Implementation of Hadoop 2 on PBS-based HPC Systems

Joshua Hull, Mark Baker, Alex Berk
School of Computing
Clemson University
Clemson, SC 29632
Email:
{jhull, mnbaker, aberk}@clemson.edu

***Abstract*—Mark Baker, Alex Berk, and Joshua Hull implemented Hadoop 2 on a PBS-based HPC System on the Palmetto Cluster in a version update from Hadoop 1. Development saw challenges to creating a cluster environment on the Palmetto cluster that maintained compatibility with existing Hadoop 1 programs. Infrastructure and PBS script editing were necessary to support the additional features and YARN components in the Hadoop 2 framework. Thorough testing with WordCount and TeraSort was used to measure the success of implementation and initialization of the Hadoop 2 cluster.**

I. Introduction

As part of a group project for Clemson University's class on Distributed and Cluster Computing Mark Baker, Alex Berk, and Joshua Hull worked to implement the myHadoop2 framework and environment on the Palmetto Supercomputing Cluster at Clemson University. This environment will allow users to run the updated Hadoop 2 framework in an on demand cluster on Palmetto. Hadoop 2 introduced additional tools that need to be handled by myHadoop2, such as the YARN resource management system. These new tools help to reduce bottlenecks in Hadoop and generally increase performance. The purpose of myHadoop2 is to give the users of Palmetto and other PBS based systems a way to set up a dynamic Hadoop 2 cluster in order to take advantage of these new features while still allowing use of Hadoop 1 programs.

In addition to implementing the myHadoop2 script the group also benchmarked the script against its predecessor in similar cluster environments on Palmetto. The group implemented two standard Hadoop benchmarks: WordCount and TeraSort. These benchmarks were implemented in nearly identical manners, changing only Hadoop version-specific setup configurations.

II. myHadoop2 Implementation

A. Technical Implementation

The group worked for several weeks to modify the existing myHadoop scripts to work with Hadoop 2. The modifications involved adjusting the scripts to configure the new YARN resource manager system as well as the resource manager on the namenode of the cluster. We also made changes to the PBS scripts to allow Hadoop 1 scripted jobs to work in the Hadoop 2 framework. The script will need to be able to modify the new components configurations based on the resources that the PBS system has allocated to the user.

B. Group Members

The group members that were responsible for implementing myHadoop2 successfully were, as previously stated: Joshua Hull, Alex Berk, and Mark Baker. Each team member brought along various pieces of experience and was responsible for components that reflect their strengths in regards to the projects requirements for success.

Joshua was responsible for benchmark development and testing along with collaborating with the partners on various parts of the project. His approximate four years in using the Linux operating system and installing and configuring software for it meant that he was able to quickly install and configure a system with installations of Hadoop and Hadoop 2. Joshua was also responsible for developing the benchmark procedures that allowed the team to compare the performance of myHadoop2 to its predecessor using the testing situations in WordCount and TeraSort. The development of the benchmarks was aided by the number of years of Java experience Joshua has learned and worked with.

Alex was primarily responsible for the development of the myHadoop2 script with assistance from Joshua. Alex is currently working with Clemson Computing and Information Technology, or CCIT, to get Hadoop 2 working on the Palmetto cluster, and was able to bring this experience to the project and successfully implement that goal for both the success of the project and the goal of CCIT. He is directly involved in the implementation of Hadoop 2 over OrangeFS as well, which greatly benefited him in progressing the myHadoop scripting on the cluster efficiently and with the best performance. He spent much of the Spring 2014 academic semester researching implementations and configurations for old and new Hadoop versions[1][2]

Mark was responsible for documentation of the project and collaboration on development with Alex and Joshua with the implementation to complete the project's objective. Mark has worked as a functional analyst for the past three years and has experience in documentation of analyses and projects. He worked closely with Joshua and Alex throughout the development and implementation processes to bring a complete and

detailed understanding of the project to its documentation and presentation.

