Many computation problems involve graphs. In the case of Google these graphs consist of trillion of node. Solving them were at most importance for Google but their existing framework Map Reduce, though could solve such problem, it often led to sub-optimal performance and usability issues. Many graphs problems involved message passing between nodes, a feature not supported by Map reduce and were often iterative. This forced Google to come up with a new framework, inspired by Valiant’s Bulk Synchronous Model purely for solving graph problems efficiently.

1. What is superstep in the Pregel graph processing model?

Ans:

In Pregel, computation consists of a sequence of iterations, called supersteps., where each superstep was divided by a ‘barrier’. In each superstep, a user-defined function is invoked on each vertex in parallel. The function specifies behavior at a single vertex V and a single superstep S. It can read messages sent to V in superstep S−1, send messages to other vertices that will be received at superstep S+1, and modify the state of V and its outgoing edges. Messages are typically sent along outgoing edges, but a message may be sent to any vertex whose identifier is known.

2. In the single source shortest path problem what computation is involved in a superstep?
In the case of single source shortest path problem, in each superstep, each vertex first receives, as messages from its neighbours, updated potential minimum distances from the source vertex. If the minimum of these updates is less than the value currently associated with the vertex, then this vertex updates its value and sends out potential updates to its neighbours, consisting of the weight of each outgoing edge added to the newly found minimum distance.

3. What does synchronicity in the Pregel’s execution refer to? What benefits can it bring?

Ans:

The synchronicity in Pregel’s execution refer to how computation is synchronous between supersteps. As mention in question 1, there exist a barrier between each superstep. The barrier only allow movement to the next step after all vertices have receive all their messages forcing the ones already completed to wait, aka a vertex is allowed to move to step $S + 1$ from $S$ if everyone else is also moving. Making the computation synchronous even though message can be received out of order.

This synchronicity allows pregel to achieve fault tolerance by check point the values on to a persistent memory device at the start of some fixed interval of steps. Due to interdependency of vertices in pregel, the fault tolerance model followed by map-reduce is infeasible here. For instance, if the above synchronicity wasn’t maintained different vertices could be at different steps. When a failure occurs, it will be impossible to recompute the graph accurately, since each value check pointed belongs to a different step.
4. How is a Pregel program terminated (completing its execution)?

Ans:

![Vertex State Machine](image)

**Figure 1: Vertex State Machine**

In pregel vertices have to states two state an active and inactive state. When computing a vertex is in the active state and move to the inactive state based on some user defined condition. When a vertex moves to the inactive it votes to halt. The votes are tallied by the “master”. In a given step if master receive ‘vote to halt’ from all the vertices in the graph the program terminates. Simply put in order for a program to terminate all vertices within the graph must vote to halt.

5. Use Figure 2 to illustrate a Pregel program’s execution.

Ans:

![Maximum Value Example](image)

**Figure 2: Maximum Value Example. Dotted lines are messages. Shaded vertices have voted to halt.**
Problem shown in the figure is that of finding the maximum value within the graph. To avoid confusion let us refer to vertices with values currently 3, 6, 2 and 1 as A, B, C and D respectively.

In step 0, all vertices send their value to their outgoing neighbours and receives the values of their incoming neighbours. For instance, B sends 6 to A and D and receives 3 and 2 from A and C respectively.

In step 1, each vertex computes the maximum value by comparing it own value with the ones it received in the previous step and assign the maximum value as its own value. If no change happens i.e., the vertex’s value is the maximum value then the vertex votes to halt. Here A and D change to 6 while B and C don’t change so they vote to halt.

In step 2, only C changes to 6, all the rest vote to halt.

And finally, in step 3, no value is changed all are the maximum of 6, so all vote to halt and thus terminating the program.

REFERENCES

[1] Pregel: A System for Large-Scale Graph Processing, Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski, Google, Inc.