This paper mainly deals with the implementation of Page Rank algorithm by three methods.

**Message Passing:**

The canonical approach is to map over the key-value pairs comprising the graph. at every vertex using the local graph structure, vertex metadata, and additional vertex state contained in the serialized graph vertices. The results of the computation are arbitrary messages to be passed to each vertex’s neighbors. This is accomplished by having mappers emit intermediate key-value pairs where the key is the destination vertex id and the value is the message. In the reducer, all messages that have the same key (i.e., same destination vertex id) arrive together, and another computation is performed, which corresponds. The mapper must also emit the vertex structure (i.e., the input value) with the vertex id as the key. This passes the vertex structure to the reduce phase, where it is reunited with messages destined for that vertex—so that the reducer can update the vertex’s internal state and write out the revised graph to disk. Without this step, there would be no way to perform multiple iterations, since we would have lost the graph structure. Thus, there are two distinct data flows in the basic implementation of graph algorithms in MapReduce: one corresponding to the flow of messages from source to destination vertices along graph edges, and the other corresponding to the shuffling of the graph structure itself. It does not handle damping factor and tangling nodes.

**Local Aggregation:**

The mappers perform a simple division to the PageRank mass, and the reducers sum incoming PageRank contributions. Therefore, the algorithm running time is dominated by shuffling large amounts of data across the network between the map and reduce stages of processing: both messages passed along graph edges and the graph structure itself. Because of this, any reductions in the amount of data shuffled across the network increases the speed of a MapReduce algorithm. Combiners in MapReduce are responsible for performing local aggregation (i.e., a partial reduce on map output), which reduces the amount of data that must be shuffled across the network. Clearly, they are only effective if there are multiple key-value pairs with the same key computed on the same machine that can be aggregated. In practice, combiners often yield dramatic reductions in algorithm running time due to decreased network traffic. If the combiners encounters messages destined for the same vertex, it sums up those partial PageRank values and emits the aggregate, while the graph structure is simply “passed along” to the reducer. For PageRank, combiners are especially effective for reducing the number of messages passed to vertices with high in-degrees. This has the additional effect of reducing the skew in the running time of reducers. PageRank is typically run on graphs whose vertex in-degrees follow power law distributions (e.g., the web graph): since reducer computations are proportional to the vertex’s in-degree (i.e., number of incoming messages), some vertices take much longer to process than others.