III. Challenges and Difficulties

A. Script Modification

During the script editing and testing of the project, Joshua came across issues in configuring a Hadoop environment using the myHadoop configuration scripts. Understanding the configuration scripts was a particular challenge. It took time for Joshua to educate himself in understanding the major components of the scripts and to find out the tools associated with Hadoop in order to configure them appropriately for our project. While working on the configuration elsewhere there were times when the configuration itself proved confusing due to its file references in Hadoop 2 compared with Hadoop 1.

With the new features prevalent in Hadoop 2, there was also the issue of implementing the new resource tracking system that wasnt initially set up when creating the system in our project, but with the work Alex had been working on for CCIT to set up their own configuration of Hadoop 2, we were able to borrow the OpenClemson source[3] for the resource system to assist us in getting ours working correctly.

B. Palmetto Environment

One of our earliest challenges was in setting up the new Hadoop environment on the Palmetto without disrupting the current Hadoop 1 framework and optimizing Hadoop 2. Fortunately, Alex had been working with CCIT in Clemson on getting this done with OrangeFS. Through research and contact, he found a partially configured open source implementation of myHadoop that was designed to work with Hadoop 2.2.0 developed by the San Diego Supercomputer Center. With a few bug fixes to let certain conditions on PBS Pro work correctly, we managed to incorporate the new myHadoop in our scripting and testing without having to greatly edit or modify an older version. The initial release of myHadoop 0.20 provided configuration files that edited the original Hadoop files for compatibility with PBS systems. This approach was difficult to read and understand, requiring the programmer to investigate each specific file for changes. The new myHadoop 0.30 has been completely reworked for simplicity and scalability. A patch file is used to alter the Hadoop configuration depending on Hadoop version, and a universal configuration file is used to generate the appropriate Hadoop xml configuration files. As a result, we now have a well-organized environment for Hadoop users.

Another challenge in getting myHadoop2 working on Palmetto was a setting present in the OpenClemson code that altered the list of nodes. This setting was designed to allow Hadoop 2 to use the InfiniBand connections available between compute nodes on Palmetto. However this setting was supplying incorrect node names. The setting was commented out, and Hadoop 2 is now using the 10 gb Ethernet connections between nodes. Seeing as this setting is only a regular expression, it can be modified in the future by someone with more knowledge on the subject and the network architecture of the Palmetto cluster.

IV. Configuration and Testing

As with all new systems, they must be tested in various configurations and with different test suites, and the implementation of Hadoop 2 for this project was no exception. Once the initial system was up and running on the Palmetto cluster, we ran a battery of tests using WordCount and TeraSort programs for easy measuring and performance testing due to their reliable and simple architecture. This allowed us to test various configurations of jobs on Hadoop 2. At the same time we also ran these same configurations and programs through the Hadoop 1 system so we could then compare results back with Hadoop 2 to see if the performance differences were correctly evaluated and if any issues arose during the actual testing and deployment of our new system.

The hardware configuration for each compute node was kept consistent during the testing. Each compute node used 17 processing cores and 10 GB of memory and was networked to the other nodes via a 10 GB Ethernet link. One node was designated the headnode and ran the NameNode, JobTracker, and YARN resource manager components of Hadoop. Each node, including the headnode, also ran the TaskTracker and DataNode components.

A. Hadoop 2 Testing

1) WordCount: The first benchmark conducted was WordCount. This benchmark counted the number of words in several classic works from the Project Gutenberg website[4]:

- A Connecticut Yankee in King Arthur's Court[5]
- Adventures of Huckleberry Finn[6]
- Around The World in 80 Days[7]
- From The Earth to the Moon[8]
- The Adventures of Tom Sawyer[9]

The benchmark used the unmodified WordCount example code from the Hadoop 1.1.2 downloaded from Apache and compiled into a Java jar file. The jar file was run via the hadoop command and timed via the time command. The recorded user times are plotted in figure one below.

