influxdata®

bauplan

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

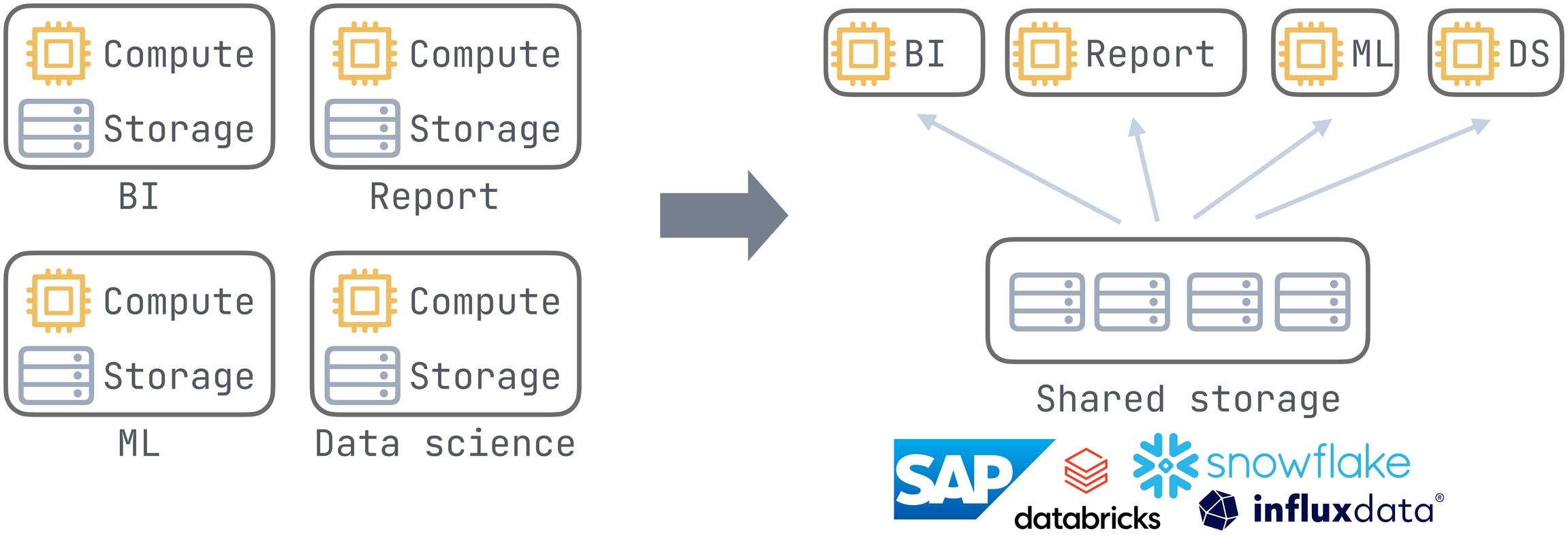
spiral

LiquidCache

Efficient Pushdown Caching for Cloud-Native Data Analytics

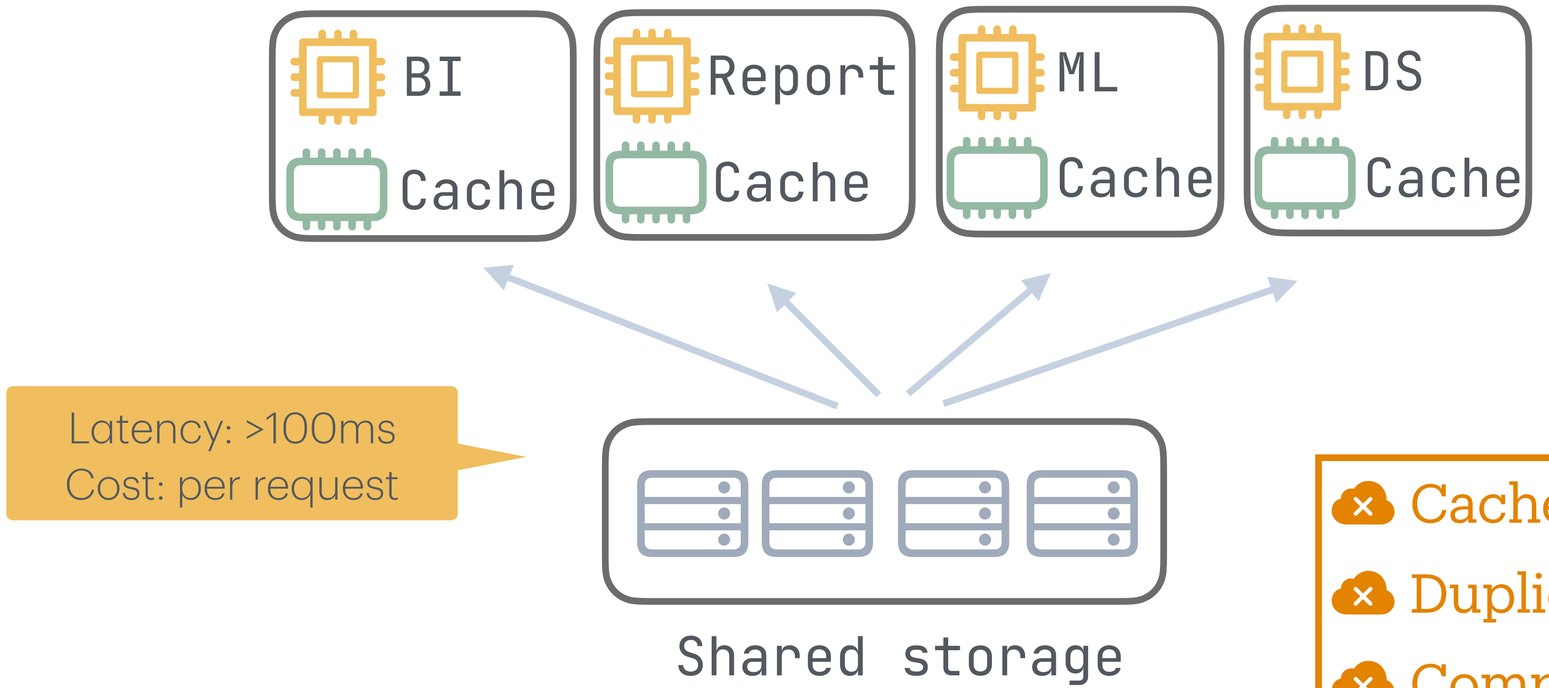
Xiangpeng Hao, University of Wisconsin-Madison
https://xiangpeng.systems

On premise → cloud (2010-2020)



