495 Project 3
Hadoop Map/Reduce

Kevin Brandstatter
Weiwei Wu

Environment:
Host machine:
- Amd phenom x6 at 3.3 GHz. 6 cores.
- 2 1TB drives in raid 1, Spinning Disk.
- 16 GB Ram DDR3 1600 MHz
- Gigabit ethernet.
- Ubuntu 12.04 Linux

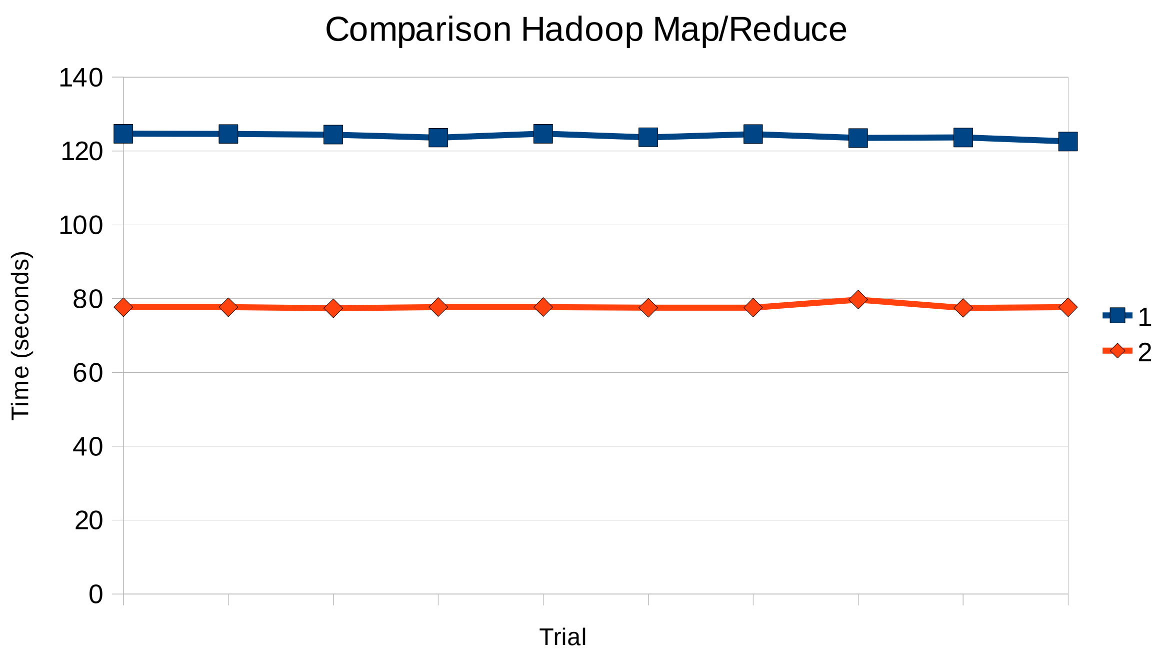
Virtual Cluster
- 8 Nodes using Oracle VirtualBox
- Each running CentOS 6.3 kernel 2.6.32
- 1 GB Ram, 1 Core
- Local network nanaged by PFSense virtual machine.
- Hadoop version 1.0.3
- Ant version 1.7.1
- Java openJDK version 1.7.0 (icedtea vm)

The virtual cluster is set up on an internal vm network that VirtualBox Manages. In order to facilitate easier intercommunication with each other and the internet, I set up a PFSense routeer as a Virtual machine to manage the LAN. It took a while to get the routing correctly configured between everything, mostly due to the way the router was connecting to the internet. With that set up, I created the master node for starting and stopping the cluster, and a single slave node that I then set up to connect to the cluster. The code base for the project and hadoop is stored in NFS mounted from the master node. I faced several problems with permissions when trying to export user directories to other machines, but they were solved with the no_root_squash option on the nfs server export.

The hadoop configuration was fairly straightforward. The master file contained only the master node, and the slaves file contained the master and slaves{0-6} depending on how many were necessary for the run. The master was also the root for the hdfs and the jobtracker.

