Novel Optimization of Task Scheduling within MapReduce/Hadoop

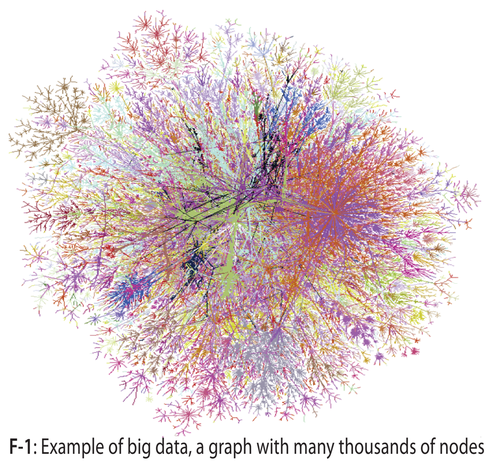
Daniel Vicory; Allan Hancock College, Computer Science
Mentor: Nan Li; Faculty advisor: Xifeng Yan
University of California, Santa Barbara, Department of Computer Science

Abstract

MapReduce, an algorithm created by Google, can be used to work with large datasets, such as indexing the web. The simple and distributed approach to problems by MapReduce has afforded its widespread use, with the leading implementation being Hadoop. However, non-uniform data and heterogeneous clusters mean that tasks usually finish executing out of sync, which, due to the nature of MapReduce, can leave computing power untapped. SkewReduce, a project from researchers at the University of Washington, unveiled a method to reduce the skew, or difference in task completion times, through the use of cost analysis functions and sample data. With these two pieces, their framework can calculate how long the algorithm will take on any given computer and partition the dataset optimally so that tasks finish together, reaching theoretical efficiency. However, because of the cost functions, it is not an out of the box solution to skew. Therefore, we propose a novel optimization of task scheduling that doesn't require these additions. Our algorithm stops tasks that have been executing for too long in comparison to other tasks and redistributes the work to the cluster. It replaces SkewReduce's optimizer and cost portions, so we can compare to previous research on the performance of SkewReduce and Hadoop's task scheduler. This project is still in progress, and we do not yet have our task scheduler complete to benchmark with. We are confident we should be able to make modest performance gains against Hadoop's task scheduler and approach the limit of what is possible without any changes to existing algorithms or knowing the cost.

Background

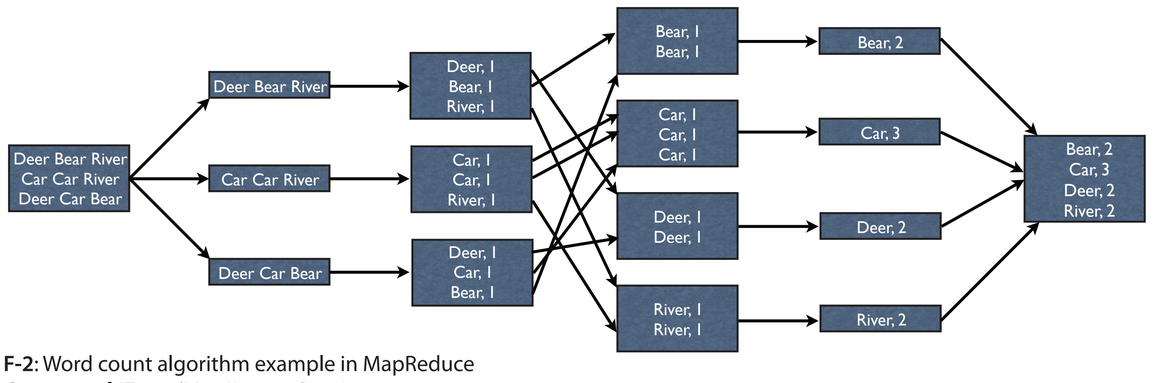
Researchers, businesses, and other organizations are collecting data at an unprecedented rate. The data collected can no longer be managed by traditional means, otherwise known as *big data* (see **F-1** for example). Data mining, a new interdisciplinary field of computer science involving mathematics, statistics, and other specialized fields relevant to the data at hand, helps make sense of the large amounts of data by extracting patterns and condensing large datasets.



