ORS Project Summary

MapReduce Simulator

Scholars: Andrey Kurenkov, Troy O'Neal, Matthew O'Shaughnessy
Mentor: Xiao Yu
Faculty Advisor: Dr. Bo Hong

Introduction/Background

MapReduce, a popular distributed computing paradigm used for processing extremely large data sets, provides a framework that enables programmers to use distributing computing without having to deal with the many inherent difficulties of parallel processing. Because executions of MapReduce problems can take enormous amounts of time to complete (on the magnitude of hours to days), optimizing the MapReduce setup and implementation is critical to those using this paradigm. Therefore, we intend to design, build, and test a simulator modeled after the architecture of Apache Hadoop, an open source implementation of the MapReduce framework, to estimate the expense of a MapReduce operation without expending the resources that executing it would require. The result of our project will help programmers improve the efficiency of their MapReduce configurations, allow cluster administrators to more effectively manage MapReduce processing clusters, and facilitate work in the Georgia Tech Parallel and Distributed Computing Lab to increase the efficiency of Hadoop.

Description of Project

Our primary goal in the design and construction of the simulator is accuracy and extensibility. It will consider the most important factors for a MapReduce job in order to achieve a good approximation of the job's actual cost with much less computation time. Although we plan to closely follow the Hadoop implementation of the MapReduce framework, we will also create a design that is extensible enough to enable easy modifications to match updates or proposed improvements to Hadoop.

In order to develop a generic and powerful simulation framework, we will follow the common strategy of using discrete events generated by independent modules configured externally from files. Users will be able to specify basic parameters (network speed, job size, etc.) with formatted configuration files that pertinent modules will load and use once simulation is complete. Users will be able to further modify the nature of what they simulate by providing their own implementations of the core modules and specifying they should be used instead of the default implementations. The core modules of the program will simulate or handle the following important MapReduce functions: job description, job execution, distributed system storage, and sophisticated network communications. These modules will all be managed and communicate from the simulation engine, which will be responsible for handling the actual simulation of events send to it by modules and for transferring messages between modules.

To create an accurate model while maintaining a rapid execution time, we will selectively choose what factors will be used to describe the simulation task. In order to ensure accuracy, we will primarily consider the factors that most affect performance: task scheduling algorithms, data storage, task execution, network setup (topology, bandwidth, etc.), and fault tolerance (machine and task failure).

In addition to these simulation considerations, we will also evaluate the best approach towards the actual development of the simulator. Existing simulation frameworks such as SimJava could lessen the time required for us to develop our own framework and increase the accuracy of our simulator. Therefore, we will carefully evaluate the difficulty of implementing each portion of the project by ourselves compared to the ease of use of existing simulation frameworks.

When completed, our simulator will be very useful to cluster administrators overseeing Hadoop setups and programmers attempting to design MapReduce programs. The former will benefit from being able to easily test different configurations, and the latter will be able to verify the feasibility of their projects before fully implementing them. More generally, all users of MapReduce will benefit from not having to implement their own testbench for runtime optimization. Finally, it will significantly aid Hadoop development by giving Hadoop contributors a tool to easily evaluate the performance of new features and updates.

Methods and Procedures

We began with exploratory research into the MapReduce paradigm to help inform both the scope and design of our simulator. Because it is one of the most popular implementations of MapReduce, we first researched the open-source Apache Hadoop framework. Specifically, we explored its design, functionality, and faults through both the source code and documentation in order to better understand how we should design our simulator and the needs it should fulfil. We next examined related works to gain an overview of the designs used by other attempts to create a MapReduce simulator. By examining their papers and source code, we will learn effective strategies used in these other implementations that should will be utilized in our simulator since they have been proven effective in these other works. Lastly, we will evaluate existing simulation engines by implementing a very simplified MapReduce module within them and then extrapolating how easy it would be to use for the rest of the simulator.

Next, we will design the software framework of our simulator to enable later modification and maintenance. We will take a typical object oriented approach by specifying several fundamental interfaces and their interactions through the simulation engine, and will possibly do this within an existing framework based on our previous research. The most important interface will be of Module, and the main modules will reflect the core concerns we outlined earlier: job description, job scheduling, job execution, storage system and network system. We will initially create default implementations of all these modules following the design on Hadoop. Afterwards, we will extend our framework to support more ambitious requirements: wider range of job specification; different scheduling policies; variations on job

execution; other storage systems than HDFS, such as Amazon S3; and different network topologies. The framework design will be modified or rewritten if the design is not general enough to support all of these extensions.

Our simulator will strive to be capable of describing any Hadoop job through the use of configuration files containing key-value pairs similar to Hadoop's. Many user-defined preferences for each execution will be described and specified in these configuration files.

We will evaluate our simulator by comparing its results (projected job runtime) and simulator execution time to other simulators as well as an actual Hadoop execution using typical MapReduce applications such as word count (counts the frequency of each word in a dataset), sort (sorts large datasets), and applied applications such as Twitter (generates networks of twitter followers). Finally, when we are satisfied with the accuracy of the simulator, we will use it to test the efficiencies of different hardware systems and network topologies with Hadoop jobs and evaluate different modifications to the Hadoop system.

Expected Timetable

- September/October—basic familiarity with Hadoop; begin developing project design; review existing works on subject; research and evaluate existing tools to help us with our implementation
- November—Develop basic framework for simulator (implementation mimicking Hadoop)
- December—Begin developing more accurate models for networking and processing
- January—Add pluggable schedulers; evaluate/iterate on our interface design
- February and beyond—Compare our simulator to simulators developed by others and actual Hadoop runtimes; evaluate different Hadoop features with our simulator

Individual Contribution

The overall design will be completed collaboratively by all members in the team. Each team member will then be responsible for designing, implementing, and testing the accuracy of one or more components of the simulator. The focus, scope, and implementation of experiments to test the simulator will be designed collectively by all members. Afterwards, each member will implement a MapReduce benchmark application with which to evaluate the simulator's result. Following that, each member will create several iterations/extensions of the component they implemented and compare their performances. When the project is complete, all team members will contribute to the final report.