Performance of Data Intensive Supercomputing Runtime Environments

Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox
Department of Computer Science
Indiana University, Bloomington, IN 47404, USA
{jekanaya,spallick,gcf}@indiana.edu

1. Research Problem

Computation and data intensive scientific data analyses are increasingly prevalent. In the near future, it is expected that the data volumes processed by applications will cross the peta-scale threshold, which would in turn increase the computational requirements. Two exemplars in the data-intensive domains include High Energy Physics (HEP) and Astronomy. HEP experiments such as CMS and Atlas aim to process data produced by the Large Hadron Collider (LHC). The LHC is expected to produce tens of Petabytes of data annually even after trimming the datasets via multiple layers of filtrations. In astronomy, the Large Synoptic Survey Telescope produces data at a nightly rate of about 20 Terabytes.

Data volume is not the only source of compute intensive operations. Clustering algorithms used for DNA sequencing are especially compute intensive even though the datasets are comparably smaller than the physics and astronomy domains.

Efficient parallel/concurrent algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such scientific data analysis. Most of these analyses can be thought of as a Single Program Multiple Data (SPMD) algorithm or a collection thereof. These SPMDs can be implemented using different techniques such as threads, message passing, and map-reduce[1]. Additionally, these SPMDs can also be deployed in various hardware configurations such as compute clusters, computational grids, and compute clouds.

There are several considerations in selecting an appropriate implementation strategy for a given data analysis. These include data volumes, computational requirements, algorithmic synchronization constraints, quality of services, easy of programming and the underlying hardware profile.

The goal of this research is to study the various parallelization techniques which can be applied to the SPMD algorithms. It is also important to understand how these different techniques can be used to improve the scalability and the performance of the scientific data analyses. Finally, this research will also make recommendations about possible improvements to these parallelization techniques.

2. Current Research

Current research focuses on designing an efficient map-reduce implementation and using it for scientific data analyses in different hardware configurations. Our research prototype –CGL-MapReduce – utilizes NaradaBrokering [2], a streaming-based content dissemination network developed by us, for all the underlying communications. The use of streaming provides performance improvements over the file-based communications adopted by the typical map-reduce implementations such as Hadoop [3]. Additionally, CGL-MapReduce (depicted in figure 1) supports iterative map-reduce computations while adding acceptable overheads to the computation task.

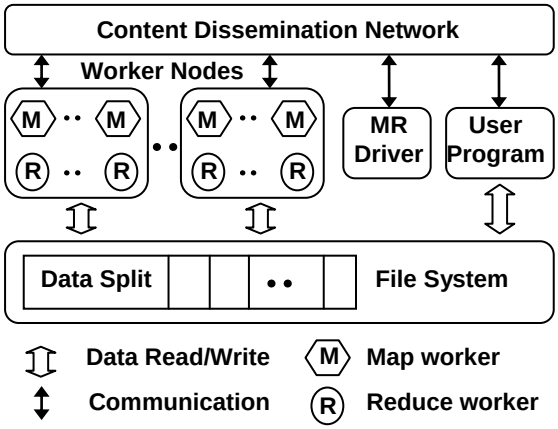

Figure 1 Components of the CGL-MapReduce

We have successfully tested CGL-MapReduce with two scientific data analyses; (i) High Energy Physics data analysis involving large volumes of data representing a single phase map-reduce computation and (ii) Kmeans clustering [4] representing an iterative map-reduce computation.

3. Results

Here we present the results of benchmarking CGL-MapReduce with scientific applications. We compare the performance of CGL-MapReduce with Hadoop for both HEP and Kmeans. For Kmeans clustering, we also compare the results with an MPI version of the same algorithm to show how these techniques

converge in performance for large data sets. Our results are depicted in figures 2 through 5.

Figure 2 describes the HEP data analysis task and how it is implemented as a map-reduce computation while Figure 3 compares the performance of CGL-MapReduce and Hadoop for this computational task.