F-1: Example of big data, a graph with many thousands of nodes

One such algorithm used in data mining is called MapReduce. First invented by Google to index the web, it is now used for many different purposes, such as distributed sort and machine learning. Hadoop is a piece of software that implements this algorithm and is used as the basis for our work.

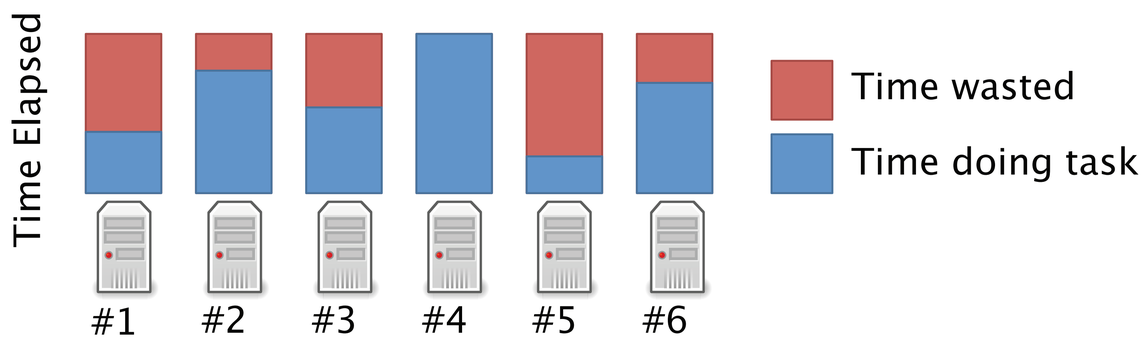
With that, how does the MapReduce algorithm work? MapReduce works with key-value pairs throughout, generating them at the mapping step and using them for each subsequent step. **F-2** demonstrates the simple word count algorithm.



F-2: Word count algorithm example in MapReduce
Courtesy of JTeam/Martijn van Groningen
<http://blog.jteam.nl/2009/08/04/introduction-to-hadoop/>

Existing Approach

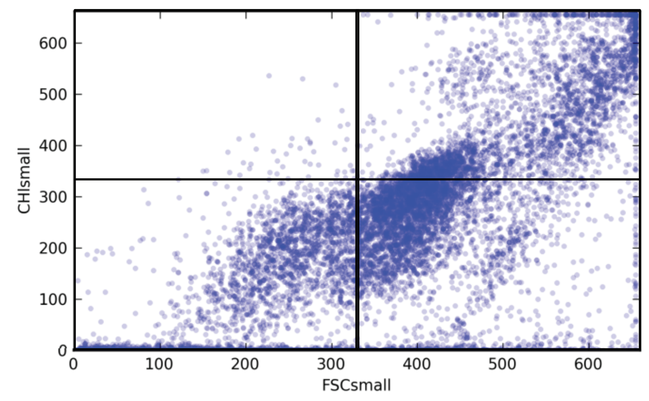
The existing task scheduling algorithm for Hadoop is dumb in that it simply pushes out a new task when an existing one is done. This allows for great *skew*, or the difference in task completion times, to be introduced. **F-3** illustrates the issue of skew, in that node #4 takes longer than everyone else, holding up the whole algorithm. A better task scheduler would reduce the amount of time wasted.



F-3: Illustration of a MapReduce algorithm experiencing skew – computer #4 takes longer than everyone else, holding up the process, while other tasks also complete out of sync

Besides heterogeneous computing environments being the cause of skew, the data itself can also be at fault. Rarely is data uniform, so there can be much skew if certain partitions of data take much longer to complete processing than others.

F-4 exemplifies this with flow cytometry data. Notice how after the data is partitioned, some partitions can be much denser than others – which would affect runtime heavily.



