CS425 MP4 report

Crane Stream Processing System

Ruian Pan(ruianp2), Carl Zhang(hz5)

Design:

We designed the Crane in a similar fashion with Storm in that it uses spouts, bolts, tuples and sinks. The entire cluster consists of 1 master, 1 back-up master, 1 client and 1-7 workers.

In our design, a complete application run includes the following steps: the client holds all the files needed to perform a job, and once the command of performing the job is entered, it sends all the files to the master. The master, upon receiving all files from client, schedule all the bolts to all the available worker and all the spouts, and send each one of them the required C++ program file and database file and information including the connections to be established. The crane worker, once received all the above information, create a bolt process that meets the above requirements and continue listen from master further instructions. The bolt process will then establish all the connections and for every incoming tuple, it runs the client program that deals with the tuple, get the output and sends to other tuples. Once a bolt receives a string indicating the end of stream, before it terminates, it sends an ack message back to master, who, once received as many ack as the number of bolts, sends an ack to client. For all the sinks, their results will be stored onto the SDFS. Once the client receives the message, the application is officially completed, and the result is available on SDFS for client to fetch.

In case of master failure, all the bolts will terminate upon detection of the failure, and the back-up master, once detected the failure, will kick in and rerun the entire job. In case of up to 2 worker failures, all the bolts will terminate upon detection of the failure, and the master, once detected the failure, will reschedule and rerun the entire job. In both cases, the client will not notice any difference and the correct result will always be returned.

Application file format:

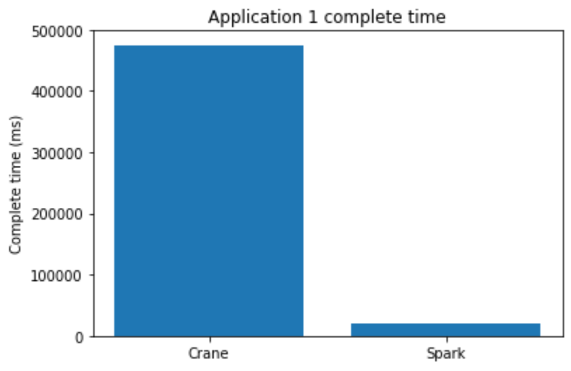
*.topo: A file that stores the topology of the application in the form of all edges
*.ds: Dataset file used for spout
*.db: Static database used for join operation
*.cpp: Bolt program

Command format on client:

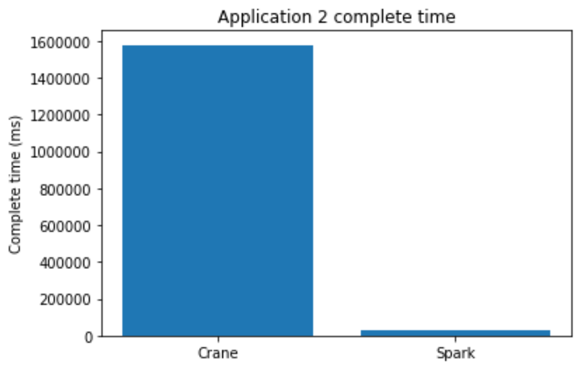
job jobname: Submit a job to Crane of the name "jobname", where all relevant files are store in the directory /jobname.
get sdfsname localname: get the result from SDFS of the name sdfsname to a local file with name localname.

Measurements (complete time length) and Plots:

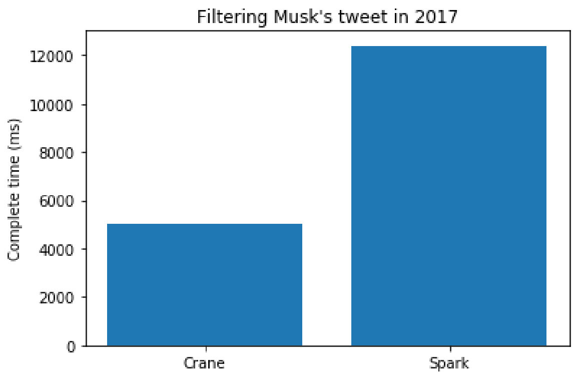
Application 1(size: 273 KB, tuple: 20000*13): transforming an Ethernet packet traces time(1st column) from seconds to milliseconds, then filtering packets(2nd column) by size < 140 MB and finally join the result with another static database(13 tuples).
Source(truncated): http://ita.ee.lbl.gov/html/contrib/BC.html



