A Collective Communication Layer for the Software Stack of Big Data Analytics

Bingjing Zhang
School of Informatics and Computing
Indiana University
Bloomington, IN, USA
zhangbj@indiana.edu

***Abstract*—The landscape of distributed computing is rapidly evolving, with computers exhibiting increasing processing capabilities with many-core architectures. Almost every field of science is now data driven and requires analysis of massive datasets. The algorithms for analytics such as machine learning can be used to discover properties of a given dataset and make predictions based on it. However, there is still a lack of simple and unified programming frameworks for these data intensive applications, and many existing efforts are designed with specialized means to speed up individual algorithms. In this thesis research, a distributed programming model, MapCollective, is defined so that it can be easily applied to many machine learning algorithms. Specifically, algorithms that fit the iterative computation model can be easily parallelized with a unique collective communication layer for efficient synchronization. In contrast to traditional parallelization strategies that focus on handling high volume input data, a lesser known challenge is that the shared model data between parallel workers, is equally high volume in multi-dimensions and required to be communicated continually during the entire execution. This extends the understanding of data aspects in computation from in-memory caching of input data (e.g. iterative MapReduce model) to fine-grained synchronization on model data (e.g. MapCollective model). A library called Harp is developed as a Hadoop plugin to demonstrate that sophisticated machine learning algorithms can be simply abstracted with the MapCollective model and conveniently developed on top of the MapReduce framework. K-means and Multi-Dimensional Scaling (MDS) are tested over 4096 threads on the IU Big Red II Supercomputer. The results show linear speedup with an increasing number of parallel units.**

I. Introduction

Data analytics is undergoing a revolution in many scientific domains owing to the incredible volume of data. Processing such huge data usually exceeds the capability of a single or even a few machines and thus requires parallelization at an unprecedented scale. Machine learning algorithms [1] are a type of algorithm for analytics which produce a model that generalizes beyond the training instances. They have been widely used in computer vision, text mining, advertising, recommender systems, network analysis, and genetics. Unfortunately, scaling up these algorithms is challenging owing to their prohibitive computation cost: not only the need to process enormous training data in iterations, but also the requirement to communicate a large model data continually in order to derive a converged model result.

Many machine learning algorithms were implemented in MapReduce [2][3][4][5]. However, these implementations suffer from repeated slow disk communication. This motivated the design of iterative MapReduce tools [6][7] which utilize memory for data caching and communication and thus drastically improve the performance of big data processing. Later on these tools expanded rapidly and formed ecosystems of software stacks, e.g. Apache Big Data Stack (ABDS) [8], with different computation models which are not limited to MapReduce and iterative MapReduce, but also include Graph models.

While the contemporary tools were improved with in-memory communication, observations show that the performance of iterative machine learning applications still suffers from the overhead of repeated large model data synchronization. To solve this problem, a proposed approach is to use collective communication [9] to improve the efficiency of model synchronization. Many existing tools are designed with fixed data abstractions and limited communication support. Therefore a separate collective communication layer is introduced which provides model data abstraction and optimized collective communication operations. It also defines a MapCollective model which serves the diverse communication demands in applications. To adapt to current ABDS, these enhancements are designed as a plug-in to Hadoop called Harp [10]. With improved expressiveness and excellent performance on collective communication, a high performance ABDS (HPC-ABDS) can simultaneously support various applications.

On top of Harp, a machine learning library is built to present the effectiveness of using collective communication in machine learning applications. The focus is to understand the features of the model data communication in the algorithms and build the correct collective communication abstractions to match the applications. K-means [11] and MDS [12] are two examples.

In the following sections, Section 2 introduces related work. Section 3 goes into details on the design of the Harp plug-in. Section 4 presents the initial results of the machine learning library. The conclusion is given in Section 5.

II. Related Work

Initially MapReduce proved to be a very successful programming model for large-scale data processing but was later considered to be insufficient for supporting sophisticated analytics, especially those machine learning algorithms involving iterations. Frameworks like Twister [6] and Spark [7] solved this issue by caching intermediate data and developing the iterative MapReduce model. Another iterative computation model is the graph model, which abstracts data as vertices and edges and executes in BSP (Bulk Synchronous Parallel) style. Pregel [13], Giraph [14] and GraphLab [15] follow this design. GraphLab was further enhanced with PowerGraph [16] abstraction to reduce the communication overhead. In addition, Parameter Server [17] and Petuum [18] allow users to program machine learning algorithms with "push" and "pull" operations on model data.