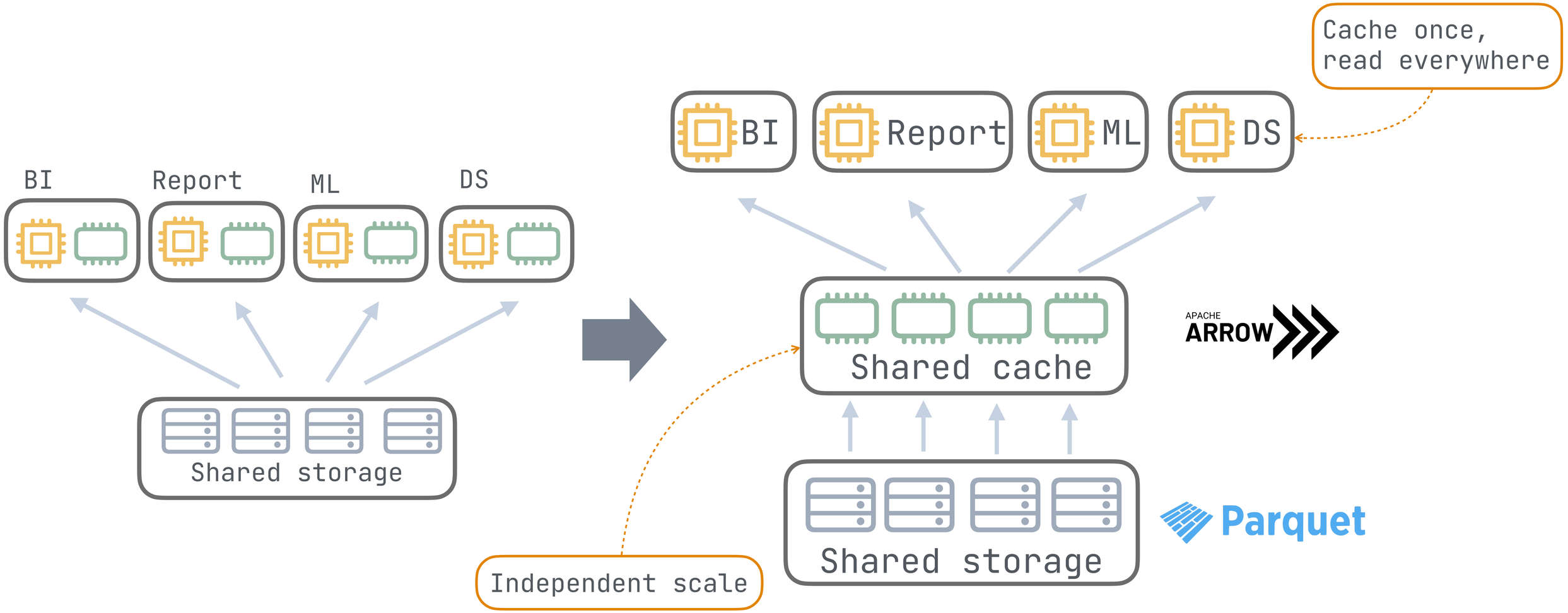
Elastic, durable, scalable

Every system has a cache (2020-2025)

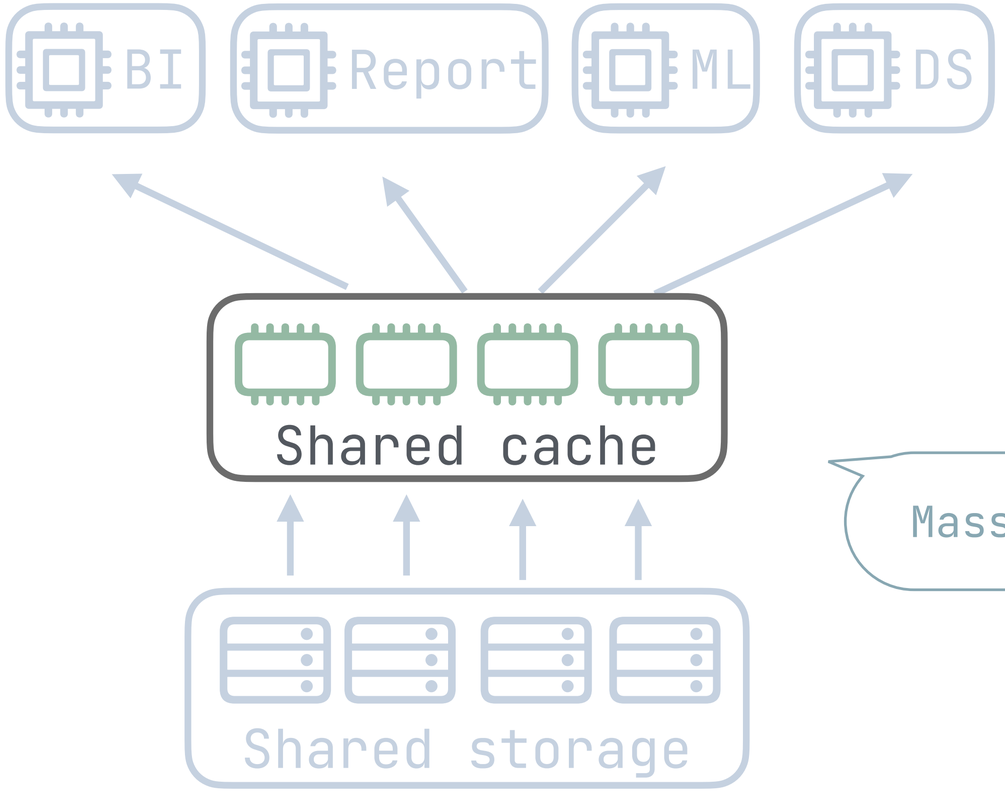


- Cache couples with compute
- Duplicative cache
- Complex data management

Vision: shared cache (2025+)



First attempt: byte cache

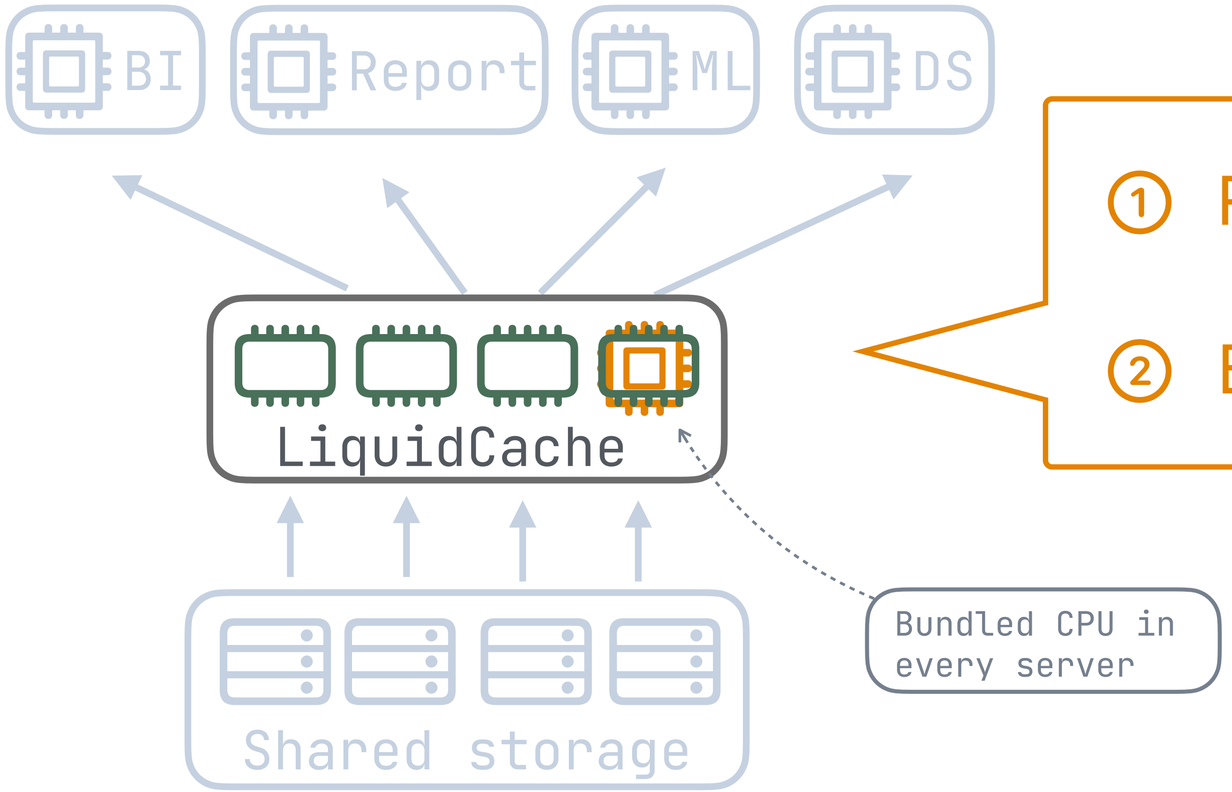


Challenge:
Network bandwidth

Massive data transfer!

