CS425, Distributed Systems: Fall 2015
Machine Programming 4 – Stream Processing

Released Date: Nov 10, 2015
Due Date (Hard Deadline): Sunday, Dec 6, 2015 (Code due at 11.59 PM)

CallMeIshmael Inc. (MP3) just got acquired by the fictitious company SpaceBook Inc. (because eventually, every company is acquired either by ABCs Inc. or SpaceBook Inc. or Bitter Inc., even though all are fictitious). SpaceBook Inc. loved your previous work, so they've hired you. Congratulations!

You must work in groups of two for this MP.

SpaceBook needs to fight off competition from their two biggest competitors: Bitter Inc. and ABCs Inc. So they've decided to build a real-time processing system that will be faster than Apache Storm and Heron (those are real systems, as we discussed in class).

This MP consists of two parts.

First, your task is to build a system called ***Crane*** that is faster than Apache Storm. Crane is similar to Apache Storm in that it uses tuples, spouts, bolts, sinks, and topologies. The only differences are that 1) Crane must be written from-scratch (i.e., only uses code from MP1-3 and not from Storm or Heron, though you can look at the code for those systems), and 2) Crane topologies are trees. Crane bolts must support the main types of functionalities supported by Storm bolts, namely: filter, transforms, join of a stream with a static database (database could just be a file). You're also welcome to implement some stateful bolts (but it's not mandatory).

Further, Crane must be fault-tolerant: with up to **two** simultaneous failures of machines, no tuples must be lost. Try to do this in a way that is more efficient than the mechanisms used in Storm.

Second, you need show that your Crane is faster than their Storm. You need to write **three** real applications using Crane, each processing a (separate) real dataset. Find a dataset from the web (at least several 10s of MB large, make it streaming, and the input stream running for at least 10 minutes), and stream it through your Crane topology. Build your Crane topology to do something useful, e.g., count the top trending topics from a Twitter feed. More information about where to look for datasets appears later below.

Compare your Crane system's performance (both without and with failures) against Storm. Make sure that you're comparing Crane and Storm in the same cluster on the same dataset and for the same topology. Draw plots to show that Crane is better. Can you show that Crane is significantly better? In particular, show at least **one** application/dataset for which Storm is better, and **one** application/dataset in which Crane is better. Based on these results, discuss when one should use Crane vs. when one should use Storm.

To measure performance, you could use either throughput, or latency, or both. Decide which one(s) are important depending on your application.

Use the code for MP1-3 in building the Crane system. Use MP1 for debugging and querying logs, MP2 to detect failures, and MP3 to store the results from the Crane topology (or input data if that's what you want to do).

As usual, don't overcomplicate your design. SpaceBook Inc. is watching, and if they find you've overcomplicated things, they will throw you into space (or throw the book at you, depending on how you look at it).

Make sure you design a reasonable UI (command line ok) to start and stop jobs, see streaming results (e.g., dashboard, etc.) These will be useful during the demo.

We also recommend (but don't require) writing tests for basic scheduling operations. In any case, the next section tests some of the workings of your implementation.

Datasets: Good places to look for datasets are the following (don't feel restricted by these):

- Stanford SNAP Repository: http://snap.stanford.edu/
- Web caching datasets: http://www.web-caching.com/traces-logs.html
- Amazon datasets: https://aws.amazon.com/datasets/

Machines: We will be using the CS VM Cluster machines. You will be using 7 VMs for the demo. The VMs do not have persistent storage, so you are required to use git to manage your code. To access git from the VMs, use the same instructions as MP1.

Demo: Demos are usually scheduled on the Monday right after the MP is due. The demos will be on the CS VM Cluster machines. You must use 7 VMs for your demo (details will be posted on Piazza closer to the demo date). Please make sure your code runs on the CS VM Cluster machines, especially if you've used your own machines/laptops to do most of your coding. Please make sure that any third party code you use is installable on CS VM Cluster. Further demo details and a signup sheet will be made available closer to the date.

Language: Choose your favorite language! We recommend C/C++/Java.

Report: Write a report of less than 3 pages (12 pt font, typed only - no handwritten reports please!). Briefly describe your design (including architecture and programming framework) and chosen applications. Be very clear and very comprehensive. Also show plots that compare Crane's performance to Storm, for each of the three applications. Think about what metrics you wish to plot, and what parameters you wish to vary. Make sure that you're comparing Crane and Storm in the same cluster on the same dataset and for the same topology. Discuss your plots, don't just put them on paper, i.e., discuss trends, and whether they are what you expect or not (why or why not). (Measurement numbers don't lie, but we need to make sense of them!)

Submission: There will be a demo of each group's project code. Submit your report (softcopy) as well as working code. Please include a README explaining how to compile and run your code. Submission instructions are similar to previous MPs.

When should I start? Start **now** on this MP. Each MP involves a significant amount of planning, design, and implementation/debugging/experimentation work. **Do not** leave all the work for the days before the deadline – there will be no extensions.

Evaluation Break-up: Demo [40%], Report (including design and plots) [40%], Code readability and comments [20%].

Academic Integrity: You cannot look at others' solutions, whether from this year or past years. We will run Moss to check for copying within and outside this class – first offense results in a zero grade on the MP, and second offense results in an F in the course. There are past examples of students penalized in both those ways, so just don't cheat. You can only discuss the MP spec and lecture concepts with the class students and forum, but not solutions, ideas, or code (if we see you posting code on the forum, that's a zero on the MP). SpaceBook Inc. is watching!

Happy Membership (from us and the fictitious SpaceBook Inc.)!