Comparing the performance of Graph Analysis algorithms using Apache Flink and Apache Spark graph processing libraries

Authors

Mohamed Gabr <mohgab@kth.se>

Óttar Guðmundsson <ottarg@kth.se>

Allocation of responsibilities

Mohamed Gabr is responsible for setting up the software packages and libraries needed to run the experiments. He will also write the algorithms in both languages and running them on a selected dataset. He is also responsible for presenting the project plan.

Óttar Guðmundsson is responsible for writing introduction, abstract, discussion and conclusion of the final report. He takes care of finding and selecting the data set that will be used by the algorithm and work on data representation by creating figures and tables. Finally he will present the final project.

Organization

The project will be carried out by a two person team that share interest and classes in distributed systems. Both have experience working with Spark so setting up the environment for both members will be done individually. When the program is ready it will run on both computers where results will be analyzed and finally, sections of the final report will be splitted depending on the divided responsibility.

Background

In recent years, big data graphs and relations between their data points has been growing at enormous speed [0]. The demand for specialized tools to analyze them has led to the development of graph processing libraries such as GraphX (Apache Spark) and Gelly (Apache Flink). Comparison between such libraries and their utilization on hardware has been researched [1], but choosing which one to pick when it comes to early development for projects that rely on complex algorithms performance requires further investigation.

Problem statement

When a graph is processed, one has to take into the account the context of the data and what info is going to be analyzed in order to choose the proper library to get the wanted outcomes. For an example, analyzing how a disease spread or looking for signs of former fraud detection can be found by processing graphs and how they are all connected [2]. On the other hand, if this analysis or detection has to happen in real time, the researcher should pick a designated library.

Nowadays, there is a myriad of APIs, all with different specifications for different purposes. Any researcher, or simply an enthusiast struggles to find the one that suits his/her needs and likely to proceed to testing all of them or choose one that is not the best for his/her analysis. This is obviously time consuming, the problem becomes more important when timelines and deadlines are tight. All the previous works [] focus on comparing the libraries in general, or between batch and stream processing but we couldn't find in previous works with a comparison of the performance for specific graph processing algorithms like Triangle Count.

Problem

Depending on the data and how it is processed, speed is of the essence if it is being analyzed in real time. In terms of complex algorithms, one library should be superior to the other one if the graphs are streamed or arrive in bulks for specific algorithms.

Which of the two libraries should be chosen for a specific graph processing problem in terms of runtime?

Hypothesis

GraphX (Apache Spark) should perform better in any of the tested algorithms when all of the data is already stored and available. However, Gelly (Apache Flink) should outperform GraphX as the data arrives in streams. We define the hypothesis confirmed based on the execution time of the algorithms, but we'll also acknowledge other supporting metrics such as CPU and memory if any slack time is available in our plan at given time.

Purpose

The purpose is to have benchmarks comparing the performance of the same algorithm/method in different graph processing libraries. Such algorithms are, as said in the background, important during graph analysis. Any researcher or user of these libraries can save time by knowing in advance which one to use for a specific task, depending on the context of the problem.

Goal(s)

A fair and complete comparison of the performance between the two libraries, GraphX in Spark and Gelly in Flink, ran on two different mainstream personal computers, showing the advantages and disadvantages in terms of runtime (and if time allows: memory and disk usage) of each in popular specific graph processing algorithms.

Tasks

Two personal computers will have the libraries installed on. The same IDE and programming language will be used on the two. Each algorithm will be ran on the two computers using each of the libraries. Data will be given as stream and as a batch. Comparison will be made between Graphx and Gelly for the two formats of the input data. Graphs that are going to be tested will be of different sizes with different complexities (more or less triangles, edges and vertices count…). The evaluation will be based on runtime (and if time allows: memory usage and disk usage). The same benchmarks, using the same tools will be done on the two computers.

Method

A full explanation of the method is not yet clear, but some of its aspects can already be stated at this early stage; The language that will be used for implementation is Scala. The two chosen computers considered as mainstream are the Retina, 13-inch, Early 2015 Macbook Pro and the xx IBM/Lenovo Thinkpad [3]. The benchmark techniques are yet to be defined. It is not yet clear if a pre-made benchmark tool will be used or if it will be designed from scratch for batch processing and stream processing. Some solutions like Yahoo Benchmark are open-source and made for testing Apache Flink and Spark and other frameworks. The complexity of the benchmarks and the variety of the metrics (runtime, disk usage…) will solely depend on the allocated time for this project.

Milestone chart (time schedule)

The project itself will start on 4th September and end on 30 September. A few key milestones are noted and seem doable in planned time.

7th September
Installation of environments should be done, being able to run basic tutorials of both Spark and Flink. Both libraries should also be installed and runnable. One or two graph datasets must be downloaded and imported into an environment project. It is done on the two computers. This milestone would be considered completed if the word count example code in the documentations of the APIs works.

14th September
Creating a function to process the graphs both as streams and batches. Developing and finalizing the graph processing algorithm(s) that we want to test on the dataset. Saving results of the metrics to benchmark to a file. This would be considered completed if there is an analysable results for the wanted metrics.

21st September
Analyzing our results by creating appropriate graphs, tables and figures to make sense of our data. We should be able to come up with the conclusion of our findings. Playing Devil's advocate to search for bias and flaws in our conclusion. This would be considered completed if there is a conclusion that either supports or refutes the hypothesis.

24th September
Starting the formal writing of the analytical results and discussion of our report. Note that introduction, background and methods used should be written along the development as stated in **Allocation of Responsibilities**. This would be

considered done if all the members in the project agree on passing the draft to peer review.

27th September
With the report's draft being completed, the slides should be created from what we have written to draw forward the most important takeaways from the report and the most informative figures. We must make sure that the slides represent key findings. This would be considered done if all the members in the project are satisfied with what is going to be presented.

30th September
Having three files ready in a shared drive called FINAL_REPORT_DRAFT.pdf, FINAL_SLIDES_DRAFT.pptx and project.zip. The project.zip file should include all code, data and a readme.txt file to explain how to reproduce the results that we based our conclusion on. This would be considered done if all the specified files are included in the .zip and that no member wants any further development in the project.

For every milestone, a collection of notes and information about what we performed and the outcomes will be written down for later reference. This will be useful when we start writing the report.

This should leave us with the initial draft of the report and our presentation at 1st of October, 2 days before the planned peer review on 3rd October. Thus, we will have extra two days to make some minor changes for the first review.

References

[0] Nick Ismail.11.Jan.2018. The growing graph database space in 2018. [ONLINE] Available at: https://www.information-age.com/growing-graph-database-space-2018-123470315/. [Accessed 3 September 2018].

[1] Perera, and Kamal. “Reproducible Experiments for Comparing Apache Flink and Apache Spark on Public Clouds.” *[1402.1128] Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*, 14 Oct. 2016, arxiv.org/abs/1610.04493. [Accessed 3 September 2018].

[2] Gorka Sadowksi & Philip Rathle .2017. Fraud Detection: Discovering Connections with Graph Databases [ONLINE]. Available at http://asiandatascience.com/wp-content/uploads/2018/01/Neo4j_WP-Fraud-Detection-with-Graph-Databases.pdf. [Accessed 3 September 2018].

[3] Athow, Desire. “Best Business Laptops 2018: Top Laptops for Work.” *TechRadar*, TechRadar The Source for Tech Buying Advice, 17 Aug. 2018, www.techradar.com/news/best-business-laptops. [Accessed 3 September 2018].