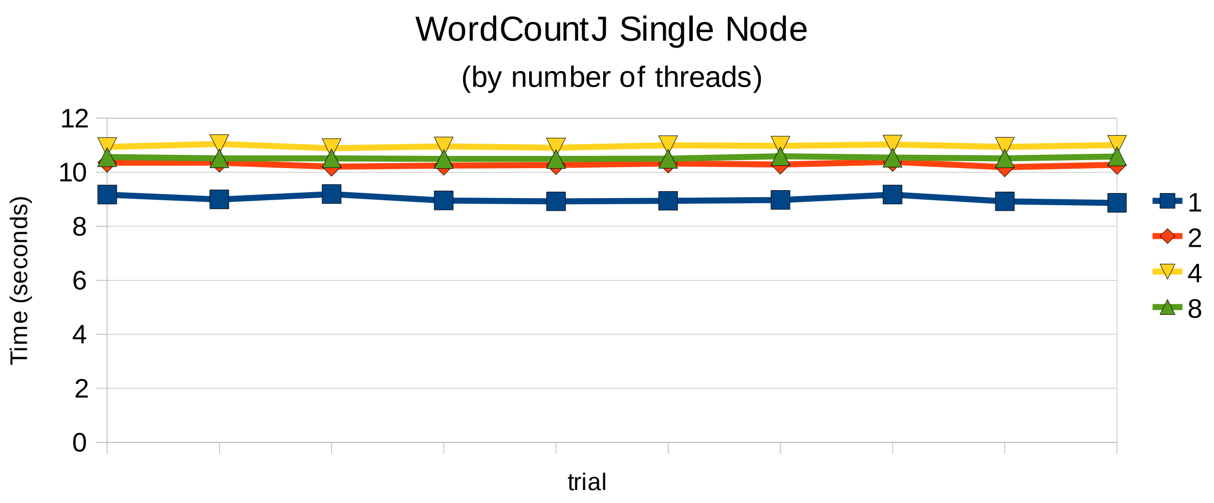
Coding:

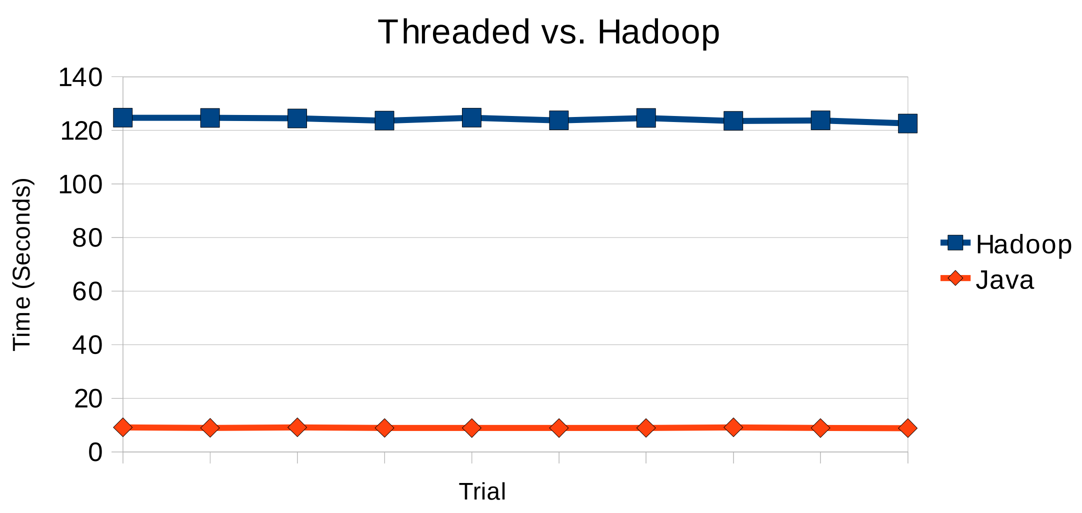
For the Hadoop implementation, we used the example source as a basis for starting and launching the job on the hadoop system and modified the map and reduce functions to handle case insensitivity and special cases of characters. The threaded version follows a similar model as map reduce by having a thread pool of mappers that each map an entire file. Then as a final step we merge and sort the resulting maps into one unified map which essentially is the reduce phase of the process. It's very difficult to model map reduce exactly using a multithreaded environment because of the differences in communication. Hadoop's map reduce is able to assign partitions and have the reducers collect the maps over the network. This is very hard to do in a threaded environment as the interprocess communication is not so straightforward which means they bottle neck on the main thread at some point which slows down the processing.



