**Less is More: Solving Spatial Big Data Problems with Small Clusters using Harp**

Judy Qiu

Indiana University

Many scientific problems depend on the ability to analyze and compute on large amounts of data. This analysis often does not scale well; its effectiveness is hampered by the increasing volume, variety and rate of change (velocity) of big data. The NSF “Big Data Middleware and High Performance Analytics Libraries for Scalable Data Science” http://www.spidal.org/ (DIBBs) project is designing, developing and implementing software and algorithm library building blocks that will enable a fundamental improvement in the ability to support data intensive analysis on a broad range of cyberinfrastructure, including that supported by NSF for the scientific community.

Three applications of the project are in particular related to spatial geographical information systems: one is led by Kansas University analyzing 3D Polar region radar data. Additionally, we are studying spatial clusters of diseases discovery using GIS locations (at patient level) to monitor public health problems [1]; the same tools search for spatial objects (at microanatomic level) in digital pathology images. For a typical study, hundreds to thousands of images are used and each pathology image could have 10 billion pixels to extract a million spatial objects from 100 million features (dozens to 100 features per object) per image. Indiana University and Stony Brook University are collaborating on developing scalable spatial clustering methods to support biomedical applications driven by pathology imaging and GIS oriented public health studies.

An open research challenge for the many communities including Geospatial Data Science is whether common big data frameworks can be developed for their scientific applications and how effectively do they run on computer infrastructures such as traditional HPC clusters and cloud platforms. We expect that spatial big data analysis will be built around optimized libraries for important kernels. Twister [2] introduced iterative computation using long-running processes or threads with in-memory caching of invariant data. We explored collective communication (e.g. broadcast, scatter, gather or reduce) on Twister and Twister4Azure, showing different implementations of abstractions on different infrastructure (HPC or Azure). Rather than building full software environments, we built on the Apache Big Data Stack for the next step, adding a separate communication abstraction where Harp [3] prototypes introduce the Map-Collective concept as a plug-in to Hadoop Ecosystem. Harp adds both Map-Collective and data structure (key-value, graph, array, pixel) abstractions to Hadoop. Table 1 summarizes different Harp communication and synchronization patterns with MPI quality but optimized for data analytics rather than simulations.

Parallelization of an application consists of a set of Map tasks which are synchronized with collective communication operations. While the input data is abstracted and partitioned as KeyValue pairs, the abstraction of the synchronized model data and related collective communication operations are specially defined. These ideas are implemented in the Harp library (open source) as a Hadoop plugin. By plugging Harp into Hadoop, the MapCollective model can be expressed on top of a MapReduce framework and efficient data synchronization for a variety of data analytics and machine learning applications is enabled. In addition, mapping a MapCollective model to Hadoop also enables two levels of parallelism. Since each Map task is a process where the collective communication operations are invoked; inside a process, multi-thread execution is enabled for fine-grained parallelism that is portable to new manycore and GPU architectures.

Table 1: Collective Operations supported in Harp

We discuss three applications using Harp: K-Means clustering, Force-directed Graph Drawing, and Multi-dimensional Scaling. The first two algorithms use a single collective communication operation per iteration while the third has nested iterations, and two different collective communication operations are used alternately. In data abstraction, the first and third algorithms use array abstraction, while the second one utilizes graph abstraction. We have shown all three applications can be simply expressed with the combination of collective communication operations and the Map-Collective model. This is illustrated by Hadoop (with the Harp plug-in) which can run K-means, Graph Layout, and Multidimensional Scaling algorithms with realistic application datasets over 4096 cores on the IU Big Red II Supercomputer (Cray/Gemini) while achieving linear speedup (see Figure 1).

|  |  |  |
| --- | --- | --- |
| K-means Cluster | Forced-directed Graph Drawing | Multi-Dimensional Scaling (WDA-SMACOF) |
|  |  |  |  |
| Figure 1 Performance of Hadoop/Harp Applications on the Big Red II Supercomputer |

**References**

1. Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox, Twister: A Runtime for Iterative MapReduce, in proceedings of the 19th ACM International Symposium on High Performance Distributed Computing conference (HPDC), June 15-19, 2010.
2. Bing Zhang, Yuan Ruan, and Judy Qiu, “Harp: Collective Communication on Hadoop,” in proceedings of IEEE International Conference on Cloud Engineering (IC2E), March 9-12, 2015.