HITACHI
Inspire the Next

Integrated Architecture of SQL Engine and Data Analytics Tool with Apache Arrow Flight and Its Performance Evaluation

2021/10/03-07 @ DATA ANALYTICS 2021
Yuichiro Aoki email: yuichiro.aoki.jk@hitachi.com
Satoru Watanabe email: satoru.watanabe.aw@hitachi.com
Research and Development Group, Hitachi, Ltd.

Short Resume

HITACHI
Inspire the Next

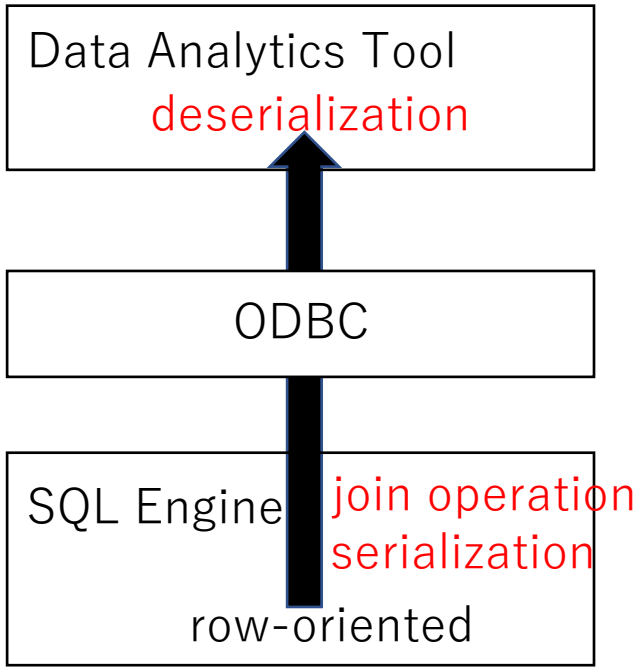
- Presenter: Yuichiro Aoki
- Senior Researcher at Hitachi, Ltd.
- Research Interest: Data science and parallel processing

Contents

HITACHI
Inspire the Next

2. Background

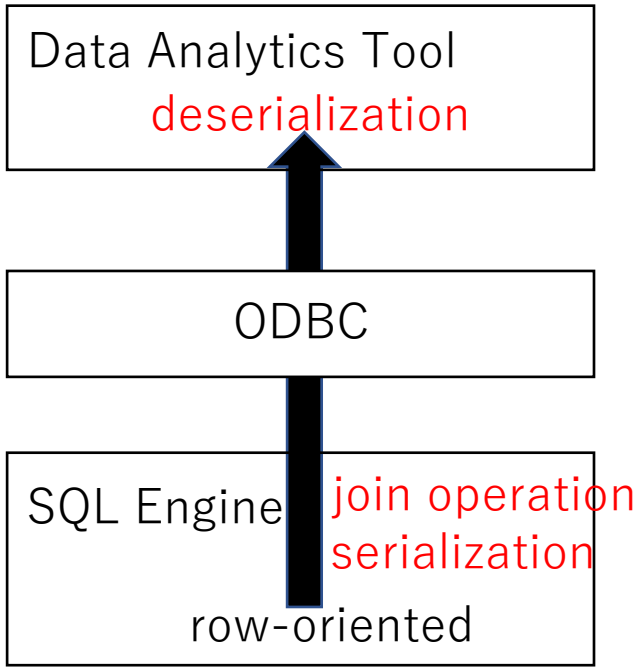
- Data analytics in enterprise systems require huge amount of data in databases.
- Conventionally, SQL engines retrieve the data from the database using traditional Open Database Connectivity (ODBC).
- In complicated data analytics, the data needs to be joined.



