Large Scale Complex Network Analysis Using the Hybrid Combination of a MapReduce Cluster and a Highly Multithreaded System

Seunghwa Kang and David A. Bader
Georgia Institute of Technology
Atlanta, GA, 30332, USA

Introduction

Complex Networks abstract interactions among entities in a wide range of domains in a graph representation, and analyzing complex networks often solves many real-world problems. For example, Albert et al. [1] study the impact of a power-law degree distribution, which is a common feature in many complex networks, on the vulnerability of the Internet.

Analyzing large scale complex networks, however, imposes difficult computing challenges. Many graphs that represent complex networks have millions to billions of vertices and edges. These graphs are often embedded in raw (and often streaming) data of terabytes to petabytes. Extracting the compact representation of a graph or a subgraph from large volumes of raw data is a significant challenge. Extracted graphs are also highly irregular and have only a low degree of locality than traditional graphs derived from physical topologies. This stresses traditional hierarchical memory subsystems as well.

We present a hybrid system of a MapReduce cluster and a highly multithreaded system as an effective tool to address the challenges in large scale complex network analysis problems (see [2]). To demonstrate the effectiveness of the proposed hybrid system, we solve a challenge problem using three different platforms: a MapReduce cluster, a highly multithreaded system, and the hybrid combination of the two. Our challenge problem first extracts a subgraph of interest from a larger graph to capture the challenge in processing large volumes of data, and finds single-pair shortest paths in the extracted subgraph to capture the challenge in traversing irregular graphs. We analyze the match between the problem and the three different platforms via algorithm and system level analyses and also by experiments. The MapReduce cluster and the highly multithreaded system reveal limitations in efficiently solving our problem, whereas the hybrid system exploits the strengths of the two in a synergistic way and solves the problem at hand.

A MapReduce Cluster

The MapReduce programming model lowers the programming complexity in using a cluster, but this model may not work well for every workload. Here, we analyze its match to our problem.

MapReduce computation often requires multiple iterations of the map, shuffle, sort, and reduce phases. We can use a directed acyclic graph (DAG) to find the minimum number of iterations to solve a problem. Assume the vertices in each level are the partitioned data chunks for the corresponding MapReduce iteration, and there is a directed edge between two vertices if the target vertex is dependent on the source vertex (see Figure 1). Then, the depth of the DAG is equal to the minimum number of required MapReduce iterations.

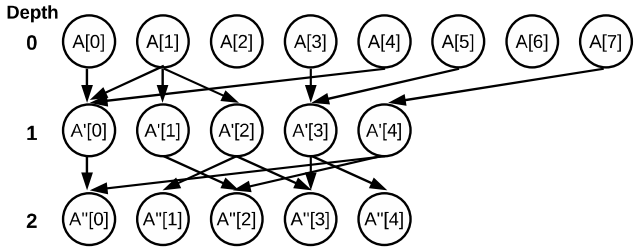

Figure 1: A directed acyclic graph (DAG) for MapReduce.

If we know the minimum number of MapReduce iterations to solve a problem and the amount of work in each iteration, we can compute the work complexity. The following equation captures this work complexity.

$$\sum_{i=1}^{k} (O(n_i + n_i f_i + n_i f_i r_i) + p_r Sort(n_i f_i / p_r))$$

k: *a number of MapReduce iterations*

n_i: *the input data* size *for the* i_{th} *itheration*

f_i: $\frac{\textit{the output data size}}{\textit{the input data size}}$ *for the* i_{th} *map phase*

r_i: $\frac{\textit{the output data size}}{\textit{the input data size}}$ *for the* i_{th} *reduce phase*

p_r: *a number of reducers*

Under our optimality criterion, a MapReduce algorithm is efficient if its work complexity is equal to that of the best sequential algorithm. A MapReduce cluster can extract a subgraph using an efficient algorithm but our analysis reveals that finding shortest paths requires significantly more work than the best sequential algorithm.

If the shuffle phase takes less than the map phase, the shuffle phase does not affect the overall execution time as it can be overlapped with the map phase. This is especially true when f_i is small as this reduces the communication volume, and is the case for the subgraph extraction. However, if f_i is large, the interconnection network becomes a bottleneck especially for large clusters with the sublinearly scaling bisection bandwidth. Finding shortest paths falls into this case. Disk I/O overhead is unavoidable if the input data overflows the main memory capacity or we need non-volatile storage. An extracted subgraph, however, has a higher chance to fit into the main memory using the sparse representation. The currently available MapReduce runtime systems (Google MapReduce and Hadoop) use

disks to save even temporary data and disk I/O overhead becomes significant in finding shortest paths.