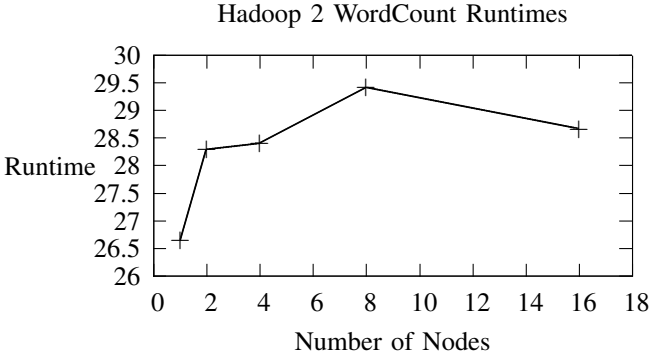

Fig. 1. Hadoop 2 WordCount Runtimes

The data above suggests that even while processing five relatively large files, the overhead of the MapReduce system inside Hadoop caused the runtimes to increase with the number of nodes. Eventually, the number of TaskTracker nodes reduced the runtime.

2) TeraSort: The second benchmark that was run was TeraSort. Due to time and resource constraints TeraGen was run with only 100000 as the input.

The benchmark used the unmodified TeraSort example code from Hadoop 1.1.2 downloaded from Apache and compiled into a Java jar file. This jar file was then run via the hadoop command and timed via the time command. The recorded user times are plotted in figure two below. The Hadoop environment is the same configuration as outlined above in WordCount.

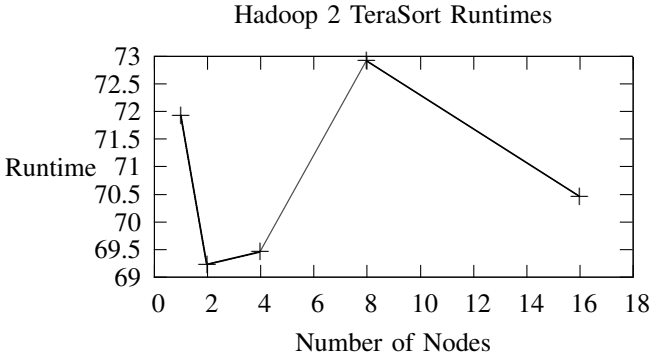

Fig. 2. Hadoop 2 TeraSort Runtimes

As previously seen with the WordCount tests, TeraSort showed increased runtimes with increased nodes due to overheads early but improved after several TaskTracker nodes were added. Furthermore, we see a significant decrease from a sequential, singular node running TeraSort in comparison with two nodes, but a sharp increase when moving up to eight nodes. This could be caused by many factors such as resource management not being optimal during testing or communications being congested since the Palmetto is always open for other users to run their programs. This could cause small, but consistent, performance drops in our runtimes.

B. Hadoop 1 Testing

The Hadoop benchmarking took place under similar conditions to the Hadoop 2 tests. There was one node specified at the headnode that ran the NameNode and JobTracker components to Hadoop. The primary difference is that the headnode did not run the TaskTracker or DataNode components. Each node used 17 processor cores and 10 GB of memory. The same Java jar file was used to benchmark Hadoop as was used in Hadoop 2. The runtimes were collected in a similar manner to Hadoop 2.

1) WordCount: WordCount on Hadoop was run using the same set of input files as listed above with Hadoop 2. The only changes were those needed due to Hadoop version differences. These include using hadoop commands that were deprecated with Hadoop 2.

