**Design Pattern for Efficient Graph Algorithm in MapReduce**

Graphs are implemented in important contexts, including ranking search, module detection of protein-protein interaction networks, and privacy analysis of social networks. Graphs are difficult to analyze because of their large structure. MapReduce provides an enabling technology for graph processing. MapReduce provides an abstraction for programmer defined “mappers" and “reducers", that both operate in parallel on key-value pairs. MapReduce supports use of combiners and partitioners which helps in graph processing.

PageRank is an exemplar of the class of graph algorithms. The power method as formulated requires only that the local topology and the uniform damping factor are considered at each step, making it especially well-suited to parallel computing. MapReduce uses key-value pairs comprising the graph will be divided into blocks and spread across the local disks of nodes in the cluster. The shuffle and sort phase of MapReduce can be exploited to propagate information between vertices using a form of distributed message passing. Combiners in MapReduce are response for local aggregation.

In mapper combining has two disadvantages, firstly combiner semantics is underspecified in MapReduce and secondly combiners reduce the amount of intermediate data shuffled across the network. For graph processing it is highly advantageous for adjacent vertices to be stored in the same block, so that any

messages passing between them can be processed in memory.

In an experiment including 25 terabyte data with Hadoop 0.20.0 with 5 iterations of PageRank with different sets of optimization enabled. With Basic HashPartitioner runtime in case of Unoptimized data is little less than 1500 seconds. Whereas the Schimmy HashPartitioner for Basic HashPartitioner is less than 1250 seconds.

Schimmy is useful because it obviates the need to reshuffle the graph structure at every iteration. Range partitioning requires only that vertices within a cluster. The future work is to improve these designed design pattern. It is difficult to implement without compromising the robustness of the system to hardware failures. MapReduce has emerged as an enabling technology for analyzing large datasets, because its generalized distributed framework facilitates many types of large-scale analysis with fewer demands on the developer relative to other paradigms.