Existing systems: only used as metadata cache

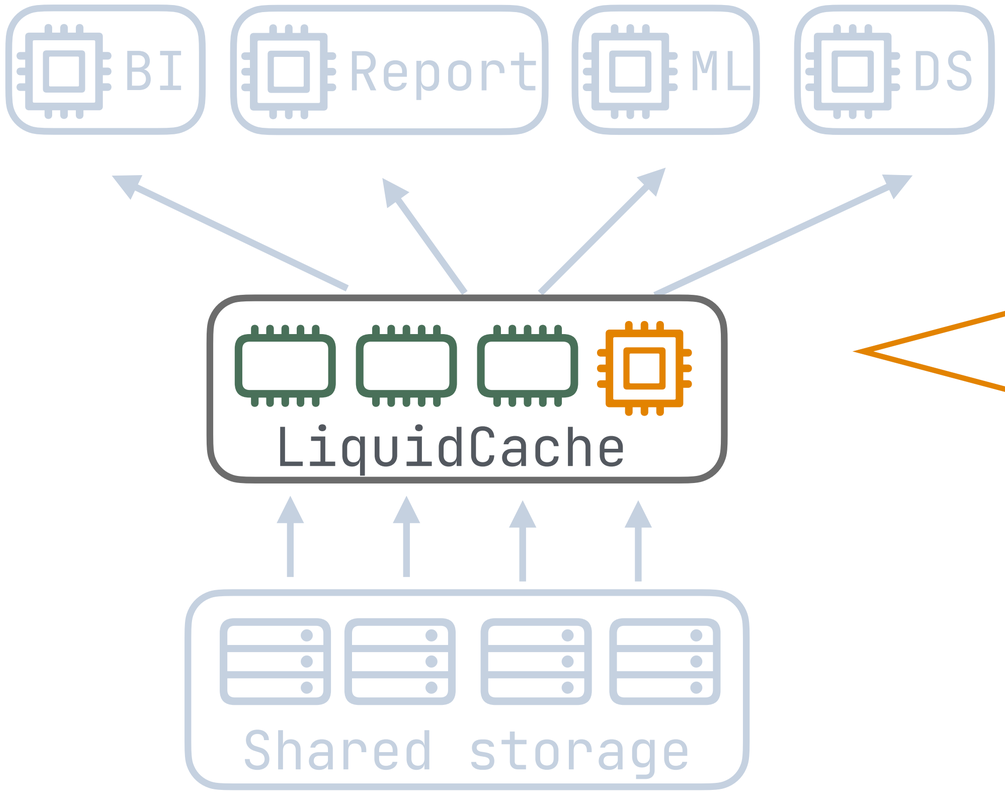
LiquidCache = compute + data



① Pushdown to reduce network

② Efficient predicate eval

LiquidCache = compute + data



① Pushdown to reduce network

② Efficient predicate eval

Prior art: push down

1. Access Path Selection in a Relational Database Management System (1979)
2. Support for Repetitive Transactions and Ad Hoc Queries in System R (1981)
3. Column-Oriented Database System (2009)

Pushdown to
{NIC,SSD,FPGA,DPU}

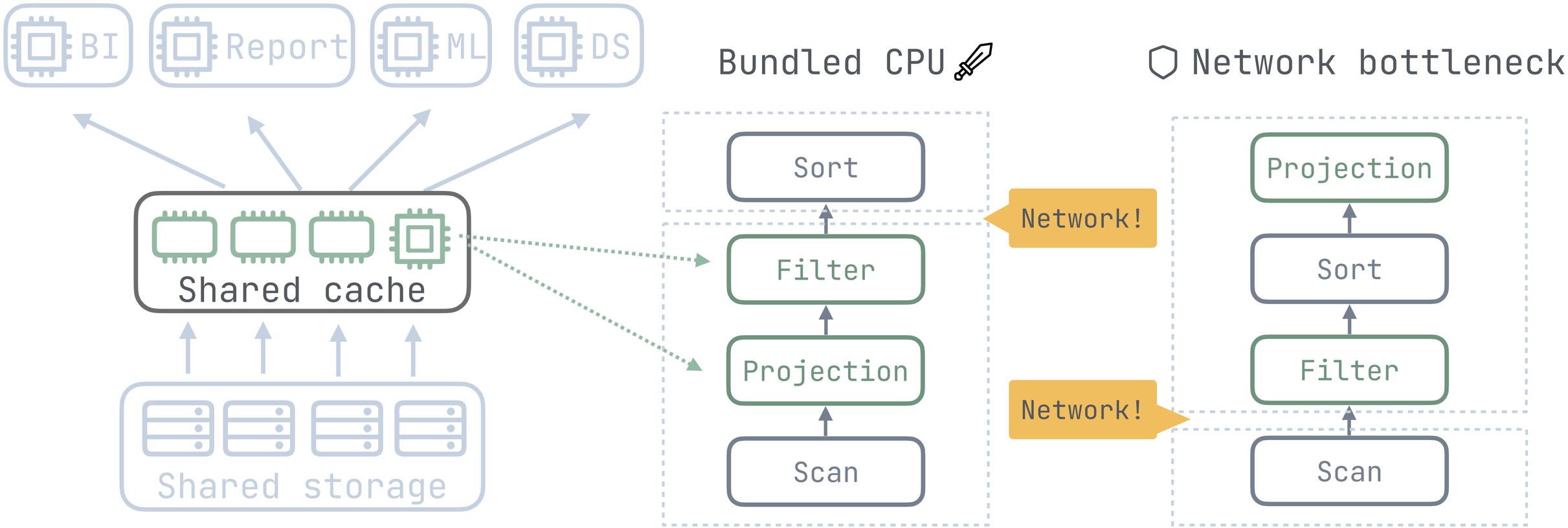
1. PushdownDB: Accelerating a DBMS using S3 Computation (2020)
2. FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS (2021)
3. Fusion: An Analytics Object Store Optimized for Query Pushdown. (2025)

Pushdown within
query engine

1. Query Processing on Smart SSDs: Opportunities and Challenges (2013)
2. Introduction to the IBM Netezza warehouse appliance. (2011)
3. DPDPU: Data Processing with DPUs. (2024)
4. Towards Accelerating Data Intensive Application's Shuffle Process Using SmartNICs (2023)

