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>

Date of project report

Wednesday, September the 12th 2018

Aims

The main goal is to help future users and researchers in choosing their correct library when they face a graph processing problem. Another side/minor goals are resource savings related to the environment by using less energy. Besides the fact that the results will save time for future users, choosing the right algorithm also means using fewer resources. The focus will be on two libraries, GraphX (Apache Spark) and Gelly (Apache Flink). The choice is going to be based on popularity.

This leads to some questions that we will try to solve. Which of the two chosen libraries outperforms the other? Is there a clear winner for batch processing or stream processing?

Our belief is that GraphX should perform better in any of the tested algorithms when all of the data is already stored and available. However, Gelly should outperform GraphX as the data arrives in streams, this is our hypothesis that we expect to solve. We define it 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.

Background and rationale

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

Such algorithms are 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.

Theory/literature

Since both Spark and Flink tend to achieve similar goals with a similar purpose, a comparison between those two has been conducted. One of these researches compares Spark and Flink in terms of runtime, CPU and memory usage for the batch as well as stream processing, Flink clearly outperformed Spark in runtime. The CPU stress and the disk utilization test also favored Flink. [1]

However, another research didn't arrive at the same conclusions as for the first one. There is no clear winner according to the authors. For them, Spark is faster for bigger graphs while Flink is the choice when it comes to smaller graphs [2]. They believe that the reasons for such differences are the JVM memory management, the pipeline execution nature of Flink and the ease of optimization in the latter (done automatically) compared to Spark.

Furthermore, interesting researches have shown that Spark does outperform Flink in scalability with respect to some libraries. The machine learning library of Spark seems to have the upper hand against Flink [3], but that might be because of less focus on machine learning in Flink and Flink having fewer contributors in that area.

Note that a more complex research on analyzing streaming capabilities of this two software (plus another one called Storm) has been done [4] but solely to mimic real-world production scenarios, not the faster processing time of algorithms.

We can conclude from the previous articles and research papers that there is no clear answer to whether Spark performs better than Flink or not in terms of the execution time of graphs. Additionally, there is no clear conclusion for the same question on either batch or stream processing only. Therefore, we believe that this topic needs further investigation. Based on this literature, we are able to determine how to evaluate the results and get some insights on the parameters that can vary the results and benchmarks.

Research Methodology

Two mainstream personal computers [5] will have the libraries installed on. The same IDE and programming language will be used on the two. Each algorithm will run on both computers using each of the libraries, where data will be given as a stream and as a batch. A 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 [6] (more or fewer 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. Since Gelly-Stream does not have documentation yet, our methodology to test stream graph processing on Gelly would be to make our own graph processing code. The same goes for Spark. We are still not settled on the benchmark tools, some are already being tested like ScalaMeter. The graphs that are analyzed are downloaded from a free, open and public database that has different examples with varying complexities [7].

Expected Outcomes

A fair and complete visual comparison of the performance between GraphX in Spark and Gelly in Flink. Both will run 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. Finally, the results should hint at a further investigation of these two libraries that we hope to address in the discussion part of our research.

Milestones/schedule, budget

Milestones

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

5th September (Done)
Installation of environments is done, we are able to run basic tutorials of both Spark and Flink. Both libraries are installed and runnable. Three graph datasets are downloaded and imported into the project environment. It is done on the two computers. This milestone is considered completed because the word count example code in the documentation of the APIs is working.

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 analyzable result 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 the 1st of October, 2 days before the planned peer review on 3rd October. Thus, we will have an extra two days to make some minor changes for the first review.

Budget

This research requires no budget. Both libraries that are going to be benchmark are open source so no license fee is needed. Secondly, these libraries are going to be tested on two mainstream computers that the authors have already acquired.

Risks

Experience
One member of the research team has already worked with Scala in Spark, thus the team already has more experience when it comes to coding the research examples. This might lead to performance bias e.g. the experienced researcher might apply some performance tricks and tips in Scala that he does not know in Flink. Time spent writing the code might also affect the results in both libraries. Being too hasty in writing the Scala code might incorporate more mistakes that might either slow the research down or create bugs that have an unforeseen effect on the results.

Lack of Flink community and Gelly documentation
Spark has been around for a longer period of time than Flink, resulting in a bigger community and more code examples/tutorials [8]. This makes it harder to find support to develop the code needed for Flink since less open questions and discussions exist on forums. In that case, the research plan could be slowed down having the team missing important milestones.

Difficulties in benchmarking
There are many benchmarking tools, but integrating them with our code is a big task, especially that we are using APIs. Although, some benchmarking tool like the runtime can be easily done without pre-made tools.

Outline

Abstract
The abstract will be a brief summary in one or two paragraphs presenting what is in the written work. It should help the reader in quickly evaluating whether this research will be of interest or not.

Introduction
We will introduce our research and work in this paragraph. It will include our problem statement, research questions, and the hypothesis.

Background
A broader view of the field that is going to be investigated with respect to other or similar experiments that gives the reader a detailed view of the research field.

Research methodology
A detailed explanation of how we will carry out and conduct our experiments.

Results and Analysis
Assumption or a clear definition is drawn from the data that the project created. Observation of the most interesting data points that hopefully will align with the former hypothesis proposed.

Discussion
Further analysis and speculations towards the execution of our implementation and the results. Hopefully a hint at further development and research in the field to investigate other metrics or strengthen our conclusion

Conclusion
An answer to the statement presented in the introduction is presented here. It will also resume the entire work done during the research and the important milestones that led to the results.

References
The references that we've already found to support our arguments, plus other that might surface up in our Results or Discussion chapters.

References

[0]“The Growing Graph Database Space in 2018.” *Information Age*, 11 Jan. 2018, https://www.information-age.com/growing-graph-database-space-2018-123470315/.

[1]Perera, Shelan, et al. “Reproducible Experiments for Comparing Apache Flink and Apache Spark on Public Clouds.” *ArXiv:1610.04493 [Cs]*, Oct. 2016. *arXiv.org*, http://arxiv.org/abs/1610.04493.

[2]*Spark Versus Flink: Understanding Performance in Big Data Analytics Frameworks - IEEE Conference Publication*. https://ieeexplore.ieee.org/abstract/document/7776539/. Accessed 11 Sept. 2018.

[3]García-Gil, Diego, et al. “A Comparison on Scalability for Batch Big Data Processing on Apache Spark and Apache Flink.” *Big Data Analytics*, vol. 2, no. 1, Dec. 2017. *Crossref*, doi:10.1186/s41044-016-0020-2.

[4]*Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming - IEEE Conference Publication*. https://ieeexplore.ieee.org/abstract/document/7530084/. Accessed 11 Sept. 2018.

[5] *Laptops, Desire Athow 2018-07-30T08:26:00Z. ‘Best Business Laptops 2018: Top Laptops for Work’. TechRadar, https://www.techradar.com/news/best-business-laptops. Accessed 11 Sept. 2018.*

[6]*KONECT - The Koblenz Network Collection. http://konect.uni-koblenz.de/networks/. Accessed 11 Sept. 2018.*

[7]*ScalaMeter | ScalaMeter*. https://scalameter.github.io/. Accessed 11 Sept. 2018.

[8]“Google Trends.” *Google Trends*, https://trends.google.ie/trends/explore?date=all&q=%2Fg%2F11btzy2gtf,%2Fm%2F0ndhxqz. Accessed 11 Sept. 2018.