Effective Deployment of an Apache Hadoop Cluster at OSC

Author: Abigail Hahn (HPC Client Services Intern)
Date: June 17, 2014

Introduction

Apache Hadoop is an open-source software framework used for the distributive processing and management of large amounts of data, which leverages data locality as a means of gaining a performance advantage. Hadoop also utilizes such failover mechanisms as data replication and rack awareness in order to ensure availability and security of data in the event of individual node failure.

The Hadoop software framework contains the following primary components[1]:

- Hadoop Common - A core set of utilities common to all other Hadoop modules.
- Hadoop Distributed File System (HDFS) - The file system used to distributively store the data to be processed, which appears contiguous from the client perspective.
- Hadoop MapReduce - An implementation of the map-reduce algorithm, used for parallel processing of the locally stored data.
- Hadoop YARN - A framework for job scheduling and cluster resource management.

Hadoop employs a master-slave architecture which consists of two types of nodes, the NameNode and DataNodes, and can be configured as a single- or multi-node cluster. The NameNode functions as the master node, managing cluster resources and file metadata within HDFS, and regulates file accesses from the client and the mapping of distributed file blocks across the DataNodes. The master node can also function as a DataNode. The DataNodes are the slave nodes, which manage the data to be manipulated. Each DataNode serves read and write requests from the client, and handles block creation, deletion, and replication as instructed by the NameNode.

Hadoop also supports block replication across multiple server racks in order to increase the availability of the data in the event of a hardware failure. Blocks are replicated across multiple racks in order to increase the total available network bandwidth while reading blocks of data. HDFS will attempt to read from file block replicas nearest to the reader in order to minimize read latency.

Problem Statement

A few users have approached the HPC Client Services staff with the desire to learn how to use Hadoop on OSC systems. Because Hadoop can be configured to trivially function on the current system architecture at OSC, meaning that the applications and utilities contained within the Hadoop project will be capable of successfully running to completion, simple “demo” use of Hadoop software on OSC systems should not pose any significant problems. However, due to the limitations of our current system configuration, users of Hadoop will likely not easily be able to achieve the performance benefit for their work that Hadoop was truly designed for. As these users begin to familiarize themselves with the Hadoop software and its capabilities, the need for a more optimal solution which allows users to operate on large persistent data while gaining a performance advantage could arise in the future.

Some of the benefits of using the Hadoop framework are:

- Hadoop is open-source.
- Hadoop comes bundled with it’s own file system, HDFS, so there’s no need to install another one on the Hadoop cluster.
- In-depth knowledge of parallel computing isn’t necessary -- Hadoop does all of the heavy lifting for you.
- The MapReduce model may provide unique opportunities for developers and computational scientists to solve computationally intensive problems that were previously, from a practical standpoint, intractable, or to improve upon existing methods of solving problems with well-known solutions.

There are also some notable tradeoffs:

- Hadoop is designed to operate on persistently stored data, which is in opposition to the goal of HPC centers providing scheduled access to compute resources.
- HDFS is not POSIX compliant.
- Because the functionality of Hadoop is highly specialized, it is fairly inflexible in terms of compatibility with other types of applications and network configurations. Modifications to the underlying framework in order to support interactions with pre-existing applications is possible, but often comes at the expense of performance.

In the following sections, the potential uses for Hadoop/MapReduce in the world of scientific computing will be described in order to clarify the potential needs of users of OSC systems. Following this will be a discussion about the limitations of the current hardware configuration at OSC, and a proposed revision to the current configuration that may be more suitable for maximizing the utility of the Hadoop software at OSC. Some currently existing alternative implementations of the map-reduce programming model, which are already tailored to traditional HPC environments, will also be mentioned.