3. Problems

HITACHI
Inspire the Next

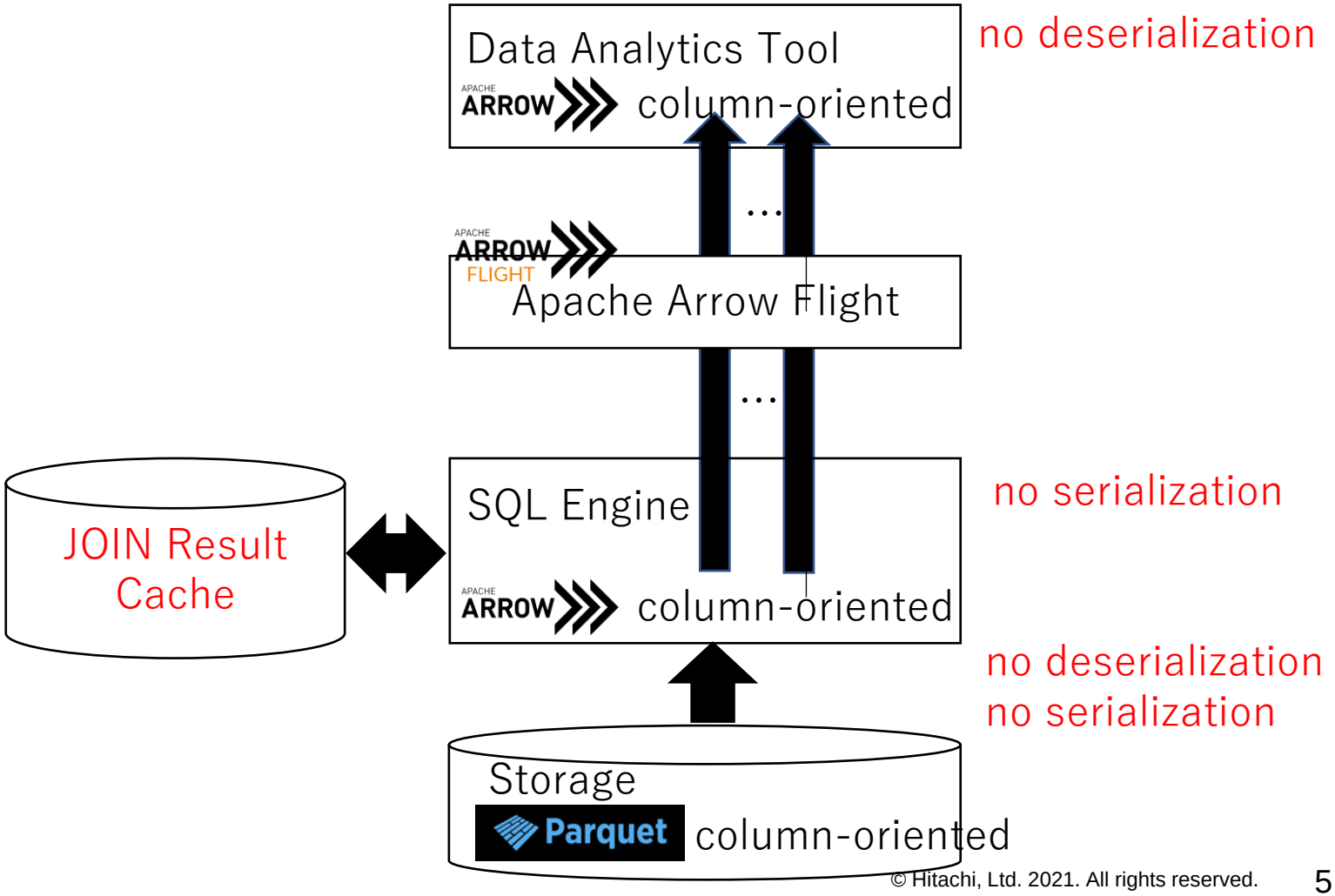
- The data is serialized in the SQL engine and deserialized in data analytics tools. Serialization/deserialization demands many memory copies and takes a lot of time.
- Complicated join operations takes a lot of time.
- They are bottlenecks of the data analytics performance.



4. Proposed Architecture

HITACHI
Inspire the Next

- We propose a new integrated architecture of SQL Engine and data analytics tool with Apache Arrow Flight.