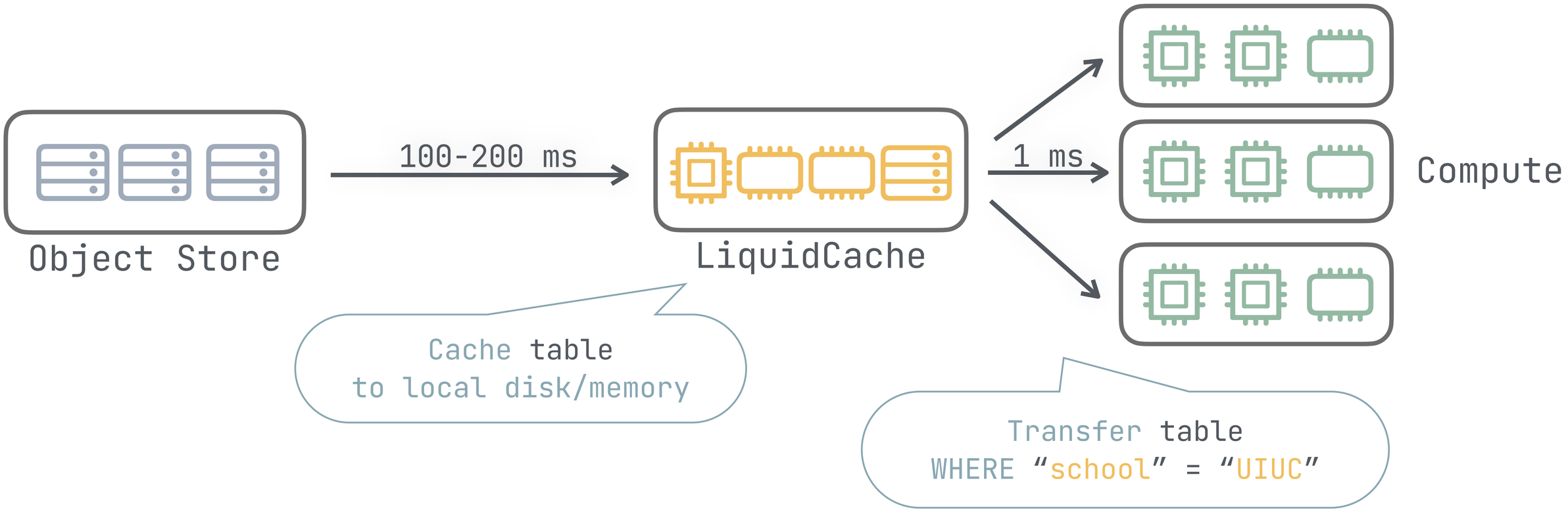
Pushdown to
Cloud storage

Pushdown to cache service



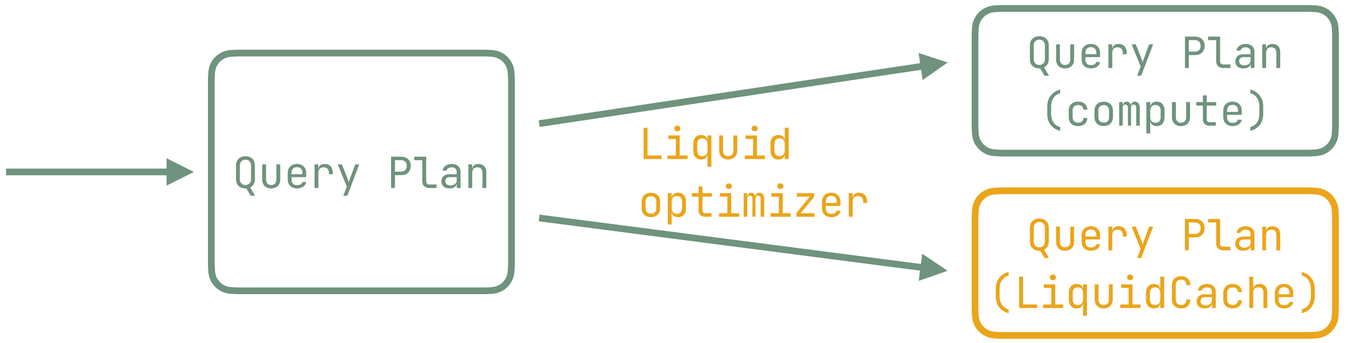
Pushdown example

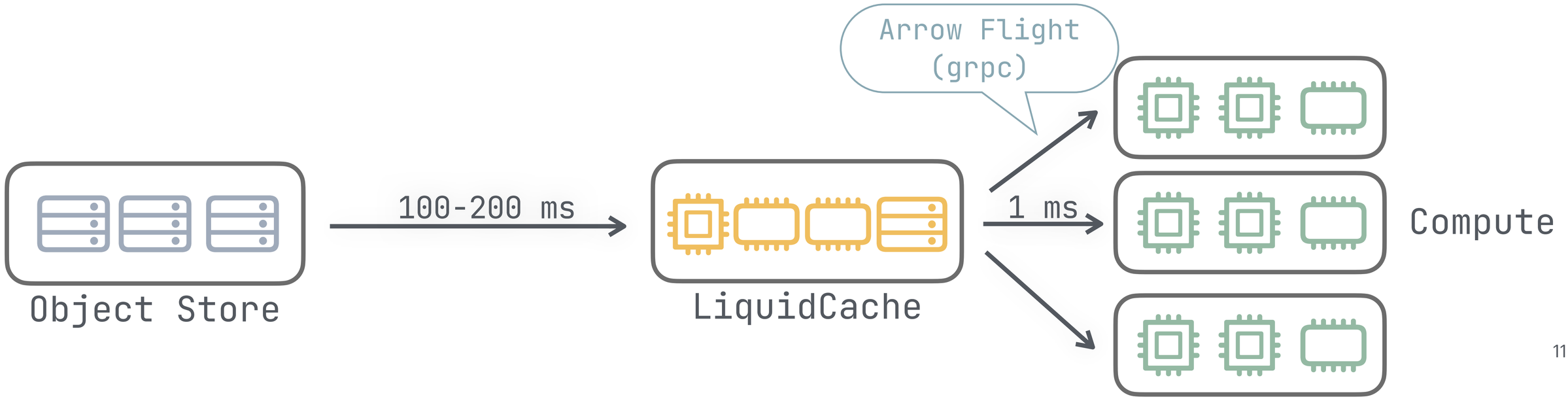
```
SELECT DISTINCT(“name”)
FROM table
WHERE “school” = “UIUC”
```



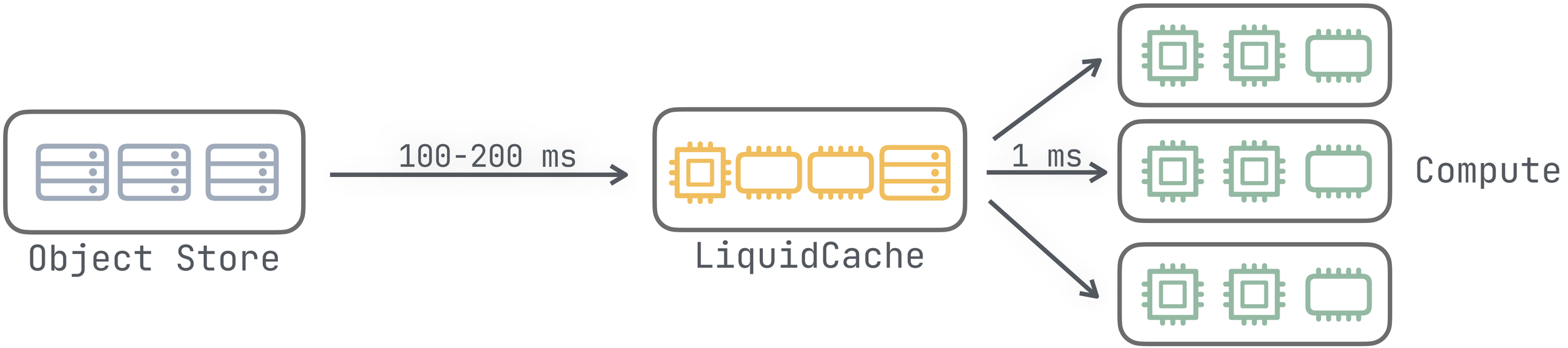
LiquidCache Architecture

```
SELECT DISTINCT(“name”)
FROM table
WHERE “school” = “UIUC”
```



