Design

In our Sava, we have 7 major components: a distributed graph processing system called Sava, 2 Sava applications(in our case, we implemented pagerank and single source shortest path), udp-based group membership and failure detection algorithms, tcp-based remote procedure call for distributed grep and reliable message transfer, SDFS manager to handle distributed file system logic and finally a CLI component to interact with user.

Sava - a distributed graph processing system

Our design is similar to the Pregel system we learned in class which uses Gather-Apply-Scatter model. The system first load and partition the graph to each worker. The master then ask each worker to run an iteration depending on the application. Each worker first gather and process messages from other nodes, run an iteration to compute for each worker and finally scatter the messages to other workers which contains vertices that need the information. When all workers finished the iteration, the master instruct whether to proceed to next iteration. At the end of each job, master gather back the partial result from each worker and aggregate them to output.

Graph partition

We used BFS to partition the graph so that we can beat GraphX in computing Single Source Shortest Path. More specifically, we use BFS to do a graph traversal and record the order that we first see the node. And then distribute equal amount of nodes into workers based on this order.

Fault tolerance

Our system supports both master failure and workers failure. In the case of master failure, we'll wake up standby master. The standby master will continue to monitor and instruct the workers to proceed from latest iteration. In the case of the worker failure, the master will restart the job using the remaining workers. The failure recovery process is transparent to the user.

Sava application

We've implemented PageRank and Single Source Shortest Path. The sava application implements the Sava framework api such as process vertices and calculate outgoing messages.

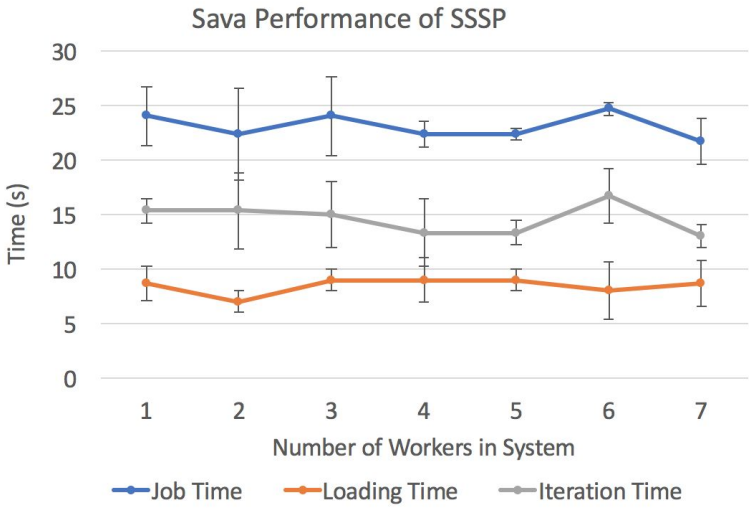
Sava Performance

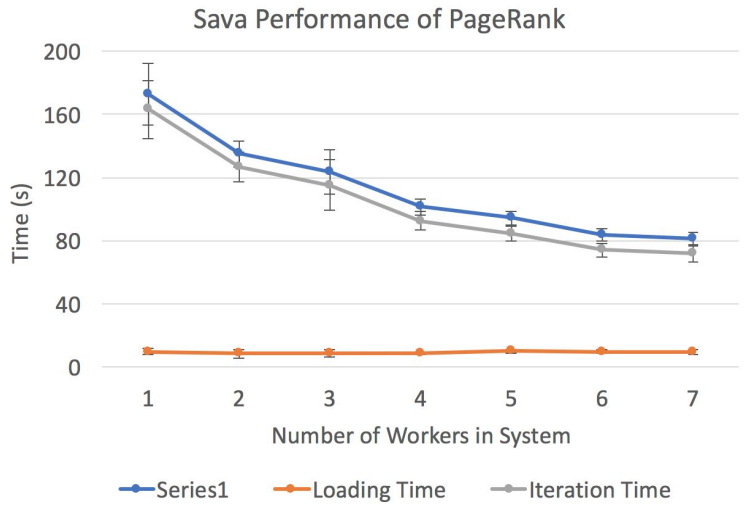
The following ideas can be summarized from the plots below:

(1) The Loading Time is about 8 seconds for both SSSP and PageRank. This is as expected, since the same graph file need to be loaded into memory and partitioned.

(2) The Job Time is mainly governed by the Iteration Time, thus they share the same trend. SSSP has much smaller Job Time and Iteration Time than PageRank. Due to our partition method, the packets passed over network is minimized for SSSP case. Therefore, the better performance of SSSP is as expected.

(3) The number of workers in the system does not affect Job Time significantly for SSSP. However, the Job Time reduces quickly with the increase of the number of workers in the case of PageRank.





Comparison with GraphX

The following ideas can be summarized from the plots below:

(1) Sava beats GraphX performance in SSSP case by about 50%, as expected. This is because our BFS partition method is optimized for SSSP problem.

(2) Sava performance increases with the increase of number of workers in the system. The Job Time decrease quickly with more workers and is getting close to GraphX performance.

(3) It is noticed that GraphX has better performance in PageRank case, because BFS partition is not optimized in this case. There may be many vertices in 1 worker that need to send their contribution to their neighboring vertices, thus creating large message (packet) size.