Prefetching in RocksDB/Apache Flink for streaming workloads

MASTER THESIS STUDENT
FREDRIK BJÖRKMAN
fbjorkma@kth.se

EXAMINER
PARIS CARBONE
parisc@kth.se

SUPERVISOR
SONIA-FLORINA HORCHIDAN
sfhor@kth.se

February 16, 2022

1 Preliminary thesis title

Prefetching in RocksDB/Apache Flink for streaming workloads

2 Keywords

Apache Flink, RocksDB, Datastreams, Prefetching

3 Background

This project will be carried out at KTH in collaboration with Boston University. Modern stream processors rely on embedded key-value stores to manage state that accumulates over long-running computations and exceeds the available memory size. One of these key-value stores is RocksDB [1] which is used by many systems, such as Apache Flink [2]. An important technique in reducing the I/O wait time for disc reads is the use of prefetching data into main memory before it is used. The challenge then arise to determine what data to prefetch, since the memory accesses might not have any obvious patterns. However some unique characteristics arises around streaming state access workloads. The state access for streaming workloads tends to have a high spacial locality [3].

In an article from 2020, the performance of two different state back-ends, RocksDB and FASTER [4], was evaluated [5]. The report showed that neither of them was superior to the other and they excelled in different areas. The article concluded that by implementing a workload-aware streaming state management there will be a significant boost in performance. This project will instead of focusing on adapting the behaviour of the application based on the workload, focus on trying to increase performance by utilizing prefetching.

The goal of this project is to leverage this result by designing and implementing a novel prefetching mechanism in RocksDB that is tailored for streaming workloads, that proactively populates the cache with data that has a high probability of being accessed in the near future. To do so, we will need to continuously identify and extract key neighbourhoods from the state access traces of streaming computations.

4 Research question and Method

4.1 Research question

By designing and implementing an effective prefetch mechanism in RocksDB, how will that affect the performance of state access on the streaming data?

4.2 Hypothesis

The expected outcome of this project is that, if an effective prefetch mechanism is in place, then this will have a positive effect on the performance.

4.3 Research method

This project will have a quantitative approach, where tests will be performed to measure the performance for state access on the streaming data and compare the performance with and without a prefetch mechanism tailored for streaming workloads.

4.4 Objectives and Tasks

The research question can be broken down into the following objectives:

- Having a working prefetching mechanism when accessing the stream data states from RocksDB.
- Benchmark the application with and without the prefetching algorithm.

Then following tasks needs to be completed in order to achieve the objectives:

- Implement a simple test program to monitor the communication with RocksDB
- Find a suitable place in the code to implement the prefetching algorithm.
- Try to sort the keys to minimise random access and see how the latency is affected.
- Try to buffer these keys and see how the latency is affected.
- Maintain statistics on the most accessed keys for the workload so we can do smart prefetching.
- Test prefetching algorithm and evaluate performance.

4.5 Ethics and Sustainability

This project aims to improve performance when accessing data from RocksDB and have very little to do with ethics and sustainability. One could argue that if an implementation would be successful and the amount of look-ups in RocksDB would decrease due to data being successfully prefteched, then the system could potentially draw marginally less power.

4.6 Limitations

This project will only look at RocksDB as the back-end database for Apache Flink. Apache Flink will also be the only stream-processing framework that this project will research. Only streaming data will be researched. There could be a use for a trained machine learning model to decide what data to preftech, but at this stage in the project it is still unclear if that will be implemented or not.

4.7 Risks

As this project will be a quantitative research project and what is to be evaluated is code written by the master thesis student, there is always a risk in implementation errors that can lead to inaccurate conclusions to be drawn. This will hopefully be avoided by a careful well thought through implementation and potential code reviews.

5 Evaluation and News Value

5.1 Evaluation

This project can be evaluated by creating a benchmark program that uses the same operations on the same data stream and comparing the current implementation with the implementation that contains the prefetch mechanism.

5.2 Expected scientific results

Currently there is no support for a prefetching mechanism in between Apache Flink and RocksDB and by performing this project (and creating a proof on concept) it may lead to if it is worth to implement this in the product to increase performance. The proof of concept and the research behind it might also map out the way this could be implemented in the product.

5.3 The work's innovation/news value

The ones interested in this work will most likely be the ones that are developing and using Apache Flink. If the developers would implement this feature it would hopefully lead to a boost in performance for the application.

This might also be of interest for those who look for similar solutions between a streaming framework and a back-end database.

6 Pre-study

This project, as said in Section 3, will be built upon previous work done by Vasiliki Kalavri and John Liagouris *In support of workload-aware streaming state management* [5]. The pre-study will originate from that work to get an understanding of the background to this project. After that there will be a lot of work to read documentation and source code for both Apache Flink and RocksDB. On top of that, information need to be acquired regarding how the implantation of the prefetching algorithm is going to take place. The literature study will need to be conducted in parallel with the implementation throughout a large part of this project as more knowledge about Apache Flink and RocksDB are acquired.

7 Conditions and Schedule

7.1 Resources

The project will be able to use an already build testbed built by Boston University for the performance testing on state access for streaming workloads. There are also available data sets for streaming data that can be used for testing of the performance. Both KTH and Boston University have expressed that they are able to assist with computational resources.

7.2 Milestone chart

This is the proposed timeline for the project:

Week	Tasks	Deadlines
6	Experimenting with Flink/RocksDB	
7	Experimenting with Flink/RocksDB. Fix and tailor thesis template. Dig into source code. Read reports and summarize.	Feb 15: Individual plan
8-10	Dig into source code. Read reports and summarize. Start writing an Introduction and Background.	Mar 11: Send first draft of intro and background
11-18	Dig into source code, implement, run experiments. Write method and results section based on experiments	Mar 25: α-draft May 3: Implementation done. May 6: Draft with method and results
19	Work on Discussion, Conclusion, Future work and Abstracts	May 13: β-draft
20	Work on report	May 20: Final draft
21-22	Prepare for presentation	Jun 3: Final presentation
23-24	Finalize report	Jun 17: Finished report

References

[1] D. B. et al, "RocksDB," *GitHub*, 12 2021, [Online; accessed 13. Dec. 2021]. [Online]. Available: https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

[2] "Apache Flink: What is Apache Flink? — Applications," 9 2021, [Online; accessed 14. Dec. 2021]. [Online]. Available: https://flink.apache.org/flink-applications.html

[3] S. Somogyi, T. Wenisch, A. Ailamaki, B. Falsafi, and A. Moshovos, "Spatial memory streaming," in *33rd International Symposium on Computer Architecture (ISCA'06)*, vol. 34, no. 2. IEEE Computer Society, 2006. doi: 10.1109/ISCA.2006.38 pp. 252–263.

[4] B. Chandramouli, G. Prasaad, D. Kossmann, J. Levandoski, J. Hunter, and M. Barnett, "Faster: A concurrent key-value store with in-place updates," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 275–290.

[5] V. Kalavri and J. Liagouris, "In support of workload-aware streaming state management," in *12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20)*. USENIX Association, 7 2020. [Online]. Available: https://www.usenix.org/conference/hotstorage20/presentation/kalavri

Acronyms

KTH KTH Royal Institute of Technology