Achieving up to Zero Communication Delay in BSP-based Graph Processing via Vertex Categorization

Xuhong Zhang
EECS, University of Central Florida
Orlando, Florida 32826
Email: xzhang@eecs.ucf.edu

Ruijun Wang
EECS, University of Central Florida
Orlando, Florida 32826
Email: ruijun@eecs.ucf.edu

Jun Wang
EECS, University of Central Florida
Orlando, Florida 32826
Email: jwang@eecs.ucf.edu

***Abstract*—Graph is widely used to model structural relationships among objects. For example, Web graph, social networks, knowledge bases and protein interactions are all modeled with graph. Facebook's social graph has scaled to trillions edges. The ever-increasing size of these real-world graph data are making analytics on them extremely challenging. Traditional MapReduce framework is not efficient at graph processing due to the special features of graph structures and algorithms. To better utilize these features, Google proposes Pregel. Pregel's model is very popular and leads to the emergence of many current widely used graph processing frameworks such as Apache Hama, Apache Giraph, GPS, GraphLab, and Mizan. All these Pregel-like system are implemented based the Bulk Synchronous Parallel (BSP) model. Graph algorithm is divided into supersteps, within each superstep, each vertex runs the same compute function concurrently. This function will first do computation to update vertex value, then asynchronously send messages to neighbor vertices. Vertices of graph are usually divide into partitions across a cluster of machines. From a higher perspective, one superstep on a machine can be viewed as two overlapping phase of computation and communication. To coordinate the parallel execution, each machine will finally proceed to an universal barrier and then wait for the completion of other machines. After all machines reach the barrier, a new superstep is started. This synchronization implementation is simple enough, but the barrier is too coarse grained. As observed, once a machine finishes computation phase of a superstep, it will do no computation other than communication till the start of next superstep. Here, we regard this time frame as communication delay. This communications delay dominates a superstep. Figure 1 shows the time break down of multiple supersteps in running PageRank on Twitter graph (41.7 million vertices, 1.47 billion edges).**

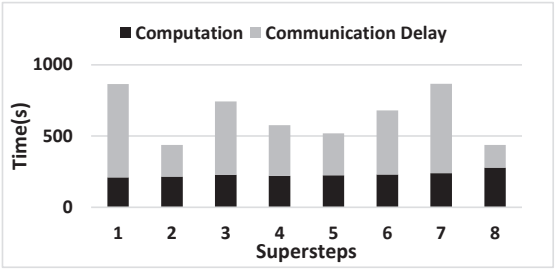

Fig. 1. Communication delay in supersteps when running PageRank on Twitter graph

This communication delay causes serve resource underutilization. So we investigate whether computation can be scheduled during this communications delay by relaxing the BSP synchronization model. Two most important synchronization properties are maintained by the BSP graph processing,

- **Consistency: At the beginning of each superstep, a vertex's computing function can be triggered if and only if all its incoming messages from neighbors have been received.**
- **Isolation: Within the same superstep, newly generated messages from any vertex will not be seen by any other vertex.**

We observe that a large portion of vertices in a graph satisfy the consistency property right after finishing computation without synchronizing at the barrier, simply because these vertices' incoming neighbors all reside on the same partition. The incoming messages of these vertices are instantly available in memory right after its belong partition finishes computation. While keeping the isolation property, the next superstep on these vertices can be directly started without synchronizing at the costly barrier. Thus some computation can be scheduled during communications delay in current superstep. In this paper, we categorize these vertices as *local vertices* and others as *remote vertices*. Figure 2 shows the percentage of local vertices when partitioning the soc-LiveJournal (4.8 million vertices, 68 million edges) data set into different partitions using Metis. Even 256 partitions still yield more than 50% local vertex. So a large portion of

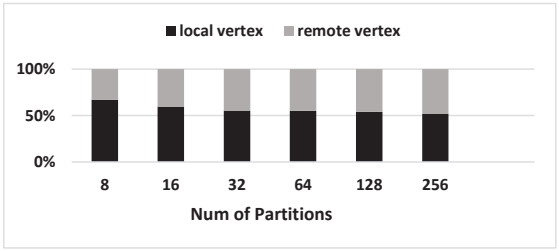

Fig. 2. Local vertex percentage. Local vertex is vertex with all its incomming neighbors residing on the same partition

computation in the next superstep can be scheduled during the communication delay in current superstep. Through this overlapping of computation and communication, our solution can dramatically mitigate the communications delay caused by BSP synchronization. Our proposed solution could totally mitigate the communication delay when it's less than computation time in a superstep multiply local vertex's percentage