Potential applications of map-reduce for users of OSC systems

Many problems suited for being solved by map-reduce are of the following types. (This list is not intended to be exhaustive)[3]:

- Counting/Summation
- Collating
- Parsing/Filtering
- Searching
- Sorting
- Distributed Task Execution
- Graph Processing
- Cross-Correlation
- Set Operations

Some of the currently evolving applications of the map-reduce model within specific scientific computing disciplines are listed on the page at this link[5].

Current Configuration

Currently all compute resources on OSC systems are configured to use one of the NFS, GPFS or Lustre file systems, with the only point of access for users being the PBS Batch System. Jobs submitted through the batch system are meant to be created, run and destroyed within the amount of time requested by the user for the job, and therefore no data will persist for the user before or after this time. In order for users to have the ability to create a Hadoop cluster within the batch system under the current configuration, an instance of HDFS must be created on the nodes within the job, and all data would need to be distributively copied to the local scratch space on each node before any computation begins. Users would have at their disposal ~1TB of local storage per node requested (on Oakley), which would need to be carefully managed according to how much data will need to be available for reading and how much data will be written during the data's processing. Results will also need to be offloaded from the job's nodes before the walltime of the job has run out.

There are some significant limitations to running Hadoop in the batch environment at OSC, the most obvious being the amount of time required to move the data back and forth between the cluster's local storage and the user's permanent storage. Other limitations include the amount of physical storage available per node, and resource allocation limits such as walltime and nodes, as well as the number of jobs a user currently has scheduled in the queue. Typical multi-node Hadoop cluster sizes range from just under 100 nodes to thousands of nodes, and these clusters operate on data sets ranging from hundreds to thousands of terabytes over a time span of a few weeks to a few months[12]. Depending on the nature of the work being done by a particular user or

research group, large amounts of compute resources would likely need to be reserved well in advance in order to avoid high queue wait times or queue rejection.

Optimal Configuration

The best way to ensure that users are able to gain a performance advantage while using Hadoop to process their data would be to set aside a subset of compute nodes for this purpose. There are a couple of ways that this might work. One option would be to simply create queue reservations for users within the batch system, using currently existing compute nodes and allowing walltimes for these jobs to be larger than 96 hours in order to create an environment facilitating the use of persistent data. This way, a user's data could be copied to the cluster once, and will not be destroyed until the batch job has ended, presumably when the reservation ends.

Another potentially longer term option would be to create a completely independent cluster of nodes outside of the batch system using Hadoop's built-in job scheduling capabilities to manage the jobs of multiple Hadoop users. This might be a more appropriate solution as the demand for Hadoop increases, since new nodes could be tailor made to better support the functionality of Hadoop. This is important in part because different hardware configurations are recommended to be used for master and slave nodes since the work done by the master nodes differs substantially from the work done by the slaves. The tables below, taken from the Hortonworks Cluster Planning Guide[11], shows the recommended hardware by node type, workload type and cluster size.

Table 1.1. For small clusters (5-50 nodes):

Machine Type	Workload Pattern/ Cluster Type	Storage	Processor (# of Cores)	Memory (GB)	Network
Slaves	Balanced workload	Four to six 2 TB disks	One Quad	24	1 GB Ethernet all-to-all
	HBase cluster	Six 2 TB disks	Dual Quad	48	
Masters	Balanced and/or HBase cluster	Four to six 2 TB disks	Dual Quad	24	

Table 1.2. For medium to large clusters (100s to 1000s nodes):

Machine Type	Workload Pattern/ Cluster Type	Storage	Processor (# of Cores)	Memory (GB)	Network
Slaves	Balanced workload	Four to six 1 TB disks	Dual Quad	24	Dual 1 GB links for all nodes in a 20 node rack and 2 x 10 GB interconnect links per rack going to a pair of central switches.
	Compute intensive workload	Four to six 1 TB or 2 TB disks	Dual Hexa Quad	24-48	
	I/O intensive workload	Twelve 1 TB disks	Dual Quad	24-48	
	HBase clusters	Twelve 1 TB disks	Dual Hexa Quad	48-96	
Masters	All workload patterns/HBase clusters	Four to six 2 TB disks	Dual Quad	Depends on number of file system objects to be created by NameNode.	