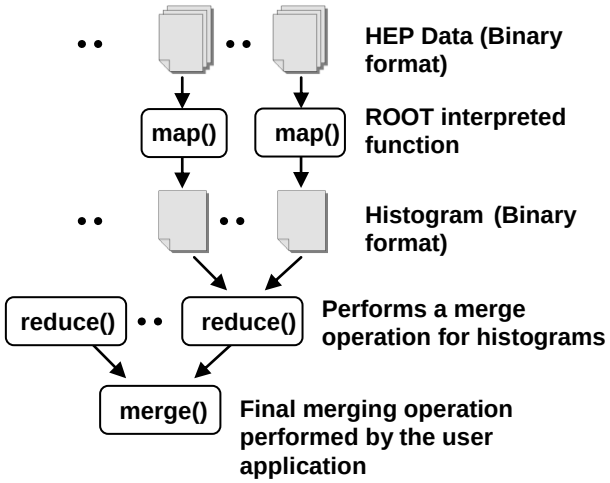

Figure 2. Map-reduce implementation of the HEP data analysis

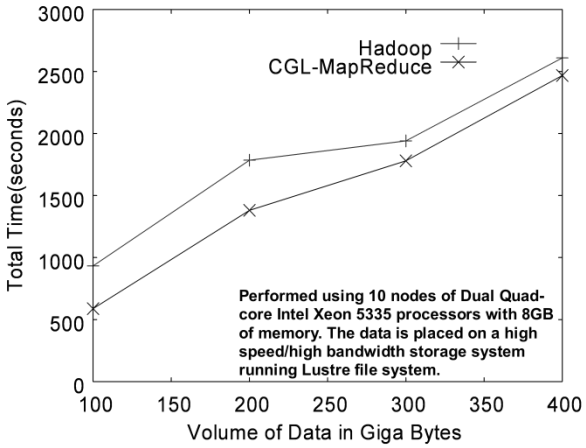

Figure 3. Hadoop vs. CGL-MapReduce for HEP Data Analysis.

Figure 4 describes the map-reduce version of the Kmeans clustering algorithm, which we use to cluster a large number of 2D data points. In figure 5 we compare the performance of Hadoop, CGL-MapReduce, and MPI versions of Kmeans.

Results in Figure 3 indicates that the overhead associated with Hadoop's robustness diminishes for large data sets while the results in Figure 5 shows how this overhead effect iterative map-reduce computations. Figure 5 also highlights that the performance of our stream based map-reduce implementation is comparable to MPI for large enough data sets.

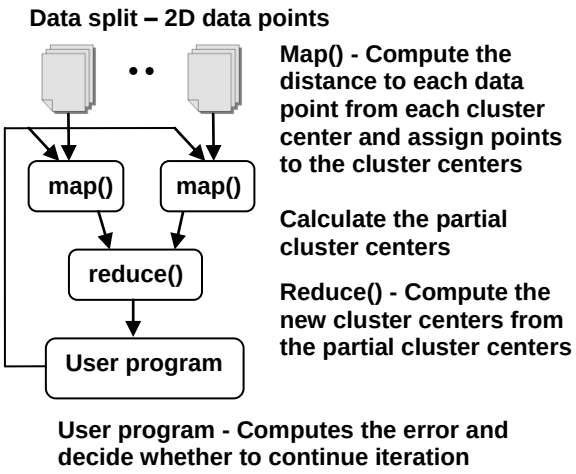

Figure 4. Map-reduce implementation of the Kmeans clustering algorithm

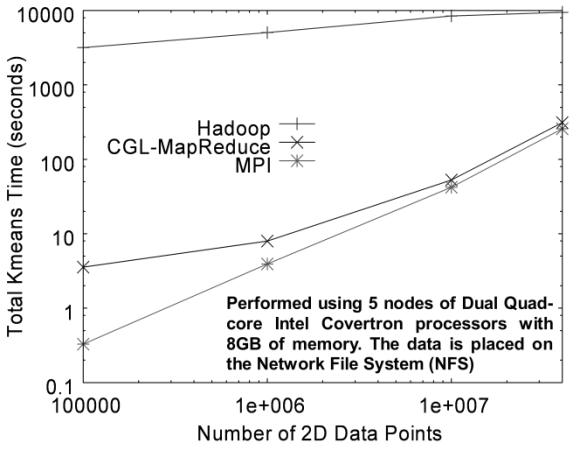

Figure 5. Kmeans clustering using Hadoop, CGL-MapReduce, and MPI (Both axes are in log scale)

4. Future Research

Future research involves analyzing the different parallelization techniques under cloud computing environments and understanding how these (and other similar) applications can be efficiently implemented for cloud computing environments.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, December 2004.

[2] S. Pallickara, G. Fox, "NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids," *Middleware 2003*, pp. 41-61

[3] Hadoop, http://hadoop.apache.org/core/

[4]J. B. MacQueen , "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1:281-297