A Highly Multithreaded System

A highly multithreaded system, with the shared memory programming model, supports random memory accesses with an effective latency hiding mechanism. These systems often have a relatively high bisection bandwidth for the system size to support a large number of irregular accesses as well. However, linearly scaling the bisection bandwidth increases the system cost superlinearly, and this limits the size of many highly multithreaded systems. This type of system is efficient for finding shortest paths in the extracted subgraph, but the limitation in system size becomes substantial in storing a large graph and extracting a subgraph from the large graph.

The Hybrid Combination

The following figure illustrates the proposed hybrid system.

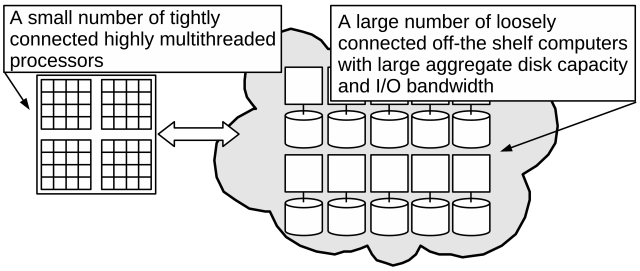

Figure 2: The proposed hybrid system.

The hybrid system connects a MapReduce cluster and a highly multithreaded system to effectively address the distinct computational challenges in large scale complex network analysis. Switching from one system to another to address the distinct computational requirements can reduce the execution time in each step but requires additional data transfer. The hybrid system becomes effective if switching from one to another saves more time than the data transfer overhead. The following equation captures this point.

$$T_{hybrid} = \sum_{i=1}^{k} min(T_{i,MapReduce} + \Delta, T_{i,hmt} + \Delta)$$

k: *a number of computational steps*

$T_{i,MapReduce}$ and $T_{i,hmt}$: *the time complexities of the* i_{th} *step on a MapReduce cluster* and *a highly multithreaded system, respectively.*

Δ: $n_i / BW_{inter} \times \delta(i-1, i)$

n_i: *the input data* size *for the* i_{th} *step*

BW_{inter}: *the interconnection network banwidth*

$\delta(i-1, i)$: 0 *if selected platforms for the* $i-1_{th}$ and i_{th} *steps are same.* 1, *if different.*

In our problem, the size of an extracted graph is much smaller than the input data size. This reduces the overhead in loading the subgraph from a MapReduce cluster to a highly multithreaded system. A MapReduce cluster requires significantly more work to find shortest paths than a highly multithreaded system and also incurs disk I/O overhead. Extracting a subgraph using a MapReduce cluster and finding shortest paths using a highly multithreaded system in the hybrid setting becomes effective as a result.

Experimental Results

We use an R-MAT graph with 4.29 billion vertices and and 275 billion edges (7.4 TB in text) as an input graph and extract a subgraph that covers 10% of the vertices in the input graph. Then, we find single-pair shortest paths in the extracted subgraph up to 30 pairs. Our MapReduce cluster has 4 nodes and 4 dual core AMD Opteron processors in each node and has 96 1 TB disks. Our highly multithreaded system has a single socket Sun Microsystems UltraSparc T2 processor and 32 GB main memory. The highly multithreaded system fails to solve the problem as the input data overflows its capacity. Table 1 summarizes the comparison between the MapReduce cluster and the hybrid system.

Table 1: Execution time to solve our problem.

	MapReduce	Hybrid
subgraph extraction	23.9 hours	23.9 hours
memory loading	-	0.832 hours
finding shortest paths (for 30 pairs)	103 hours	2.6 seconds

The hybrid systems outperforms the MapReduce cluster by far in the experiments. In particular, once the subgraph is extracted and loaded into the main memory, the hybrid system analyzes the subgraph five orders of magnitude faster than the MapReduce cluster. Using a larger MapReduce cluster and a faster interconnection technology may significantly reduce the graph extraction time and the memory loading time, respectively. However, reducing the time to find shortest paths is much more challenging due to its high communication requirements, and this further highlights the effectiveness of the proposed hybrid system in large scale complex network analysis.

References

[1] R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," Nature, 406:378-382, 2000.

[2] S. Kang and D. A. Bader, "Large Scale Complex Network Analysis Using the Hybrid Combination of a MapReduce Cluster and a Highly Multithreaded System," Workshop on Multithreaded Architectures and Applications (MTAAP), Atlanta, GA, April, 2010.