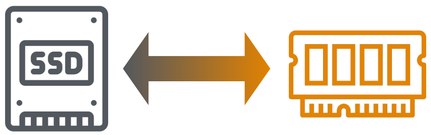


Fine-grained caching



Parquet file

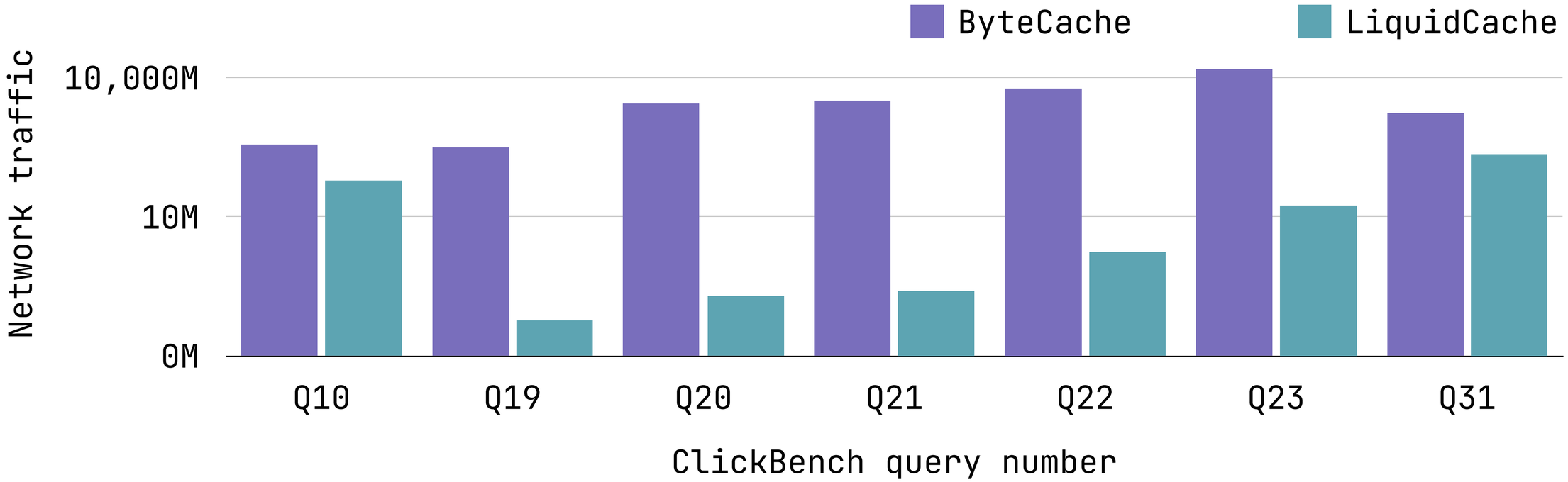
Parquet file

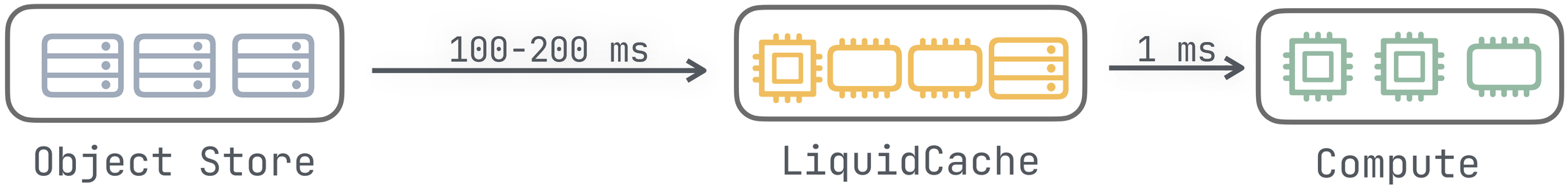


Fine-grained caching

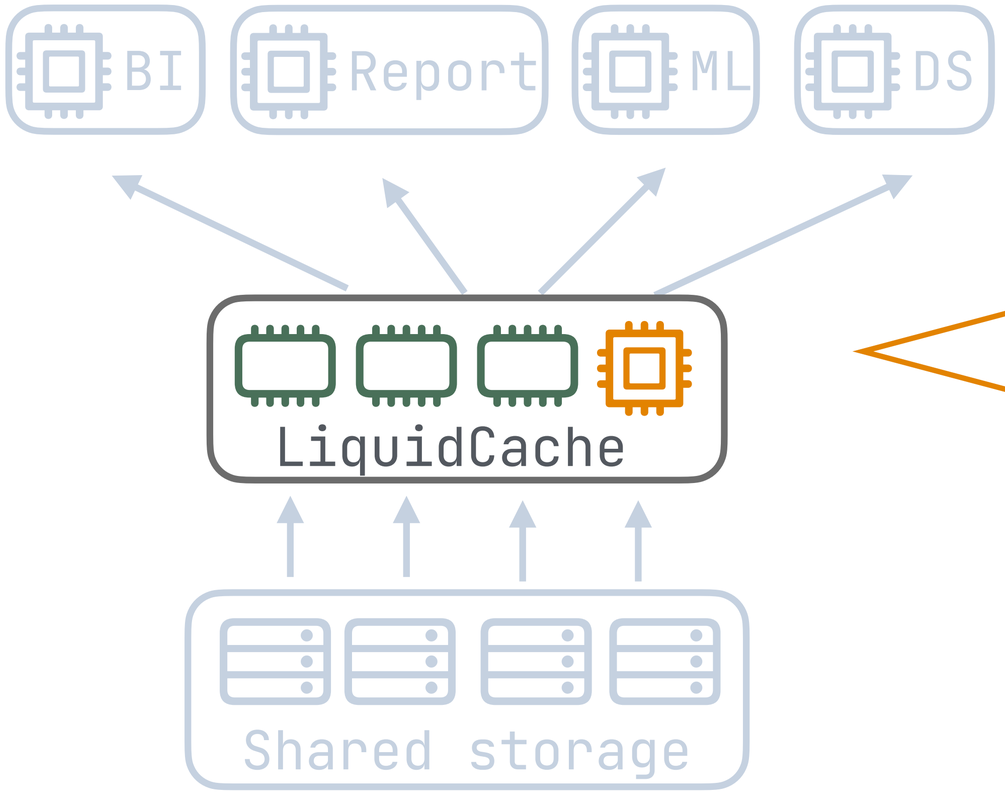
- Hot data in memory
- Cold data in disk
- Unused data are not cached

Push down reduce network traffic





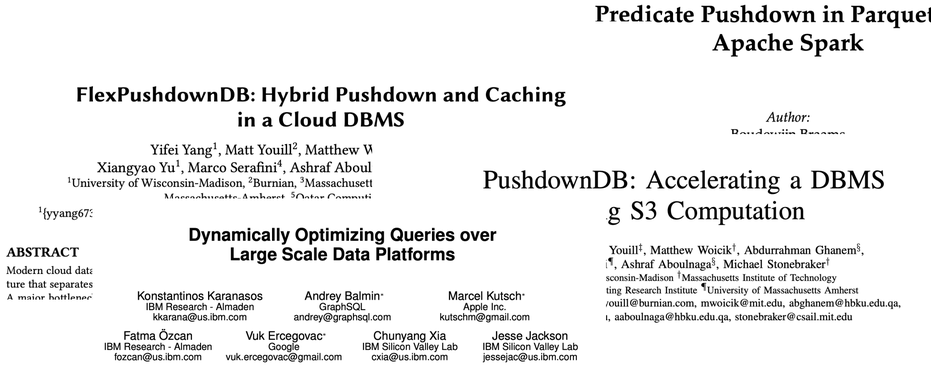
LiquidCache = compute + data



① Pushdown to reduce network

② Efficient predicate eval

Pushdown is expensive



FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS

Yifei Yang[1], Matt Youill[2], Matthew W
Xiangyao Yu[1], Marco Serafini[4], Ashraf Aboul
[1]University of Wisconsin-Madison, [2]Burnian, [3]Massachusett

[1]{yyang67

ABSTRACT

Modern cloud data
ture that separates

Dynamically Optimizing Queries over Large Scale Data Platforms

Konstantinos Karanasos, IBM Research - Almaden, kkarana@us.ibm.com
Andrey Balmin, GraphSQL, andrey@graphsql.com
Marcel Kutsch, Apple Inc., kutschm@gmail.com
Fatma Özcan, IBM Research - Almaden, fozcan@us.ibm.com
Vuk Ercegovac, Google, vuk.ercegovac@gmail.com
Chunyang Xia, IBM Silicon Valley Lab, cxia@us.ibm.com
Jesse Jackson, IBM Silicon Valley Lab, jessejac@us.ibm.com

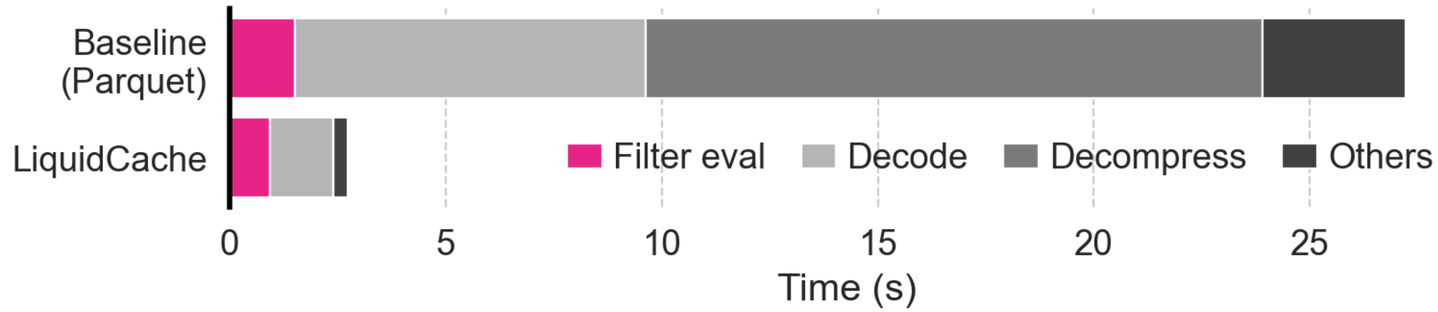
Predicate Pushdown in Parquet ar
Apache Spark

Author:

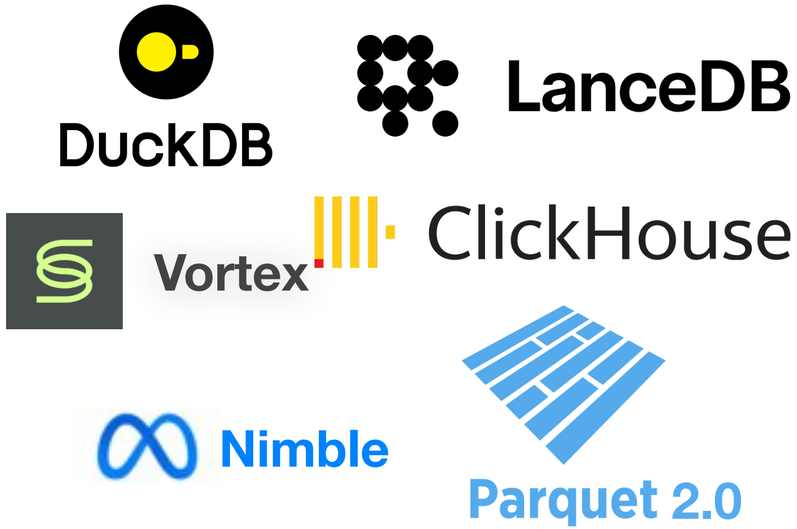
PushdownDB: Accelerating a DBMS
g S3 Computation

Youill[‡], Matthew Woicik[†], Abdurrahman Ghanem[§],
[¶], Ashraf Aboulnaga[§], Michael Stonebraker[†]
sconsin-Madison [†]Massachusetts Institute of Technology
ting Research Institute [¶]University of Massachusetts Amherst
/ouill@burnian.com, mwoicik@mit.edu, abghanem@hbku.edu.qa,
, aaboulnaga@hbku.edu.qa, stonebraker@csail.mit.edu

Data decoding takes too long!