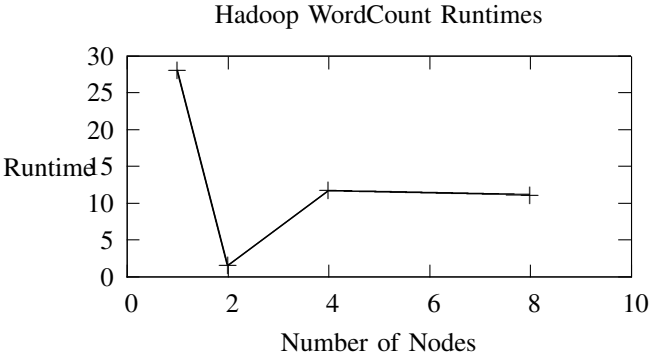

Fig. 3. Hadoop WordCount Runtimes

We can see in figure three that moving from sequential to multiple nodes improved runtime efficiency on Hadoop 1 as well and even had similar effects in overhead resource slowing down performance in the smaller number of node tests that improved later on. However, the Hadoop 1 tests could not replicate the testing circumstances of Hadoop 2 for larger amounts of datanodes due to Hadoop 1 and the Palmetto architecture working together. Because Hadoop 1 has a namenode setup that does not allow it to be a data processing node as well, in order to achieve a 16 node performance the system must allocate 17 nodes instead to account for the non-working namenode. In Palmettos current architecture, accessing more than 16 nodes requires allocation to a specific set of nodes on the cluster. These resources are much harder to acquire because of the uncommon request compared to many of the uses Palmetto sees. Therefore testing in Hadoop 1 to optimize time and efficiency could only test up to 8 working datanodes and so performance of a higher amount of nodes must be inferred based on available data.

2) TeraSort: The TeraSort benchmarks were performed in a similar matter the Hadoop 2 Benchmarks. TeraGen was given 100000 as an input parameter.

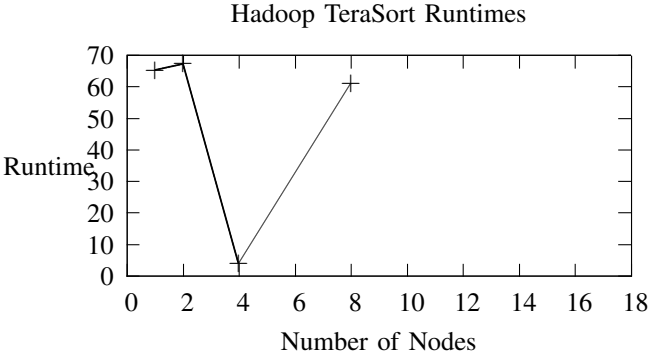

Fig. 4. Hadoop TeraSort Runtimes

From the TeraSort results we see a different performance from Hadoop 2 than its predecessor in runtimes during testing. While on average Hadoop 1 maintains a somewhat constant runtime speed for every node configuration there is an extreme outlier at four datanodes that has a much lower computation time compared to its other times in Hadoop 1 and compared to the tests run in the Hadoop 2 environment. Other circumstances may have affected the total runtime. Communication time on the cluster was not accounted for. It is also possible that the TeraSort program favored a four datanode setup and was not optimized to run using alternative node resources. Its performance graph does bare similar resemblance to performance speedups with Hadoop 2 but with a slightly different scale.