The main issue is that all these tools have little or no support to model data abstraction or model data collective communication. In Twister/Spark, the existing work focuses on processing the input data efficiently. Although some research work [19][20][21][22] tries to add or improve collective communication operations, they are limited in operation types and constrained by the computation flow. The model data is viewed as a transient intermediate data and often required to be gathered back to the client and redistributed again. As a result, these tools don't scale well for large model data problems. In graph based tools, the training data abstractions and the model data abstractions are mixed in a graph presentation. The restriction of vertex/edge based communication limits the possibility of using optimized collective communication. In Parameter Server/Petuum, though model data abstractions exist, the communication is abstracted at a low level and requires further optimization.

III. Harp

The central idea of this research is to unify collective communication abstractions, optimize collective communication operations, and match them to machine learning applications. There include the following contributions.

Firstly a collective communication library layer is abstracted and separated out from other layers in the software stack of big data processing. In this layer, a set of abstractions for model data and related collective communication operations for synchronization are defined. Data are horizontally abstracted as arrays, key-values, or vertices/edges, and constructed from basic types into partitions and tables vertically. Communication includes operations abstracted from MPI, MapReduce, graph-based tools, and communication patterns from popular machine learning frameworks.

Secondly, on top of this abstraction layer, the MapCollective programming model is defined, which follows the BSP style and allows users to invoke collective communication operations to synchronize parallel workers. There are two levels of parallelism: inter-node level and intra-node level. Failure recovery poses a challenge because the execution flow in the MapCollective model is very flexible. Currently job level failure recovery is applied. Worker-level recovery by resynchronizing execution states between new launched workers and other old live workers is also under investigation.

All these ideas have been implemented in the Harp open source library. By plugging Harp into Hadoop, the MapCollective model can be expressed in a MapReduce framework and enhance Hadoop with efficient in-memory collective communication that enables support for a variety of machine learning applications (see Fig. 1).

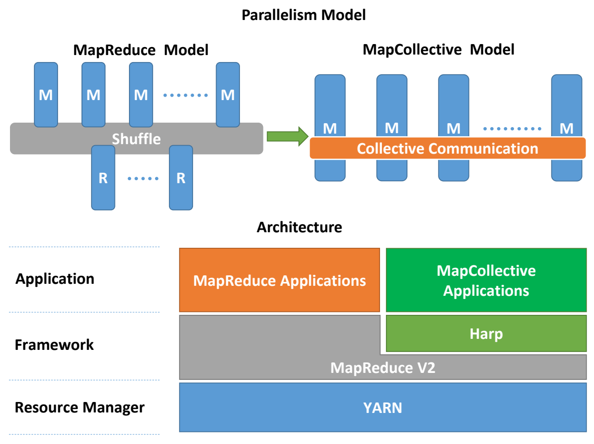

Fig. 1. The Parallelism Model and Plug-in Concept of Harp

IV. Machine Learning Library

The machine learning library is built on top of Harp. The structure of model data, the computation and the communication patterns in machine learning applications are studied and categorized into different computation models. Because of the high frequencies of model communication, caching the model data to memory and performing in-memory communication is necessary. In addition, optimized collective communication on in-memory model data can improve model update rate and output a converged model result in shorter time than other communication methods.

Here K-means and MDS are two applications implemented as examples (see Table. I). Both K-means and MDS are tested on Big Red II [23]. K-means runs on 500M 3D points with 10K clusters and 5M 3D points with 1M clusters. The execution times and speedup are shown in Fig. 2a and Fig. 2b. The speedup is close to linear. MDS runs with 100K, 200K, 300K and 400K points, each of which represents a gene sequence [24]. The execution times and speedup are seen in Fig. 2c and Fig. 2d. In most cases, the speedup lines are close to linear. Only 100K problems show low efficiency. This is a standard effect in distributed computing where the small problem size reduces compute time compared to communication. The research of other machine learning applications with big model data such as LDA [25] is ongoing.

