

Tianyu Chen (tchen72), Chris Benson (cebenso2)  
CS 425 Indranil Gupta  
September 17, 2017

### MP3 Report

**Language:** Golang

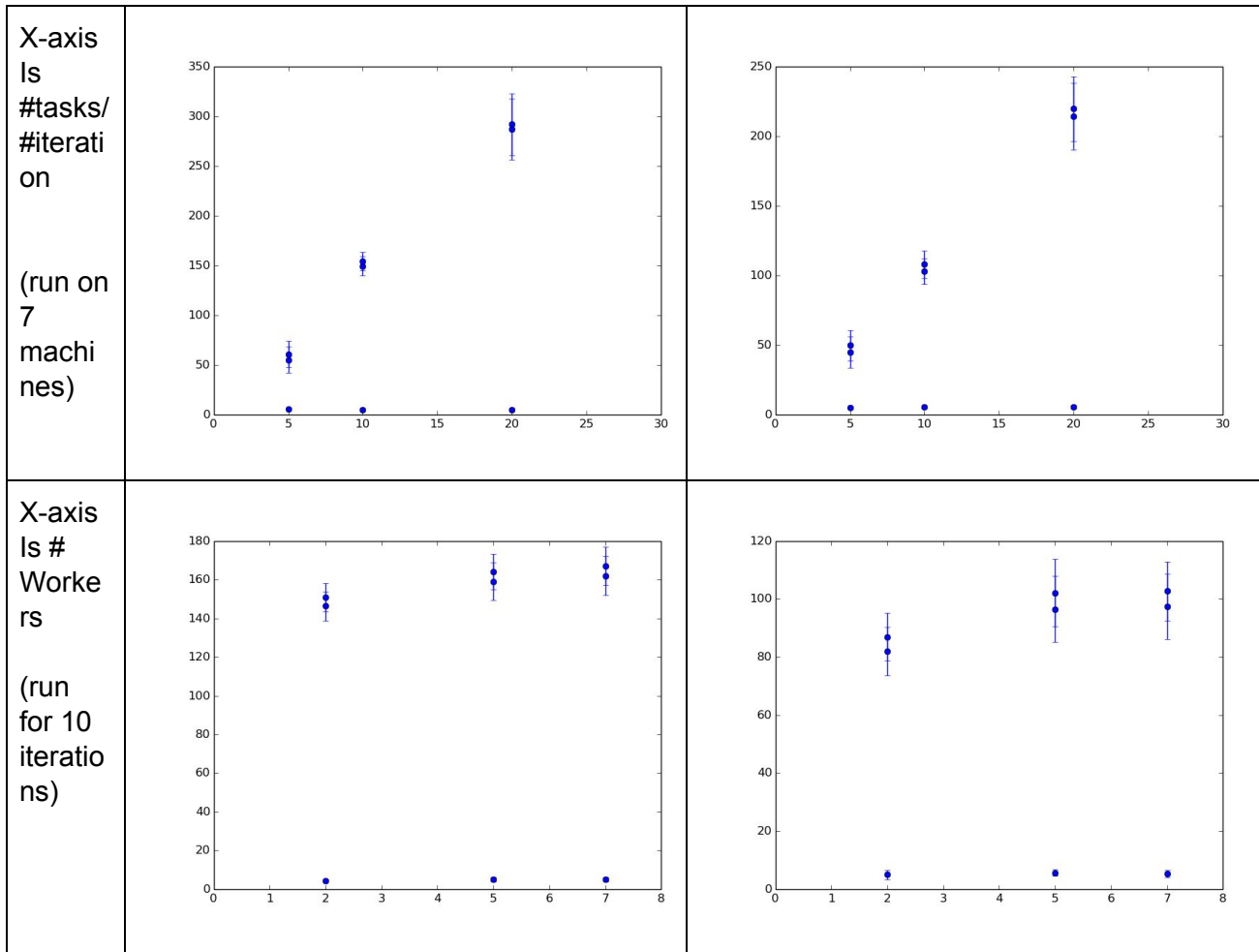
**Design:**

- Algorithm used: Basically our implementation is similar to pregel, with slightly modifications. To start a SAVA program, the client sends an RPC call to the master(default to be the 1st in membership list) with the exe file and the Inputgraph, and the master sends the exe file and the Inputgraph to all the workers, and then tell the workers their portion of the graph and go processing it. The worker then iterates through the nodes in its portion, calling exe file to each node, storing the results and send the results back to master after finishing its portion. After the master received response from every single worker, the master does a reduce function that output the messages for every node, then send the (nodeid,messages) array to its worker(the one who has the node in its portion of graph).
- Failure Handling: If a client or multiple client failed, when the master detects the failure, it restarts the SAVA program with the remaining workers. If a master failed, the backup master(the 2ed on membership list) will detects the failure, and the backup master will tell the clients the Superstep that was running and continue SAVA program(In Every iteration, the master sends superStep number the the backup master), the client will pick up its current outputs that was stored in global, and send those output back directly. Then the backup master will do the reduce as before and continue the supersteps.

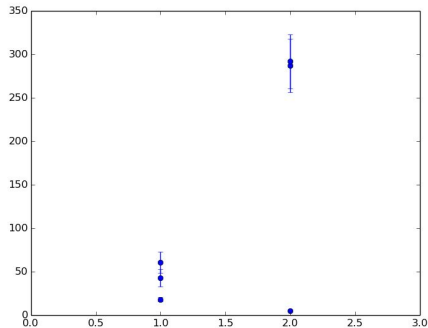
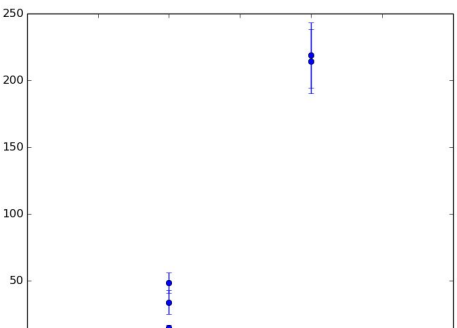
**Measurement:**

Vertical axis is seconds

	Page Rank	SSSP
--	-----------	------



For both PageRank and SSSH algorithm, we failed to beat the SparkX program because of our implementation (Sending everything back to master and do reduce there). For the trends, as number of iterations increase, the time it takes increase linearly also. However, when number of workers increase, the time it takes also increase slightly, which is caused by our implementation that master need to wait for every single one, and to do the reduce in the end.

	PG	SSSP																								
Spark run Time is at 1																										
Sava run time is at 2																										
X axis has no meaning otherwise	 <table><caption>PG Data Points</caption><tr><th>X</th><th>Y</th><th>Y_min</th><th>Y_max</th></tr><tr><td>1.0</td><td>50</td><td>40</td><td>60</td></tr><tr><td>2.0</td><td>280</td><td>260</td><td>300</td></tr></table>	X	Y	Y_min	Y_max	1.0	50	40	60	2.0	280	260	300	 <table><caption>SSSP Data Points</caption><tr><th>X</th><th>Y</th><th>Y_min</th><th>Y_max</th></tr><tr><td>1.0</td><td>40</td><td>30</td><td>50</td></tr><tr><td>2.0</td><td>210</td><td>190</td><td>230</td></tr></table>	X	Y	Y_min	Y_max	1.0	40	30	50	2.0	210	190	230
X	Y	Y_min	Y_max																							
1.0	50	40	60																							
2.0	280	260	300																							
X	Y	Y_min	Y_max																							
1.0	40	30	50																							
2.0	210	190	230																							

We were able to load the data slightly faster but overall run time was much slower.