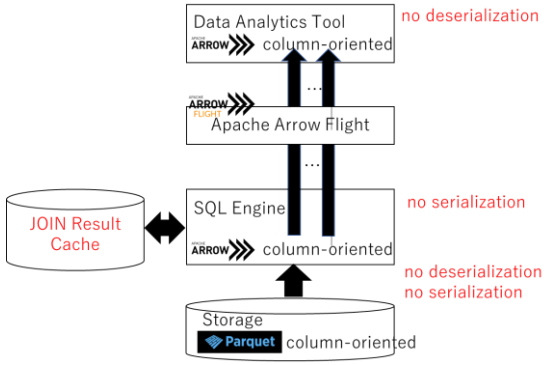


4. Proposed Architecture: Apache Arrow Flight

HITACHI
Inspire the Next

- To reduce the number of serializations/deserializations, we use:
  - Data transfer framework between the SQL engine and the data analytics tool: Apache Arrow Flight
  - Data format: Apache Arrow in-memory column-oriented data format.

- The same column-oriented data format does not need data copy (serialization/deserialization) at the boundary of SQL engine/data transfer framework and data transfer framework/data analytics tool.

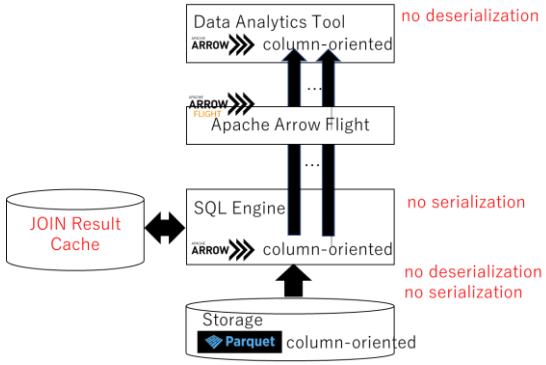


4. Proposed Architecture: JOIN Result Cache

HITACHI
Inspire the Next

- To reduce complicated join time, we use:
  - JOIN Result Cache

- How it works:
  - a join query is precomputed and cached if it has the same columns as previous join queries but has different tables. (ex. POS data, such as sales20210803, sales20210804, ・・・)
  - The tables are inferred from the history of table usage in previous join queries.
- Effects and Use cases:
  - If cached, it reduces join operation time.
  - For example, it is useful for daily processing of POS system.



