Michael Dunnegan Datadog Challenge Writeup

General Notes

* I developed and ran my code using IntelliJ, and that is probably how you should run the code
* I used Gradle as the build tool, as that’s what I am most comfortable with.
* I chose the versions of Spark and Spring that we use at my workplace, because I know they’re compatible
* I downloaded the blacklist and have it stored in the code repo. More on this later.
* The date convention of the wiki data files is preserved in my program. For example, the file referenced by URL <https://dumps.wikimedia.org/other/pageviews/2015/2015-05/pageviews-20150501-010000.gz> contains data for the hour ENDING at 1am on 20150501 (and starting that midnight). My output file names follow this convention.
* My program requires a startTime input, and optionally accepts an endTime input. Both of them are required to be in the Java LocalDateTime format. The program will run for all hours between the two given times, inclusive. When endTime is not included, it assumes the value of startTime. There is a restriction on input dates, due to a limitation with Wikipedia’s data. Nothing before May 2015 is available. Additionally, job runs cannot be done for times in the future. The following two bullet points are valid inputs:
  + - --startTime="2015-05-01T01:00:00" --endTime="2015-05-01T02:00:00"
    - --startTime="2015-05-01T01:00:00"
* The output of the job runs are stored in the code repo, under the output folder
* The downloaded gz file is deleted after the output file is written
* I have made a choice to display a maximum of 25 rows per subdomain. In other words, ties are broken in a nondeterministic way. When I apply the window function to each partition (the dataset is partitioned by subdomain), I select rows with Row Number <= 25, instead of Rank <= 25. Multiple rows can have the same Rank, but no two rows can have the same Row Number.
  + - I would talk about this choice with other people before making it.
* Performance could probably be improved, if necessary. Each date/hour job takes a little over a minute

Follow-up questions

* What additional things would you want to operate this application in a production setting?

I would set up a real spark cluster/EMR instance in AWS. For my current code, a small number of executors works fine. If we were running this job every hour, then a small number of executors would also be fine. If our workload was such that we have to run for very large ranges, I would change my code, and would like a larger spark cluster. I would change the code so that multiple date/hour jobs could run in parallel. I would use an executor service for this. Ideally, each date/hour job would run on different spark executors, hence the desire for more executors.

Additionally, I’d store the blacklist and the results on S3 or HDFS, instead of locally, so a change to the blacklist wouldn’t require a commit/deploy

* What might change about your solution if this application needed to run automatically for each hour of the day?

I’d want to set up a logging strategy, so we can see the stack traces for failed jobs. Additionally, we’d need some way of scheduling the jobs to run, but that’s outside the application itself. Another thing to consider might be whether or not we choose to keep the cluster alive between runs. I haven’t used AWS, but an EMR

* How would you test this application?

I’d write an automated test, and have it run against a dataset that handles all edge cases I can think of.

* How you’d improve on this application design?

Since this application is relatively simple, I’ve chosen to keep everything in a single class. If this application grew much bigger, I’d separate out responsibilities into different classes. I could have a separate class for input validation, one for the wiki file download, one to load/process/store the data with spark, and one for unique utilities, like the SSL check disabler.

**I did enjoy working on this project! The idea of working on more complex challenges at petabyte scale is extremely exciting to me, and I hope to come onsite and learn more shortly. Datadog remains my #1 choice over the other companies I am currently interviewing at.**