F-4: Flow cytometry data that has been partitioned, notice some partitions are denser than others, which greatly affects runtime speed and can introduce skew
Courtesy of YongChul Kwon, Magdalena Balazinska, Bill Howe, and Jerome Rolia

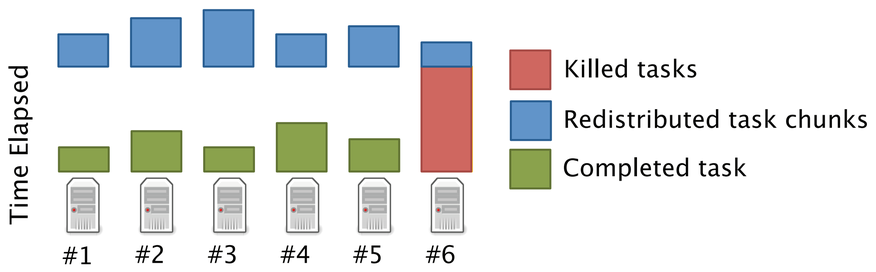
Currently, a research project from the University of Washington known as SkewReduce solves the issue of skew with sample data and cost analysis functions. By knowing how long any one partition will take to execute on any given computers, it can partition the data very optimally with almost no skew. However, that is not an out of the box solution and requires sample data and cost functions. We'd like to have better performance without more work to implement on the user-side.

Our Work

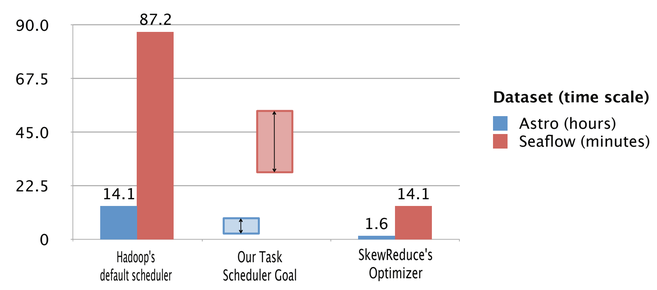
SkewReduce includes an API and optimizer which includes the partitioner and cost functions. The API involves dealing with very specific types of data so we will be ignoring that portion of their work and focusing solely on replacing the optimizer with our own task scheduler that removes the need for sample data and cost functions.

First we needed a good understanding of MapReduce, Hadoop, and SkewReduce. Additionally we required that Hadoop and SkewReduce be working well, since we will be using SkewReduce as a base. After that, how would our algorithm work? We wanted to try a simple yet novel method of optimizing task scheduling. What our task scheduler does is stop tasks on individual computers that have been taking too long in comparison with all other running tasks. Since we also know how far along the task is, we can make an informed decision about stopping the task. Once we know we'd like to stop a task, we re-partition the data for that task so that every available computerin the cluster can get a small chunk of that task. This redistributes the work and "fast-tracks" tasks to completion. **F-5** is an illustration of this process.

Our goal is to have performance about between Hadoop's default scheduler and SkewReduce's optimizer (see **F-6**). We will measure this using the same data SkewReduced used, which is cosmology simulation and flow cytometry data.



F-5: Illustration of our task scheduling algorithm, computer #6 took too long so it was killed and had its work redistributed back to the cluster



F-6: Where we'd like to see performance in overall runtime of our task scheduler in comparison to previous research by SkewReduce authors

Future and Acknowledgements

We'd like to finish our work on the optimized task scheduler so we can compare performance with Hadoop's default scheduler and SkewReduce's optimizer. We also have some other ideas about how to further optimize the task scheduler.

I'd like to thank my mentor Nan Li, Prof. Xifeng Ya, and Shengqi Yan from UCSB. I'd also like to thank the researchers YongChul Kwon and Magdalena Balazinska from the University of Washington with their help with SkewReduce. And, of course, friends and family who helped support me and everyone at INSET that made this posible.

Our work is funded by the Network Science Collaborative Technology Alliance (NS-CTA) in conjunction with the US Army Research Laboratory and INARC.

ALLAN HANCOCK COLLEGE
UCSB
INARC
NETWORK SCIENCE CTA