5. Performance Evaluation

HITACHI
Inspire the Next

■ Performance Evaluation Environment

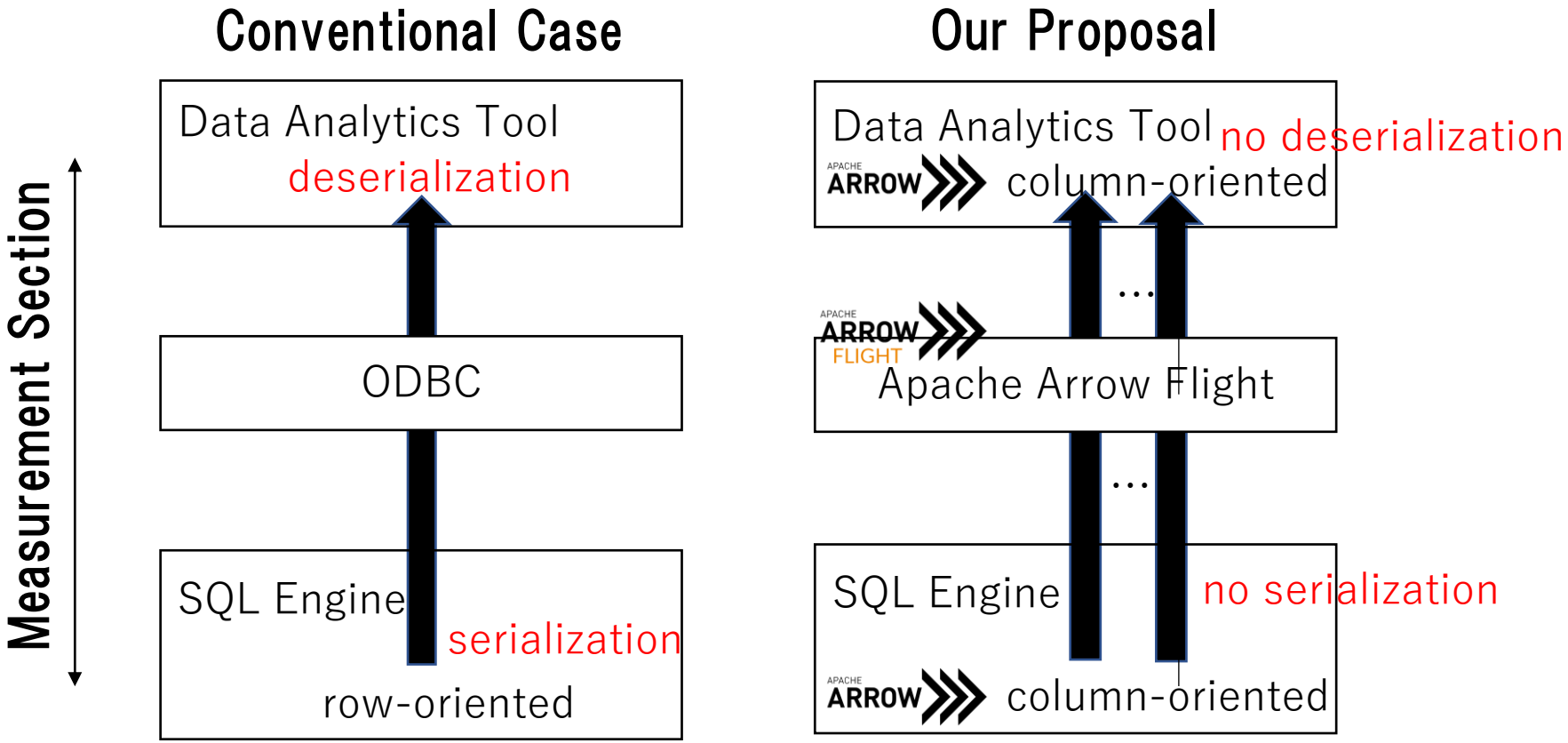
- We used a virtual machine and CentOS 7.8.

Item	Content	
CPU	Intel® Core™ i7-8665U (4 cores / 8 threads)	
Memory	32GB	
Host OS	Windows 10 Pro 2004	
Virtual Machine (VM)	Oracle VM Virtual Box 6.1.14	
	VM CPU	4 processors
	VM Memory	16GB
Guest OS	CentOS 7.8	

5. Performance Evaluation (1) Apache Arrow Flight

HITACHI
Inspire the Next

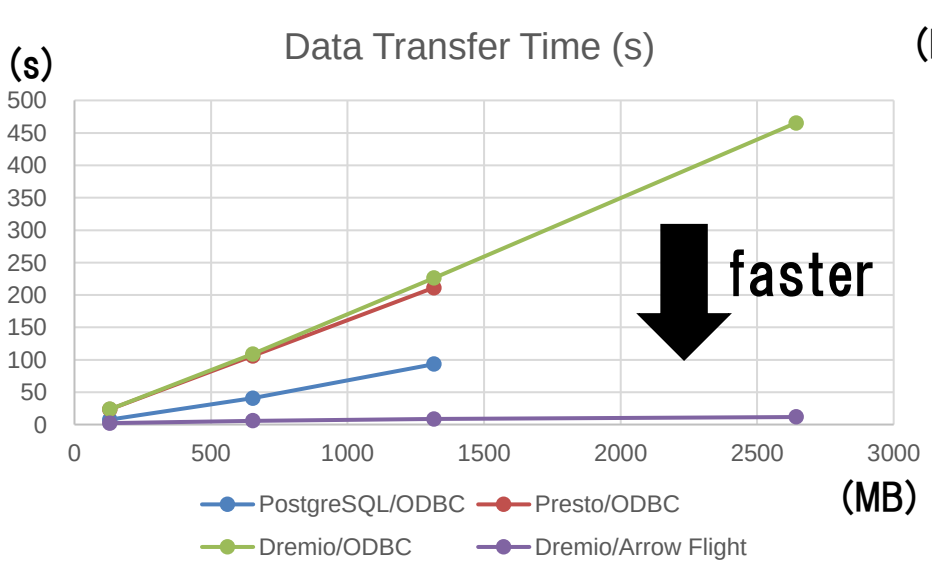
- Evaluation (1): Apache Arrow Flight Performance
  - Measure execution time of data transfer from SQL engine to data analytics tool
  - Dataset size: 130MB ~ 2.6GB