Alternative Map-Reduce Implementations

A number of alternative map-reduce implementations have been developed in order to better support work in traditional HPC environments.

Academic Projects

The following academic projects are mentioned below to provide additional insight about the potential for development of a map-reduce framework using traditional HPC architectures, but their implementations appear not to be publicly available at this time.

- *MARISSA (MApReduce Implementation for Streaming Science Applications)*[6]
- *MARIANE (MApReduce Implementation Adapted for HPC Environments)*[7]

Note: MARIANE is a successor to MARISSA.

Open-Source Projects

The following open-source projects are publicly available solutions that could offer users of OSC systems an opportunity to use the map-reduce framework in the interim while the cost-benefit analysis of Hadoop cluster deployment is under investigation.

- *Spark*[8]
- *MapReduce-MPI*[9]
- *Pheonix*[10]

Concluding Remarks

Apache Hadoop may offer improved methods of solving computationally intensive problems for users in particular research fields, and should be investigated further in order to measure the weight of these benefits against the demand for and cost of implementing a fully-functioning Hadoop cluster. In it's present state, the current hardware configuration at OSC isn't well-equipped to provide a performance benefit in terms of data locality, since all data needed for a particular Hadoop batch job would need to first be copied to the scratch space on the node before processing, though the performance of MapReduce operations will remain stable in this scenario. However, data sets exceeding ~1TB per node requested will lose the performance benefit gained from the MapReduce model since the data will need to continue to reside in persistent locations on already existing file systems outside of the compute cluster, which will create I/O bottlenecks during processing. In order for for users to experience the full benefit offered by the map-reduce model, an entirely new subsystem will likely need to be

built or configured to support it, if the implementation of choice is Hadoop; otherwise, alternative implementations should be investigated.

References

1. The Official Apache Hadoop Documentation
http://hadoop.apache.org/docs/current/

2. The National Energy Research Scientific Computing Center (NERSC)
https://www.nersc.gov/users/software/vis-analytics/hadoop/

3. The Highly Scalable Blog: MapReduce Patterns, Algorithms, and Use Cases
http://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/

4. Introducing MapReduce to High End Computing
http://www.pdsi-scidac.org/events/PDSW08/resources/slides/mackey-Introducing_MR_to_HEC.pdf

5. Mapreduce and Hadoop Algorithms in Scientific Computing
http://atbrox.com/2011/05/16/mapreduce-hadoop-algorithms-in-academic-papers-4th-update-may-2011/

6. MARISSA (MApReduce Implementation for Streaming Science Applications)
http://dx.doi.org/10.1016/j.future.2013.12.007

7. MARIANE (MApReduce Implementation Adapted for HPC Environments)
http://dx.doi.org/10.1016/j.future.2013.12.007

8. Apache Spark
http://spark.apache.org/

9. MapReduce-MPI Library
http://mapreduce.sandia.gov/

10. The Phoenix System for MapReduce Programming
http://mapreduce.stanford.edu/

11. Hortonworks Cluster Planning Guide
http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.2-Win/bk_cluster-planning-guide/content/ch_hardware-recommendations.html

12. What Do Real-Life Apache Hadoop Workloads Look Like?
https://blog.cloudera.com/blog/2012/09/what-do-real-life-hadoop-workloads-look-like/

13. Algorithms for Large-Scale Astronomical Problems
http://www.cs.cmu.edu/~binf/thesis_proposal/proposal.pdf

Additional information:

http://www.hpcwire.com/2014/02/11/hpc-hacking-hadoop/
http://www.slideshare.net/hortonworks/large-scale-math-with-hadoop-mapreduce
https://wiki.csiro.au/display/ASC/Hadoop
https://cs.uwaterloo.ca/~ashraf/pubs/hpcs09scientific.pdf
http://blog.cray.com/?p=6617