HTAP(Hybrid Transactional and Analytical Processing) research

Haiyang Qi and Yuting Fang
Department of Computer Science and Engineering of The Ohio State University

ABSTRACT

Hybrid Transactional/Analytical Processing (or HTAP), which is a state of art database architecture; it has both OLTP(online transaction processing) and OLAP(online analytical processing) capabilities. The main goal of this project is to improve the overall efficiency of processing multiple transactions and analyses on the database, especially in the situation where there are large amounts of transactions and analyses happening at the same time.
.

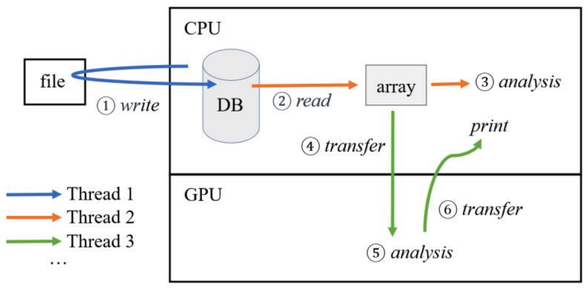
INTRODUCTION

HTAP is a term created by Gartner to describe systems that can support both OLTP (On-line transaction processing) as well as OLAP (on-line analytics processing) within a single transaction. However, the term HTAP is currently used more broadly, even for solutions that support insertions (not necessarily ACID transactions) as well as OLAP queries. Some of these systems have the ability to run analytical queries over the very recent data, while others need some delay before the queries see the latest data..

METHODOLOGY

The method we are using here is that we want to run multiple threads concurrently in the system to simulate the process that there are a lot of transactions and analyzations running in the system concurrently all the time. There are 3 main threads in the system:

- Thread 1: uploading data from the environment into the system (database)
- Thread 2: CPU analyzation: getting data from database and doing analyzation on CPU
- Thread 3: GPU analyzation: transferring data from CPU to GPU and doing the same analyzation as CPU

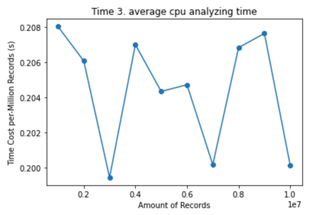


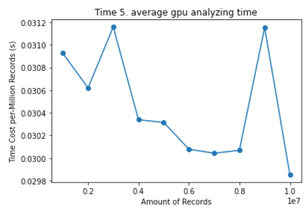
RESULTS

Based on the experiment we described in the methodology, we run multiple transactions on thread1, and randomly getting data from database, running same model on CPU and GPU, we get the results are follows:

Per- million transactions:
Time 3. average CPU analyzing time : **0.50**
std of average CPU analyzing time : **0.0042**

Time 5. average GPU analyzing time is : **0.03**
std of average GPU analyzing time : **0.0006**





CONCLUSION

From the experiment, we can tell that, we run multiple regression or classification models(linear regression model, multiple regression model, SVM model, logistic regression model, etc) , and in most cases, the GPU will perform better than CPU on the condition that the data is big enough to overcome the disadvantages of transferring data from CPU to GPU.

Future Work

Since we can see the difference of the CPU and GPU running on some model, what we can do next are:

- Running multiple more complex models separately on CPU and GPU and compare the results, even neural network models.
- Using CUDA, Pytorch and etc trying to combine the CPU and GPU resources together to solve the problem like transferring data between CPU and GPU, since GPU storage resources are more limited.
- Using POWWOW project raw data(voice monitor data, waves) to run some audio analyzation model and combine this project with POWWOW project.

References

[1] Apache Parquet. https://parquet.apache.org/.
[2] R. Appuswarmy, M. Karpathiotakis, D. Porobic, and A. Ailamaki. The Case For Heterogeneous HTAP. In *CIDR*, 2017.
[3] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia. Spark SQL: Relational Data Processing in Spark. In *SIGMOD*, pages 1383–1394, 2015.
[4] J. Arulraj, A. Pavlo, and P. Menon. Bridging the Archipelago Between Row-Stores and Column-Stores for Hybrid Workloads. In *SIGMOD*, pages 583–598, 2016.
[5] R. Barber, C. Garcia-Arellano, R. Grosman, R. Mueller, V. Raman, R. Sidle, M. Spilchen, A. Storm, Y. Tian, P. Tözün, D. Zilio, M. Huras, G. Lohman, C. Mohan, F. Özcan, and H. Pirahesh. Evolving Databases for New-Gen Big Data Applications. In *CIDR*, 2017.
[6] A. Boehm, J. Dittrich, N. Mukherjee, I. Pandis, and R. Sen. Operational analytics data management systems. *PVLDB*, 9:1601–1604, 2016.
[7] P. Boncz, M. Zukowski, and N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *CIDR*, 2005.
[8] Apache Cassandra. http://cassandra.apache.org.
[9] A. Costea, A. Ionescu, B. Răducanu, M. Switakowski, C. Bârca, J. Sompolski, A. Luszczak, M. Szafrański, G. de Nijs, and P. Boncz. Vectorh: Taking sql-on-hadoop to the next level. In *SIGMOD '16*, pages 1105–1117, 2016.