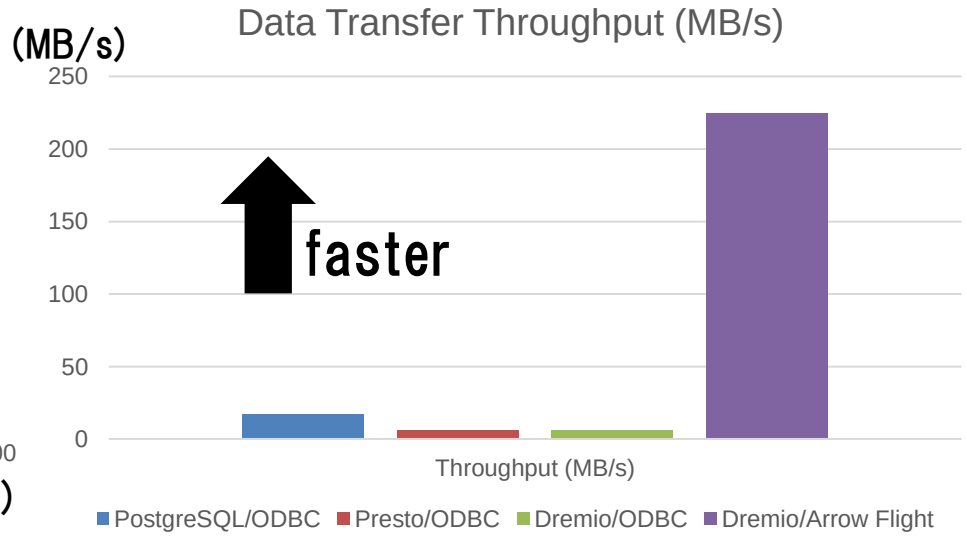


5. Performance Evaluation (1) Apache Arrow Flight

HITACHI
Inspire the Next

- Evaluation (1): Apache Arrow Flight Performance
  - Dremio/Apache Arrow Flight runs 13.1～37.4 times faster than PostgreSQL/ODBC, Presto/ODBC, and Dremio/ODBC.
  - Data transfer throughput of Dremio/Apache Arrow Flight is 224MB/s.