Prior Art: “better” file formats



Industry makes new file formats

1. FSST: Fast Random Access String Compression (2020)
2. BtrBlocks: Efficient Columnar Compression for Data Lakes (2023)
3. The Fast Lanes Compression Layout: Decoding 100 Billion Integers per Second with Scalar Code (2023)
4. ALP: Adaptive Lossless floating-Point Compression (2023)
5. The FastLanes File Format (2025)
6. AnyBlox: A Framework for Self-Decoding Datasets (2025)
7. F3: The Open-Source Data File Format for the Future (2025)

Researchers propose new encodings

Battle tested

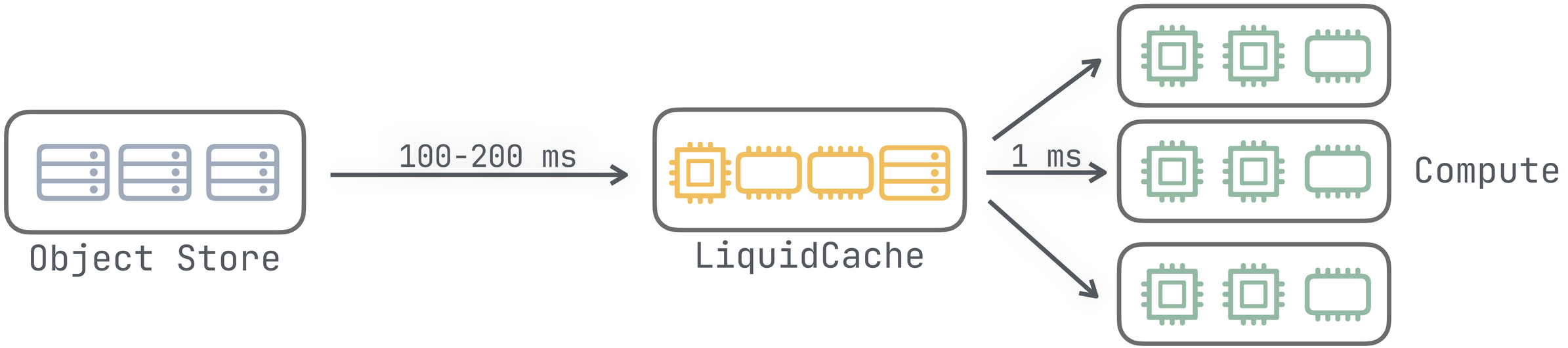
Open & stable governance

Cross-engine sharing

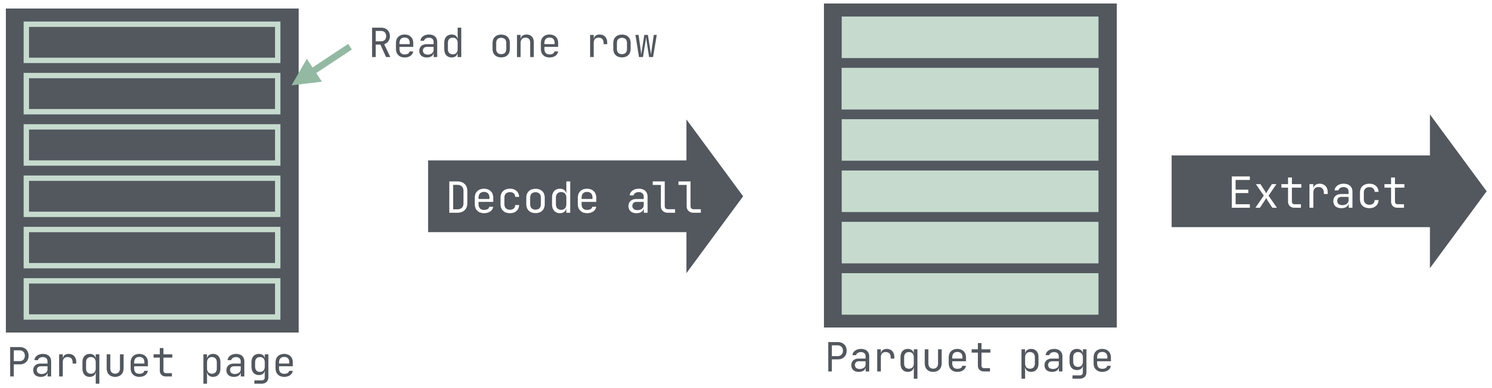
Reasonable performance

