CS 839: Topics in Database Management Systems

Lecture 7: Transaction Processing-1

Xiangyao Yu

9/27/2023

Updates

Uploaded a list of project ideas to the course website

Please start to prepare for the course project early!

- **Project proposal is due on 10/16**

Group Discussion from Lecture #5

Multiple papers suggest a new “layer” between compute and storage (e.g., spectrum layer in redshift, S3 select, metadata layer in Lakehouse). Please summarize the key functions that previous work proposed to push to such a layer.

Can you think of other functions that can be pushed to such a layer as well?

Can such a layer enable further performance optimizations or new use cases?

Group Discussion from Lecture #5

Previously developed functions in this layer

- Filtering and querying (S3 select)
- Intermediate layer can be used to store intermediate results (Snowflake)
- Query optimization (spectrum)
- Log replay (Socrates)
- Transactions and metadata store (Lakehouse)
- Data shuffle and aggregation (Dremel)
- Access control for security and data isolation (in-storage service)

Group Discussion from Lecture #5

New functions in this layer

- Cache materialized views that are potentially shared across multiple users
- Data format translation (e.g., csv, json, parquet)
- Time travel that access data in different time snapshot
- Real-time analytics and stream processing
- Data virtualization: allow disparate data sources to appear as a unified source
- ETL functions
- GPU cluster
- Memory disaggregation

Transaction Processing-1 – Q/A

What happens if there is a gray failure in the primary? Do you just get degraded performance?

Correctness of Socrates for race conditions?

Cost efficiency of Spanner?

How exactly is TrueTime better than logical time?

Spanner and Aurora targeting the same goal?

Spanner is not very efficient for analytical queries. Solutions?

Use Spanner outside Google?

Discussion Question

Socrates and Aurora use a storage disaggregation architecture but supports only a single write node. Spanner supports multiple write nodes, but largely follows a shared-nothing architecture. Is it possible to support multiple write nodes in a storage disaggregation architecture? How would you design such a system? What are the advantage and disadvantages of your design compared to Spanner?

Besides logging, storage, and computation, what other functions are good candidates for disaggregation in a transactional database?