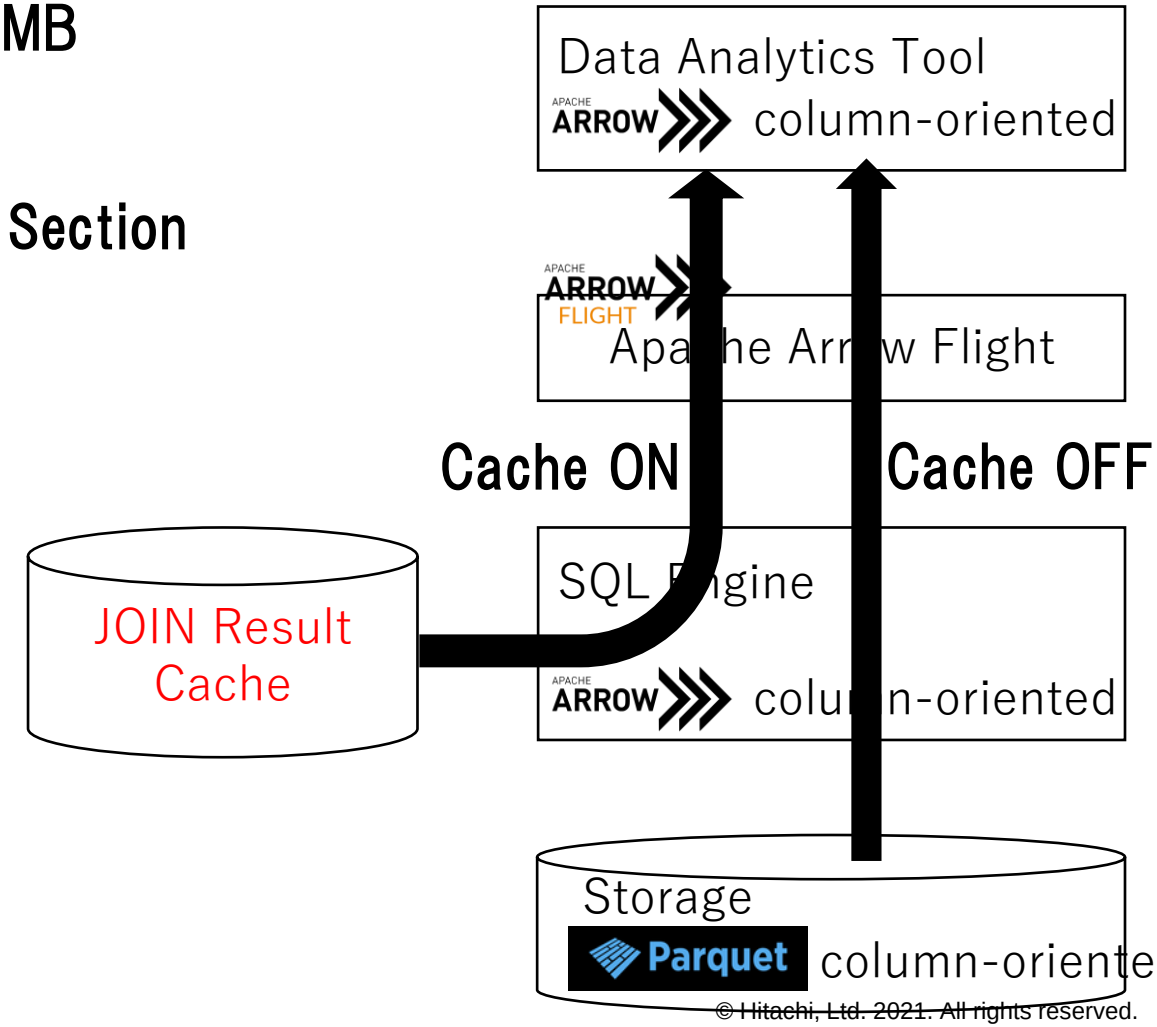


5. Performance Evaluation (2) JOIN Result Cache

HITACHI
Inspire the Next

- Evaluation (2): JOIN Result Cache Performance
  - Measure execution time of query “SELECT * FROM table”. The table is generated using complicated joins.
  - Dataset size: 256MB

