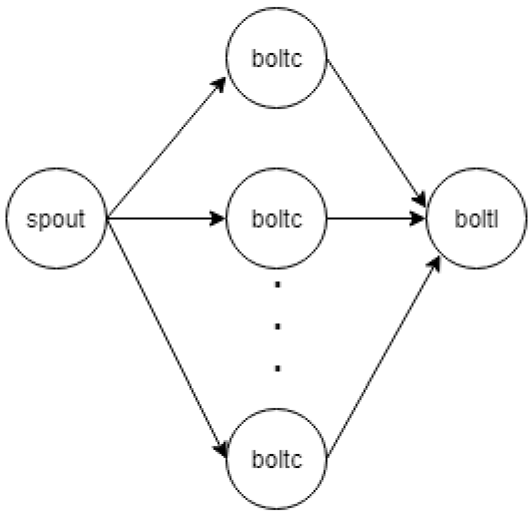
CS425, Distributed Systems: Fall 2018 Machine Programming Crane Report

Hang Yuan(hyuan13) Yihan Zhang(yzhng249)

Architecture

In this assignment, we have a Nimbus and the rest are worker nodes. When Nimbus starts and all worker nodes join the group, a client(daemon process of worker node) will send topology to Nimbus. Nimbus then assigns roles of spout and bolt to all worker nodes and worker nodes start the streaming process.

Our topology is very simple. Given the number of working nodes present, we assign the spout task to one worker node, and the rest nodes are bolts. Among all bolts, only one bolt is the bolt that collects result, called boltl(bolt leave) and others are boltc.



Failure Detection

When a machine fails, the failure detector will detects the failure and reports it to Nimbus, who is also the introducer in failure detector. Nimbus then distributes the id of the failure machine to all other worker nodes. Once the worker node receives the failure message, it quits the current running spout/bolts process and restarts. In this way we can guarantee that the output of our stream processing system is always correct.

Applications

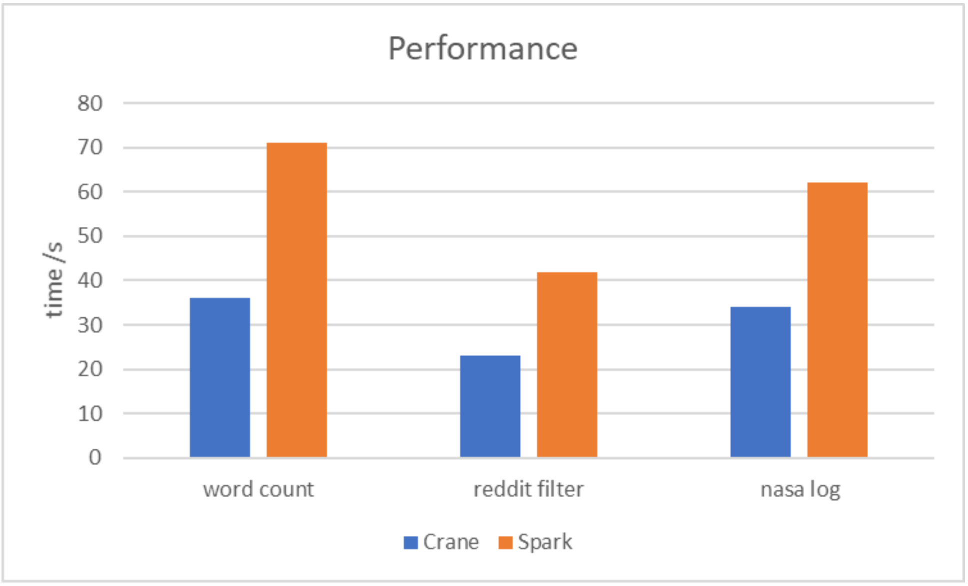
We have three applications. In all applications, the spout keeps an “index” indicating who is the next boltc to send data, reads dataset line by line and sent it to the corresponding boltc indicated by “index”.

The first is counting word frequency. The dataset https://snap.stanford.edu/data/bigdata/memetracker9/ contains sentences of news articles from various websites and the date, and we truncate it to 30 MB. So the spout will read file line by line, make a map with keys line number and line, marshal the data to JSON and pass it to bolt. The bolt then filter out all lines that are URL addresses and dates and pass lines to boltl. Botl then counts word frequencies and writes final result to a file.

The second application is figuring out the top 50 reddit users who have the most number of posts which have a positive score(number of upvotes – number of downvotes). The dataset we use is from http://snap.stanford.edu/data/web-Reddit.html which is a collection of 132,308 reddit.com submissions. We first filter out all posts with a positive score and then count number of posts of each reddit user. Finally, we sort the number of posts grouped by user and list the top 50 users with most posts. The dataset is in CSV format. So the spout will read CSV file line by line, make a map with keys rawtime, title, total_votes, reddit_id, score, username, marshal the data to JSON and pass it to bolt.

In the third application, we count for each host how many websites they have requested(GET) in a month and what those websites are. The dataset is traces of http requests to NASA Kennedy Space Center server in Florida in July 1995. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html. The spout will read log file line by line, make a map with keys host, request, url, status, marshal the data to JSON and pass it to bolts.

Performance Comparison on Spark & Crane



We start Apache Spark on 5 VMs and Crane on the other 5 VMs to test the performance of our application word count, reddit filter and nasa log. For each application, time consuming of Crane is smaller than Apache Spark, which satisfies our expectation. Because compared with Apache Spark, our Crane's implementations are much simpler.