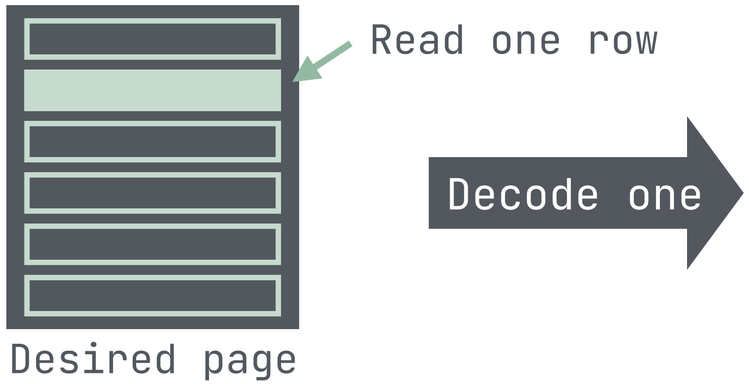
But Industry is locked to old Parquet

LiquidCache: cache-only format

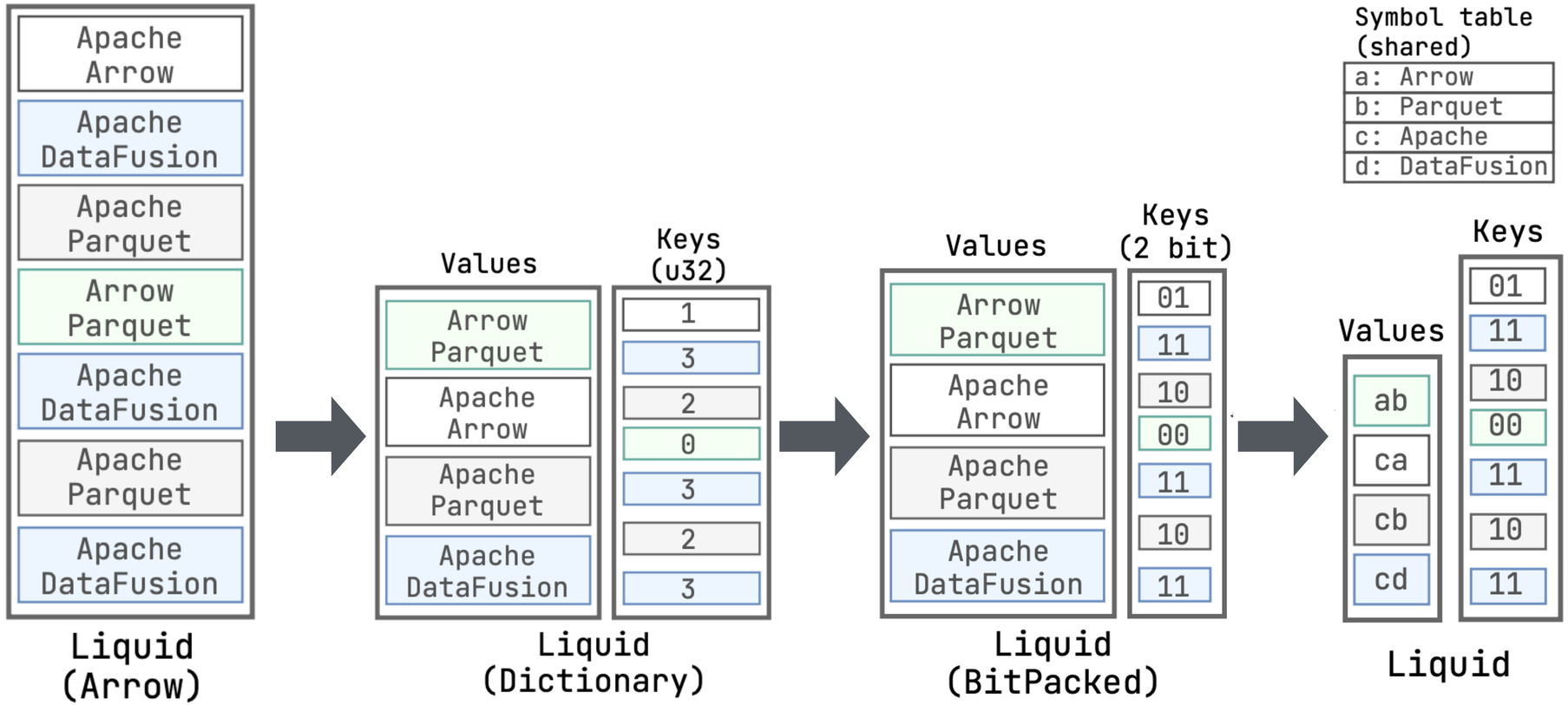


Principle:
Each row must be independently decodable





Each row must be independently decodable (string example)



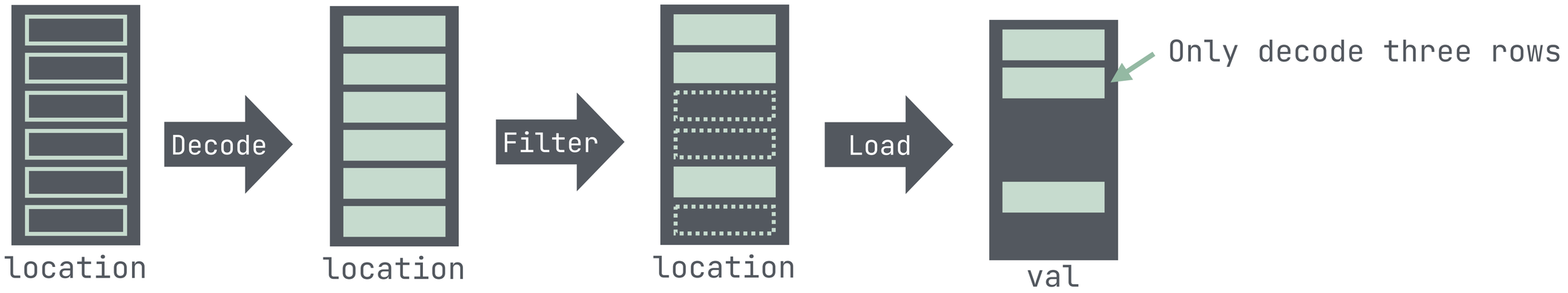
No general purpose compression

Leverages state-of-the-art encoding schemes

Carefully designed encoding/layout for each data types

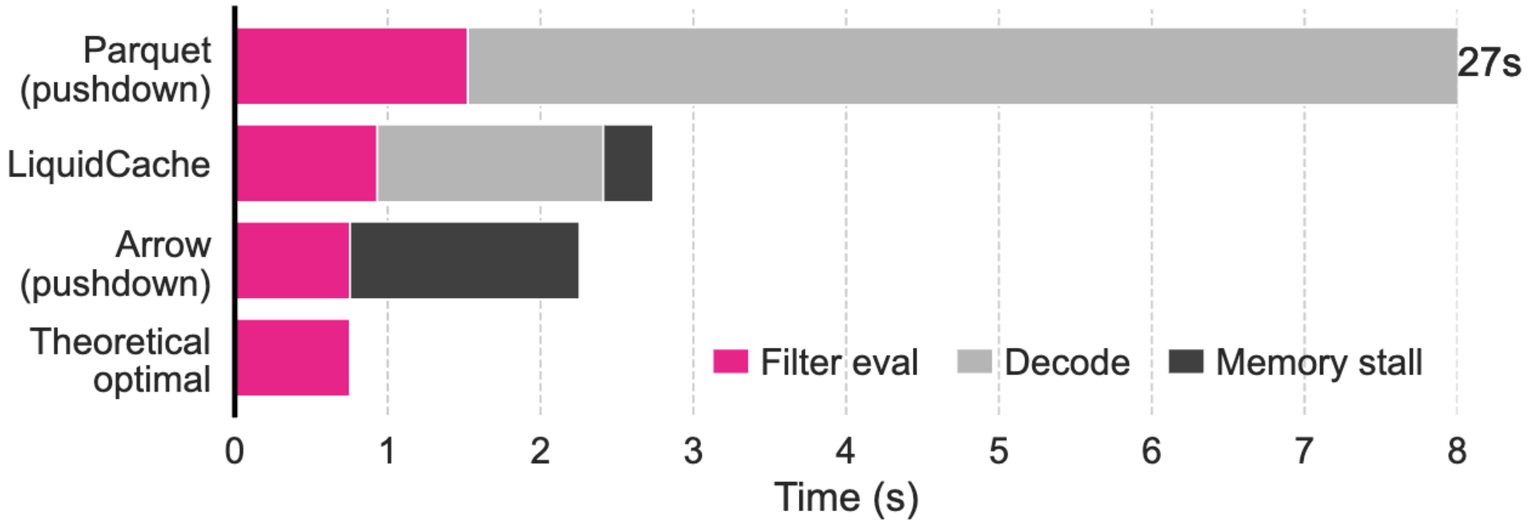
Co-design with filter pushdown (selective decoding)

```sql
SELECT val, location
FROM sensor_data
WHERE location = 'office';
```



More in our paper:

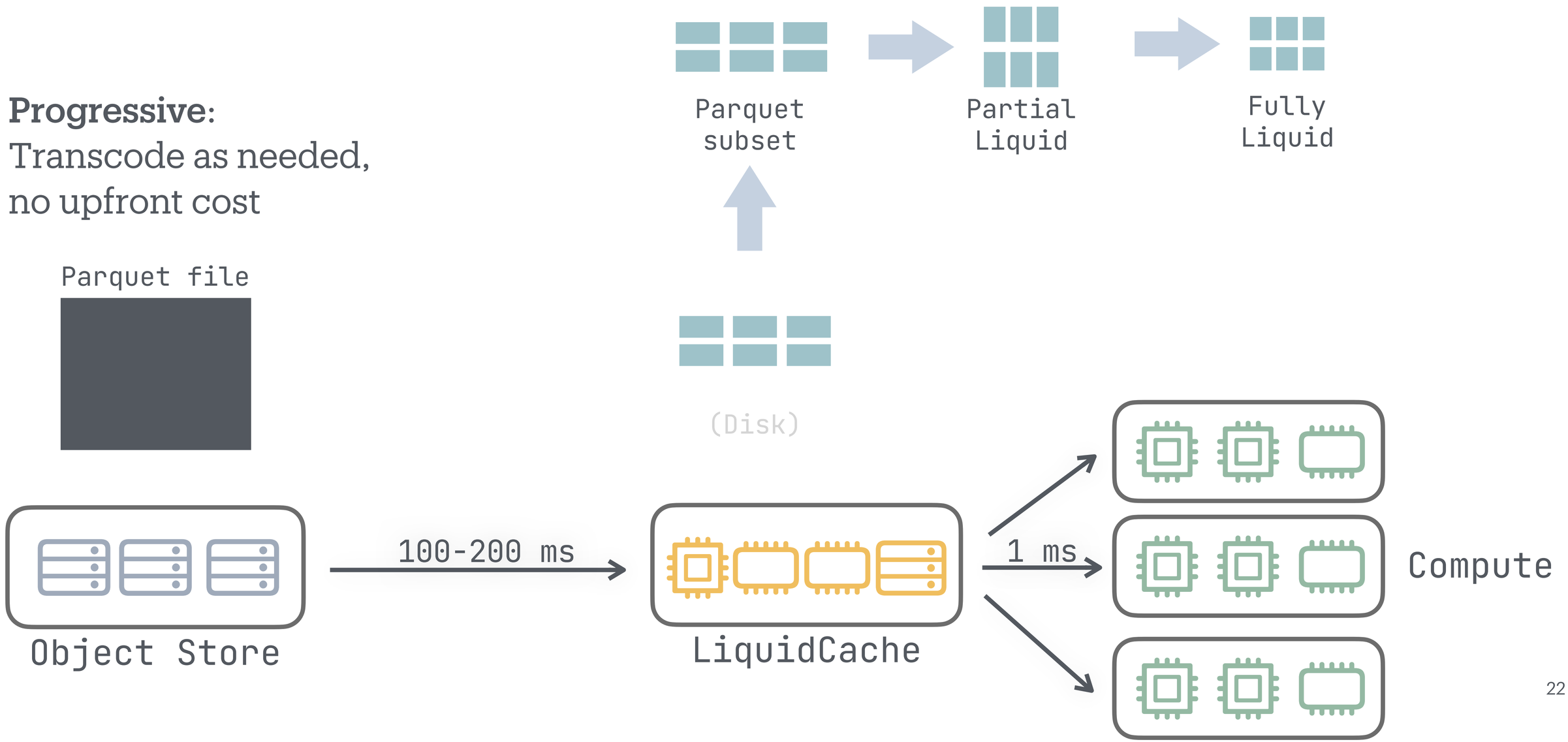
- Late materialization
- Evaluate on encoded data
- Partial decoding



