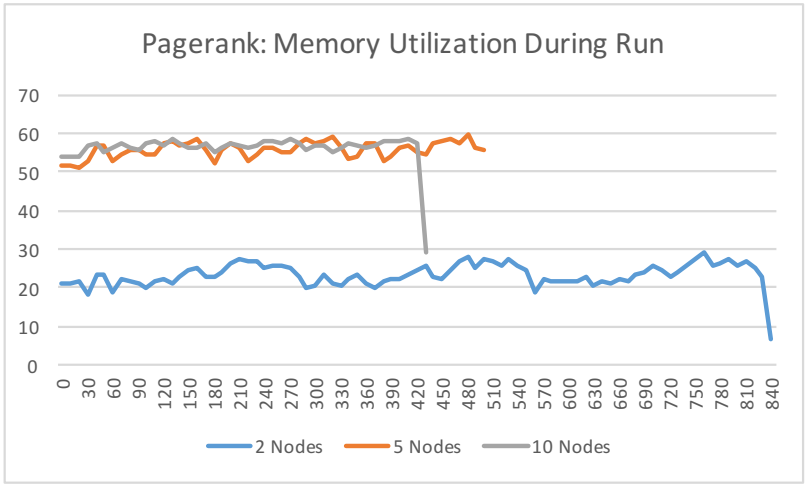


*Figure 15. Pagerank: Memory Utilization during the run*

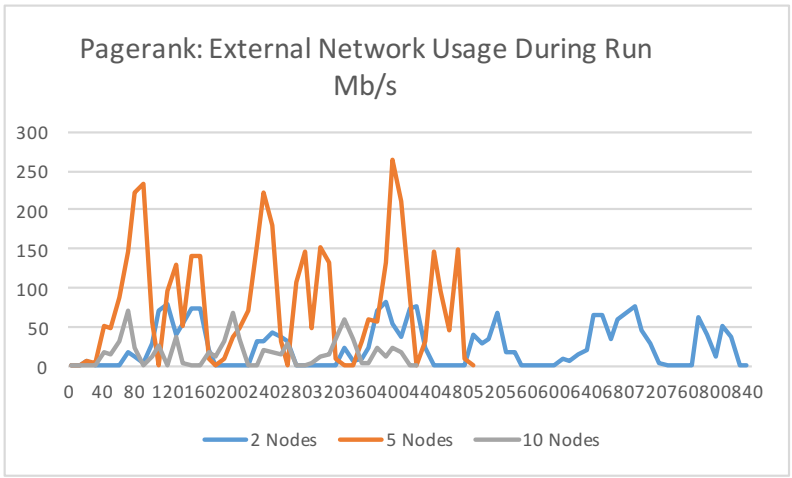


*Figure 16. Pagerank: External Network Usage during the run*

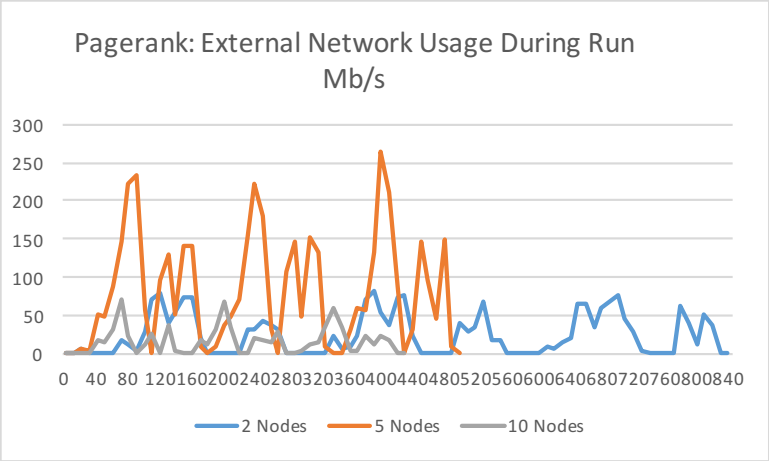


*Figure 17. Pagerank: Local Host Network stack usage during the run*

In addition to the disk tps notes above relative to the pagerank benchmark we see a few other interesting features in the detailed performance runs. First, in CPU utilization, Figure 14, we see that the CPU utilization is highest for the smallest cluster indicating that the analysis part of the workload is more intensive than the communications part of the workload. Second, in Memory utilization, Figure 15, show that after the disk tps and CPU load are alleviated by increasing the number of nodes from 2 to 5 and then 10 the memory load increases and likely becomes the next limiting factor in performance. Curiously for both internal and external networking the 5 node network showed significantly higher usage. Perhaps this can be explained as an artifact of other operations in progress on the system at AWS by other users. It would be interesting to see if similar results would be achieved by a standalone system

## Summary:

We have shown how we can use a set of industry standard micro-benchmarks to better understand how the Hadoop cluster system scales and for which types of workloads scaling helps. Through the use of detailed system performance analysis, we can find the causes of the limitations of performance and perhaps modify our algorithms or adjust or system parameter to increase performance. Examples of adjusting system parameters include: 1) Where disk tps is the limiting factor, we can select and SSD volume 2) If memory is a limiting factor we can select larger instance types in amazon. We can scale up some workloads to get better performance, like pagerank. Other workloads, like sort, need perhaps some more work on the algorithms used as scaling number of nodes does not improve performance.

There is a considerable amount of additional work that could yield additional insights into how to scale cluster workloads in AWS. The key next step is to compare these results with those from Amazon Cloudwatch. Since detailed Cloudwatch statistics is a premium service from amazon, an open source alternative using tools like sysstat could be developed. These would have the advantage of being able to be run on PaaS services other than AWS. Another next step could be to see if recommendations could be developed for modifications to either make jobs run faster or run in similar time frames with less expensive instance configurations.

## References:


[1] Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics, Shi Et Al., Proceedings of the VLDB endowment2015
s

[2] The Performance of MapReduce: An In-depth Study, Jiang et. al. VLDB 2010

[3] Improving MapReduce Performance in Heterogeneous Environments, Matei Zaharia Et al 8th USENIX Symposium on Operating Systems Design and Implementation

[4] Hadoop Performance Evaluation, Lab Report, Tien Duc Dinh

[5] Characterizing the Performance of Analytics Workloads on the Cray XC40, Ringenburg Et. Al. Cray Users Group CUG2016 Proceedings

[6] Analysis of Hadoop's Performance under Failures, Florin Dinu T. S. Eugene Ng, TR 11 Rice University