V. Conclusion

The research shows that a collective communication layer on the software stack of big data analytics can help abstracting model data and expressing model data communication in iterative machine learning applications. Future research directions in the next 6 to 12 months will focus on understanding

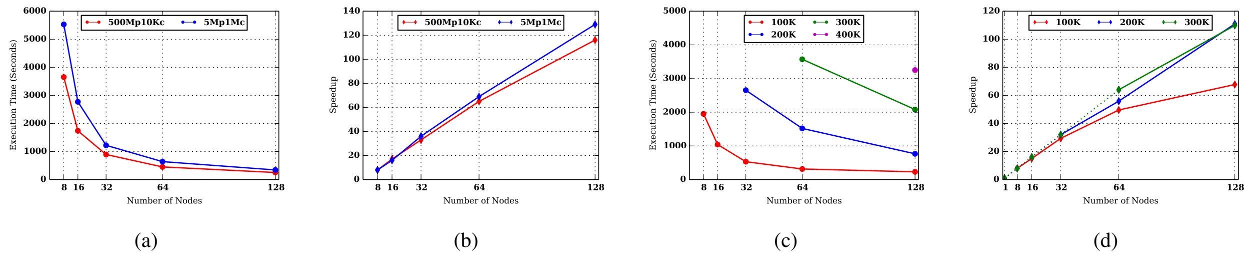

Fig. 2. (a) Execution Time of K-Means (b) Speedup of K-Means (c) Execution Time of MDS (d) Speedup of MDS

TABLE I
MACHINE LEARNING APPLICATIONS ON TOP OF HARP

Application	Model Size	Collective Communication
K-means Clustering	Usually in MB level, but can grow to GB level	allreduce
WDA-SMACOF	A few MBs	allgather & allreduce
LDA	From a few GBs to 10s of GBs, or even larger	other model sync methods

the characteristics of data, computation and communication patterns in many other machine learning applications. The current study imply that communication plays an important role in iterative computation. Similar features may apply to other applications and result in a general and effective communication abstraction in many machine learning applications.

ACKNOWLEDGMENTS

The author gratefully acknowledges support from Intel Parallel Computing Center (IPCC) Grant, NSF 1443054 CIF21 DIBBs 1443054 Grant, and NSF OCI 1149432 CAREER Grant; The author appreciates the system support offered by FutureSystems at Indiana University.

REFERENCES

[1] R. Kohavi and F. Provost, "Glossary of terms," http://ai.stanford.edu /~ronnyk/glossary.html.
[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[3] "Hadoop," http://hadoop.apache.org.
[4] C.-T. Chu *et al.*, "Map-reduce for machine learning on multicore," in *NIPS*, vol. 19, 2007, p. 281.
[5] "Apache Mahout," https://mahout.apache.org.
[6] J. Ekanayake *et al.*, "Twister: a runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 810–818.
[7] M. Zaharia *et al.*, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, vol. 10, 2010, p. 10.
[8] S. Kamburugamuve, "Survey of apache big data stack," Indiana University, Tech. Rep., 2013, http://grids.ucs.indiana.edu/ptliupages/publicat ions/survey_apache_big_data_stack.pdf.
[9] E. Chan *et al.*, "Collective communication: theory, practice, and experience," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 13, pp. 1749–1783, 2007.
[10] B. Zhang, Y. Ruan, and J. Qiu, "Harp: collective communication on hadoop," in *Proceedings of IEEE International Conference on Cloud Engineering (IC2E)*, 2015.
[11] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.
[12] Y. Ruan and G. Fox, "A robust and scalable solution for interpolative multidimensional scaling with weighting," in *IEEE 9th International Conference on eScience*, 2013, pp. 61–69.
[13] G. Malewicz *et al.*, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 135–146.
[14] "Giraph," https://giraph.apache.org.
[15] Y. Low *et al.*, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.
[16] J. E. Gonzalez *et al.*, "PowerGraph: distributed graph-parallel computation on natural graphs," in *OSDI*, vol. 12, 2012, p. 2.
[17] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *OSDI*, 2014, pp. 583–598.
[18] E. P. Xing *et al.*, "Petuum: a new platform for distributed machine learning on big data," in *KDD*, 2013.
[19] J. Qiu and B. Zhang, "Mammoth data in the cloud: clustering social images," in *Clouds, Grids and Big Data*, ser. Advances in Parallel Computing. IOS Press, 2013.
[20] B. Zhang and J. Qiu, "High performance clustering of social images in a map-collective programming model," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013.
[21] M. Chowdhury *et al.*, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 98–109, 2011.
[22] T. Gunarathne, J. Qiu, and D. Gannon, "Towards a collective layer in the big data stack," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 2014, pp. 236–245.
[23] "Big Red II," https://kb.iu.edu/data/bcqt.html.
[24] Y. Ruan *et al.*, "Integration of clustering and multidimensional scaling to determine phylogenetic trees as spherical phylograms visualized in 3 dimensions," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 2014, pp. 720–729.
[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–102, 2003.