Application 2(size: 19,532 KB, tuple: 1000000): filtering an Ethernet packet traces by packets(2nd column) with size < 140MB first, then transforming and outputting all time(1st column) in milliseconds (originally in milliseconds)
Source: http://ita.ee.lbl.gov/html/contrib/BC.html

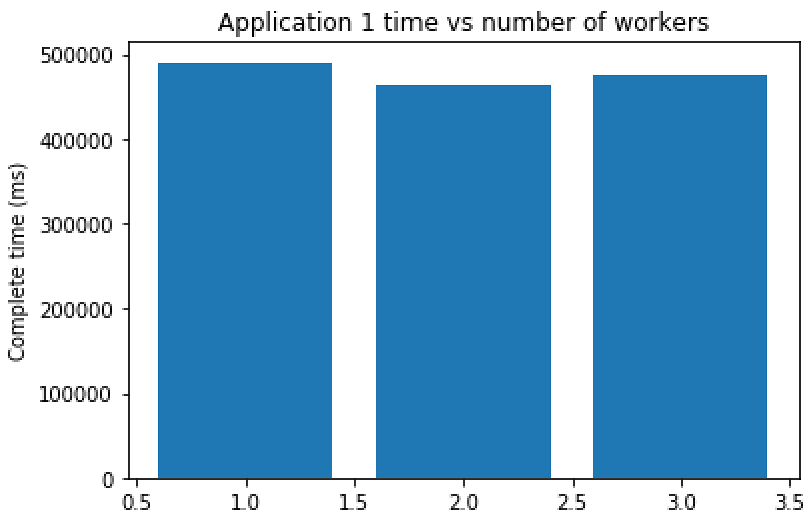


Application 3(size: 393 KB, tuple: 2819): filtering Elon Musk's tweets in 2017 from 2010-06-04 to 2017-04-05 and transform to output the content.
Source: https://data.world/adamhelsinger/elon-musk-tweets-until-4-6-17



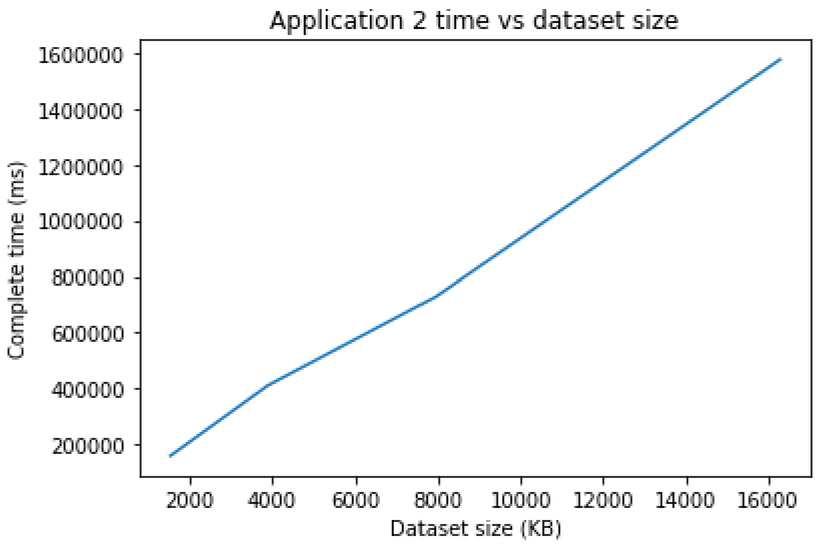
Analysis: Compared to Spark, Crane has better performance in case of small dataset and flat dataset where the size of each tuple is large and the number of tuples is small. That's because Crane takes less time to do the setup and can start working on applications earlier. In cases where the dataset is large or where the number of tuples is large, Spark does a significant better job. The optimization in terms of application processing on Spark is way better than that of Crane.

Application 1 time vs number of workers:



Analysis: The runtime is generally not affected by the number of worker machines, because no matter the number of worker machines, Crane will generate the same number of bolt threads and process concurrently.

Application 2 time vs dataset size:



Analysis: The runtime of application 2 is proportional to the size of the data, which makes sense since for the same tuple format, bigger dataset means more tuples.