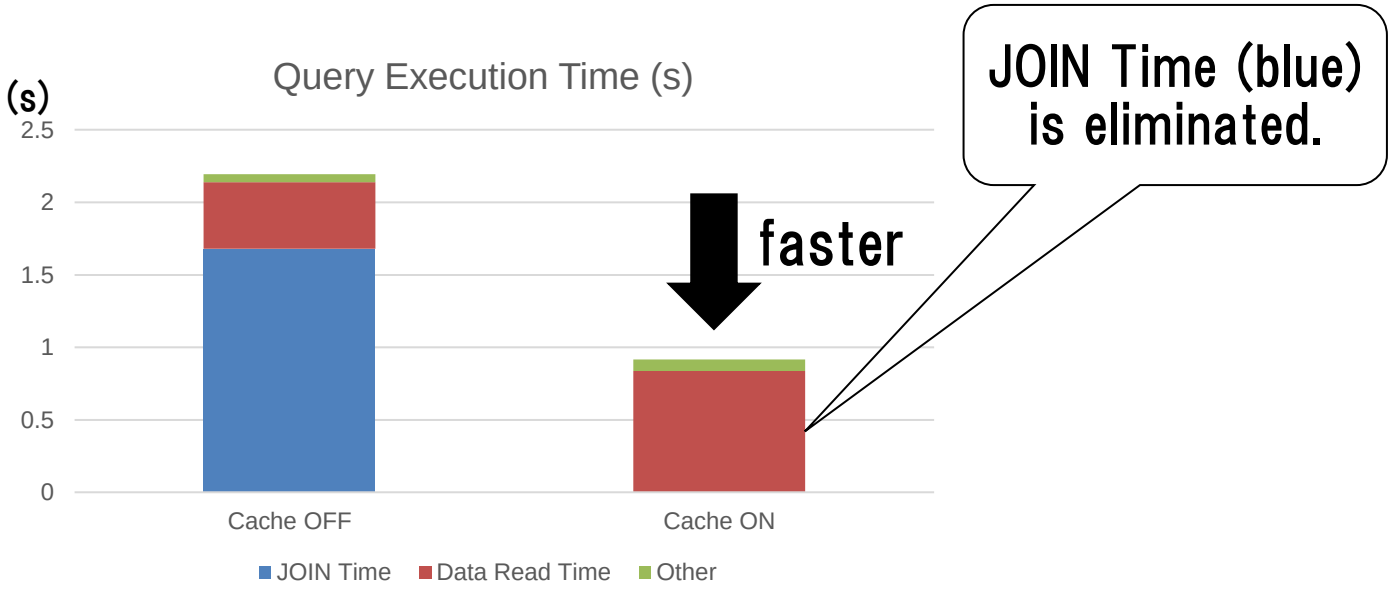
➡ Measurement Section



5. Performance Evaluation (2) JOIN Result Cache

HITACHI
Inspire the Next

- Evaluation (2): JOIN Result Cache Performance
  - In Cache ON case, JOIN time is eliminated. Join query execution time in Cache ON case is 2.4 times faster than in Cache OFF case.



6. Discussion

HITACHI
Inspire the Next

- Apache Arrow Flight does not need the serialization of the data before data transfer and the deserialization after it. This is why Apache Arrow Flight outperforms ODBC.
- However, in many cases, data analytics tools written in Python® use DataFrame when users analyze the data. DataFrame does not use Apache Arrow format and serialization of the data is needed. This may be another bottleneck of the performance.
- After serialization/deserialization disappears, JOIN time is a next bottleneck. JOIN Result Cache reduces JOIN time.
- This architecture can cross the cloud boundaries, because no specific hardware, such as RDMA, is not needed.
- In addition, if the system resides in one cloud, we can use memory-mapped files to read to/write from the storage. It is much faster than usual I/O system calls.

7. Related Work

HITACHI
Inspire the Next

- We use Apache Arrow/Apache Arrow Flight.
- We have JOIN Result Cache.
- We show data transfer performance comparison.
- We show an integrated architecture of data analytics system.

	Apache Arrow	Apache Arrow Flight	JOIN Result Cache	Data Transfer Performance Comparison	Integrated Data Analytics System
Dremio	✓	✓	✓		
Li et al.	✓	✓		✓	
Magpie	✓	✓		✓	
ImmVis					✓
InfluxData	✓	✓			
RAPIDS DataBricks BigQuery Snowflake	✓				
This Study	✓	✓	✓	✓	✓

8. Conclusion

HITACHI
Inspire the Next

- We proposed a new architecture for a data analytics system using column-oriented Apache Arrow/Arrow Flight and JOIN Result Cache.
- Performance evaluation shows:
  - Apache Arrow Flight transfers the data 13.1-37.4 times faster than ODBC because serialization/deserialization of the data is eliminated.
  - JOIN Result Cache accelerates the query by 2.4 times using precomputed JOIN results.
- In future work, we will design and implement such a data analytics system using Apache Arrow and Apache Arrow Flight.

HITACHI

Inspire the Next