Weston Odom

16 November 2013

Cloud Computer – Paper Review

Vrbsky

Review for “MapReduce for Data Intensive Scientific Analyses” by Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox

They authors of this paper were interested in processing scientific data that fit into what they called "the class of composable applications". Their reasoning was that as time goes on, the sheer amount of data being produced in fields such as astronomy and physics is immense and demands efficient computation techniques.

They looked at three potential approaches – Threads, Message Passing Interface (MPI), and MapReduce – and decided on MapReduce for the following reasons:

* Its relaxed synchronization constraints do not impose much of an overhead for large data analysis tasks.
* The simplicity and robustness of the programming model supersede the additional overheads.

The goal of the paper research was to “compare the performance of these implementations in the context of these scientific applications and make recommendations regarding the usage of MapReduce techniques for scientific data analyses.”

The authors gave a brief description of Google’s MapReduce implementation and Hadoop, then gave a somewhat lengthier introduction to CGL-MapReduce, an implementation created by the authors.

It looks like the main contribution that the author’s work has is that their implementation of MapReduce was customized for the way they wanted to load and save data, resulting in performance gains for particular types of data sets over established implementations like Hadoop. In a test on one terabyte of data using 12 compute nodes, CGL-MapReduce was about 17 minutes faster (91 minutes vs 108 minutes). Additionally, they listed three observations that were confirmed by their test results:

1. Most scientific data analyses that have some form of SMPD algorithm can benefit from the MapReduce technique to achieve speedup and scalability.
2. As the amount of data and the amount of computation increases, the overhead induced by a particular runtime diminishes.
3. Even tightly coupled applications can benefit from the MapReduce technique if the appropriate size of data and an efficient runtime are used.

There were a couple of things that gave me pause when reading the paper. The first thing was that this really doesn’t seem like a very innovative idea. I don’t mean to say that it’s not a good idea – it’s just that it seems like a natural extension of the programming paradigm rather than an important research topic.

The second thing was that the authors said "Since the overall performance is limited by the I/O bandwidth, we use only one processor core in each node for this evaluation." This led me to think that a potentially more valuable research topic would be something to do with increasing I/O efficiency in these scenarios.

As far as questions, I would be curious to know how the time required for I/O could be reduced. As it was the bottleneck for some of their exercises, this seems like it would be a logical next step. With regards to the first type of data analyzed – the physics data histograms – would it have been possible to analyze the data prior to histograms being formed, and would this have had a significant impact on the time required to compute the results?