Can Commercial BigData Ideas Benefit Analysis of Instrument Data?

Gagan Agrawal
Computer Science and Engineering
The Ohio State University, Columbus, OH 43210
{agrawal}@cse.ohio-state.edu

Introduction

Analysis of data from large-scale instruments often requires *in-situ* or *streaming* analytics. Moreover, it also requires high level of parallelism to allow analysis of large-scale data with either real-time constraints, or otherwise with acceptable response times.

Both streaming data analysis and in-situ data analysis have received attention. Particularly, in the HPC community, in-situ analysis has been a subject of significant research. This work has been in the context of analyzing simulation outputs, and not instrument data analysis. The landscape of the current in-situ analytics research mainly falls into two levels: 1) in-situ algorithms at the *application level*, including indexing, compression, visualization, and other analytics; and 2) in-situ resource scheduling platforms at the *system level*, which aims to enhance resource utilization and simplify the management of co-located analytics code. These in-situ middleware systems mainly play the role of a *coordinator*, aiming to facilitate the underlying scheduling tasks, such as cycle stealing and asynchronous I/O.

Despite a large volume of recent work in this area, an important question remains almost completely unexplored: "*can the applications be mapped more easily to the platforms for in-situ and/or streaming analytics*?". In other words, we posit that *programming model* research on in-situ/streaming analytics is needed. Particularly, in-situ algorithms are currently implemented with low-level parallel programming libraries such as MPI, OpenMP, and Pthread, which offer high performance but require that programmers manually handle all the parallelization complexities.

Streaming and in-situ applications (or at least the ones documents in the literature) perform statistical analysis, feature extraction, data mining, preprocessing, or other closely related tasks. For this class of applications, and from the programmability (and not necessarily performance) view-point, MapReduce is the most widely adopted programming model [3]. Not only the MapReduce API simplifies parallelization of an application, but also MapReduce implementations handle much of scheduling, task management, and data movement. However, performance of these frameworks has usually not matched that of MPI-based systems. *Overall, it is an interesting question whether a MapReduce-like framework can support analysis of (streaming) large-scale scientific data, improving productivity and not compromising the performance.*

Our Middleware Series

Our group at Ohio State has developed a series of middleware systems providing MapReduce(-like) APIs [4, 2, 5]. However, unlike the existing commercial frameworks, our systems have been developed in languages like C or C++. Moreover, these systems involve a number of design choices that improve performance. One of the common features is that there is no need to load data into a specialized file system (like the Hadoop Distributed File System (HDFS)). As another example, a common design choice has been the use of a variant of the original MapReduce API, to ensure that key-value pairs need not be emitted. Our optimization achieves reduction in memory requirements well beyond what is possible with combination functions.

Also noteworthy is the most recent version of the system that focuses on in-situ analysis of the output of a scientific simulation [5]. Our system can support a variety of scientific analytics on simulation nodes, with minimal modification of simulation code and without any

specialized deployment (such as installing HDFS). Compared with traditional MapReduce frameworks, our middleware supports efficient in-situ processing by accessing simulated data directly from memory in each node of a cluster or a distributed memory parallel machine. To address the mismatch between parallel programming view of simulation code and sequential programming view of MapReduce, our middleware can be launched from parallel (OpenMP and/or MPI) code region once each simulation output partition is ready, while the global analytics result can be directly obtained after the parallel code converges. Further, we have developed both *time sharing* and *space sharing* modes for maximizing the performance in different scenarios. Additionally, for memory-intensive window-based analytics, we improve the in-situ efficiency by supporting early emission of *reduction object*.

Application to Analysis of Instrument Data

In a collaboration with Argonne National Labs, a middleware from our group has been successfully used for analysis of data (image reconstruction) of the output from x-ray tomography systems [1]. The context of the work was as follows. The x-ray tomography systems available at the imaging beamlines of the Advanced Photon Source (APS, located at Argonne National Laboratory) are routinely used in materials science applications where high-resolution and fast 3D imaging are instrumental in extracting valuable information. Scientists often want *quasi-instant* feedback so that they can check results and adjust the experimental setup. More specifically, Quasi-instant feedback can help identify optimal experimental parameters (beamline condition and sample environment such as temperature and pressure) and accelerate the end-to-end scientific process. Specifically, our work developed two parallelization techniques, `per-slice` and `in-slice`, for tomographic reconstruction algorithms. The MapReduce-like framework, MATE [4], is extended and optimized to help implement these parallel methods efficiently. We extensively evaluate the proposed methods and middleware-based implementations (for different reconstruction algorithms and real-world datasets). Our experimental results show that our middleware can scale almost linearly up to 8K cores and can achieve execution times on 32K cores that are $\geq$ 95.4% less than those on 1K cores.

1. REFERENCES

[1] T. Bicer, D. Gürsoy, R. Kettimuthu, F. D. Carlo, G. Agrawal, and I. T. Foster. Rapid tomographic image reconstruction via large-scale parallelization. In *Euro-Par 2015: Parallel Processing - 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, August 24-28, 2015, Proceedings*.

[2] L. Chen and G. Agrawal. Optimizing MapReduce for GPUs with Effective Shared Memory Usage. In *Proceedings of Conference on High Performance Distributed Computing (HPDC)*, June 2012.

[3] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, pages 137–150, 2004.

[4] W. Jiang, V. Ravi, and G. Agrawal. A Map-Reduce System with an Alternate API for Multi-Core Environments. In *Proceedings of Conference on Cluster Computing and Grid (CCGRID)*, 2010.

[5] Y. Wang, G. Agrawal, T. Bicer, and W. Jiang. Smart: A mapreduce-like framework for in-situ scientific analytics. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, 2015.