Comparison of hadoop on one and two nodes:

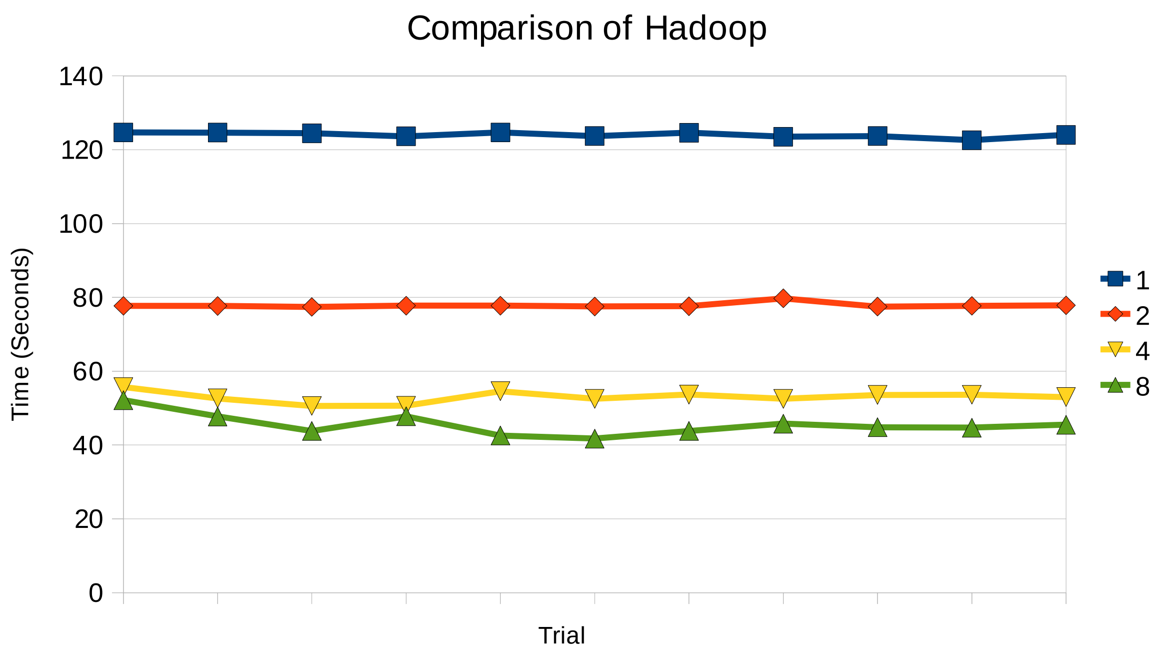
From the chart there is a significant improvement when the application is moved from one to two nodes. The speed up is almost 60%. While if perfectly parallelized the speed up would be 100% there is a large amount of overhead of communication introduced by the framework and network communication and transfer of data.





Comparison Of WordCounJ and WordCountMR on one node:

The first chart shows something very interesting about the threaded java application. The more threads we run at the same time, the slower it gets. This is due to the fact that it is running on a single core and thus more threads means more competition for resources. On the other hand the second chart shows that for this test, the threaded java word count is far faster than the hadoop implementation. This is because of the enormous amount of overhead introduced by the map reduce framework. While one might be tempted to use this to say that the hadoop framework is slow, it will scale far better than the threaded version because the threaded version can only be run on a single machine. On the other hand, the hadoop implementation shows improvement in performance by using more nodes. Also, this test is fairly small for a map/reduce workload, and a task of much larger size would cause the overhead of the framework to appear much less significant than in this case.



Hadoop on more than 2 nodes.

For our cluster, we were able to expand it to 8 nodes. From our tests we see that the speed up with each increase in cluster size becomes less and less noticable, with there being very little speedup from 4 to 8 nodes. This flattening out of the speedup is no doubt do to resource contention of the virtual machines as there are a total of 8 VMS running on 6 nodes. This means that they no longer have dedicated cores and severely limits the speedup potential of the application.

Contributions:
Kevin Brandstatter: Cluster setup and configuration, Hadoop setup and tests.

Weiwei Wu: Threaded Java wordcount, regex patterns for mapping of special cases.