C. Overall Performance

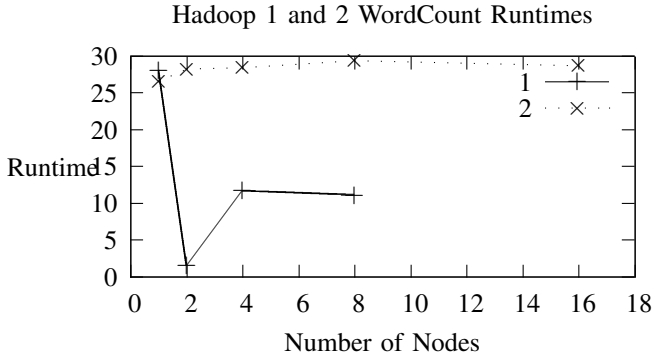

Fig. 5. Hadoop 1 and 2 WordCount Runtimes

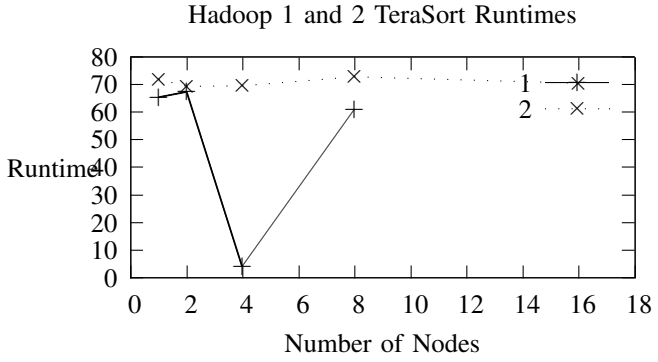

Fig. 6. Hadoop 1 and 2 TeraSort Runtimes

As the data suggests, there are differences in performance between Hadoop 1 and Hadoop 2 as well as some interesting points to note. The data showed that while both implementations improved program performance as more nodes were added, Hadoop 2s resource management and usable resource namenode allowed it to have a more significant improvement to performance based on available node resources as compared to Hadoop 1 in setup. At the same time, evidence suggests that while Hadoop 2 had greater improvement with respect to its own times it was Hadoop 1 that had the best overall performance for running WordCount and TeraSort programs. This conclusion could be somewhat erroneous due to many circumstances such as when the tests were run for each system on Palmetto, what kind of resources were available, and even perhaps if our current Hadoop 2 setup was not fully optimized to its potential. There is lots of room for improving our Hadoop 2 system to make better use of its resources and its implementation specifically in the Palmetto cluster but early results are promising. Hadoop 2 can and should outperform our Hadoop 1 setup and while our tests showed definite advantages in terms of multi-node runtime efficiency there is still opportunity in pushing the overall runtime down to surpass Hadoop 1s and improve overall performance of big-data computation on the cluster.

V. Conclusion

This project had many challenges and situations where it took us time to learn and try different methods in getting our system to run through trial-and-error. While the system may not be perfect for all possible computing needs, our group believes that the new Hadoop 2 system has proven itself to work and to have better performance and reliability than Hadoop 1. This project has laid the groundwork for further testing and expansion to replace the old system with ours for everyday use on the Palmetto cluster. The authors of this paper, Joshua, Alex, and Mark, have managed to create a new system for our users on the cluster for better big data computation. We suggest that our work should be continued to be improved upon to make it the best system possible. We ask that our work[10], all the source code and the environment, be open to others to use and to develop with, to create fixes and improvements to our implementation, and to move the Hadoop 2 system to a full replacement of Hadoop 1. Ideally, the cluster would have a dedicated system in place with scalability options for expansion and upgrade of Hadoop going forward. We invite our peers to make full use of our PBS-based HPC system with Hadoop 2, and we thank you for the opportunity to make it possible.

References

[1] T. White, *Hadoop: The Definitive Guide*. O'Reilly, 2012.
[2] A. J. Fonseca. Hadoop yarn installation: The definitive guide. [Online]. Available: http://www.alexjf.net/blog/distributed-systems/hadoop-yarn-installation-definitive-guide
[3] [Online]. Available: https://github.com/OpenClemson/myhadoop
[4] [Online]. Available: http://www.gutenberg.org
[5] M. Twain, *A Connecticut Yankee in King Arthur's Court*. Gutenberg Project, 2006.
[6] ——, *Adventures of Huckleberry Finn*. Gutenberg Project, 2006.
[7] J. Verne, *Around the World in 80 Days*. Gutenberg Project, 2008.
[8] ——, *From the Earth to the Moon*. Gutenberg Project, 1993.
[9] M. Twain, *The Adventures of Tom Sawyer*. Gutenberg Project, 2006.
[10] [Online]. Available: https://github.com/joshua-hull/myhadoop