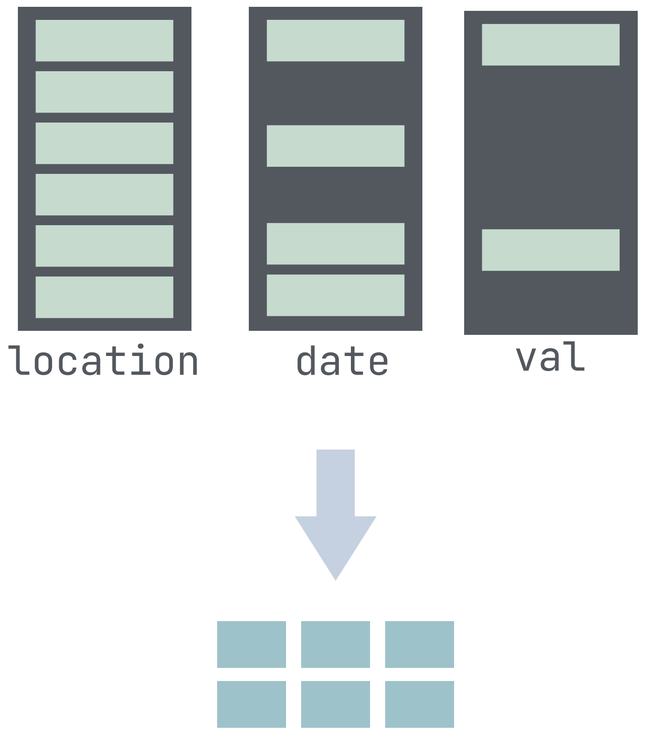
Decoding cost: close to theoretical optimal

Progressive transcoding

Progressive:
Transcode as needed,
no upfront cost

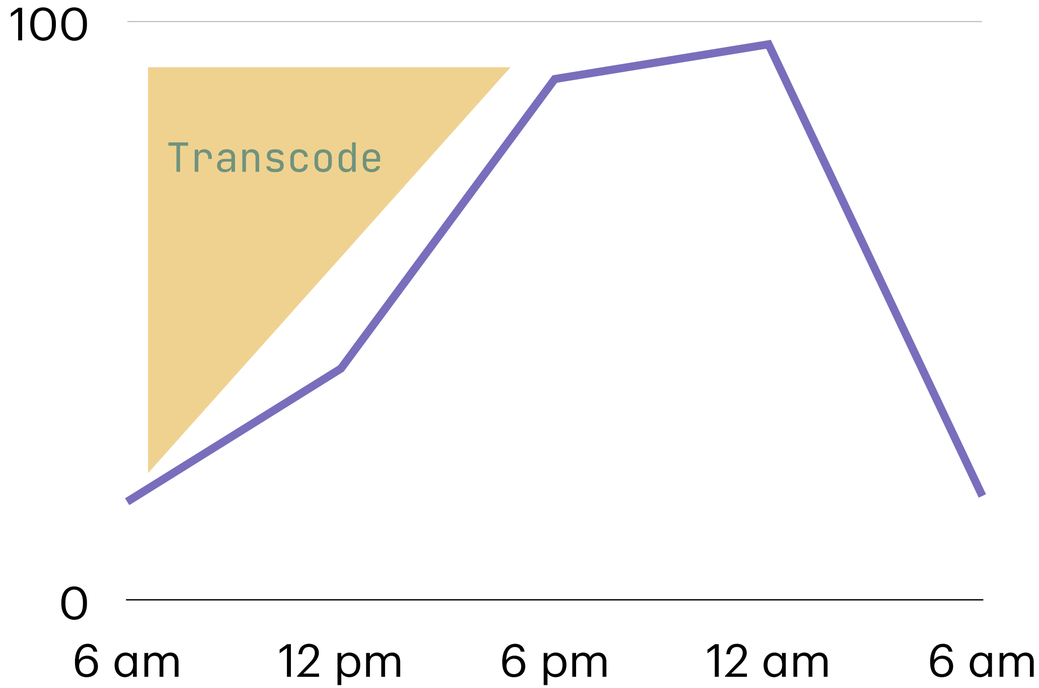


Selective transcoding



Selective:
Transcode only touched data

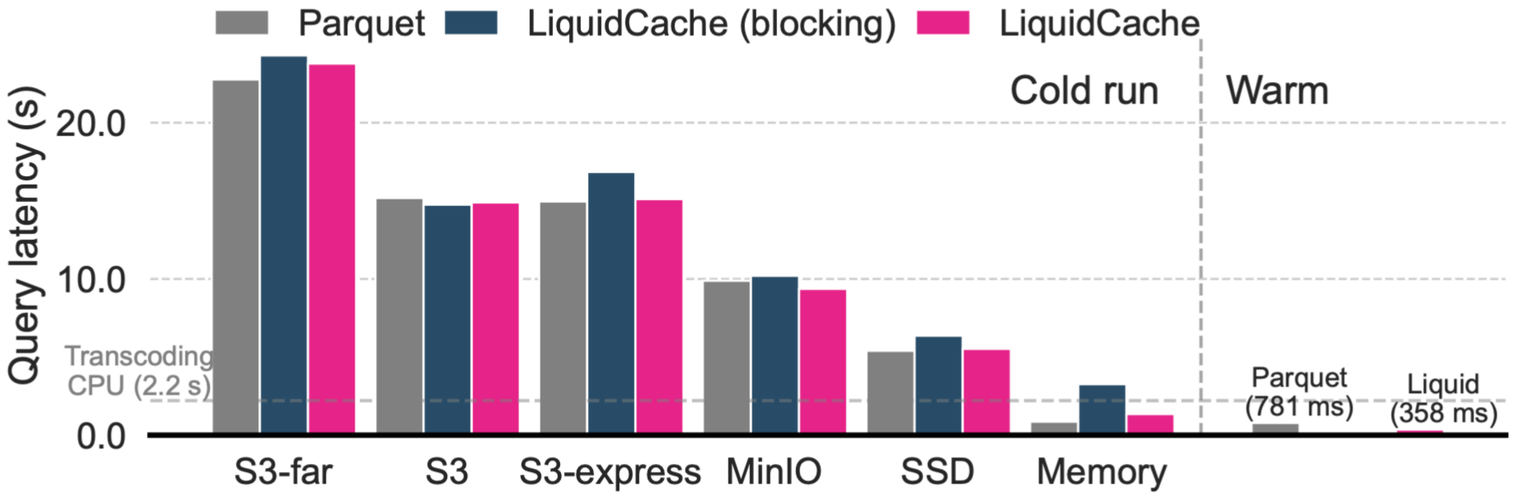
Background transcoding



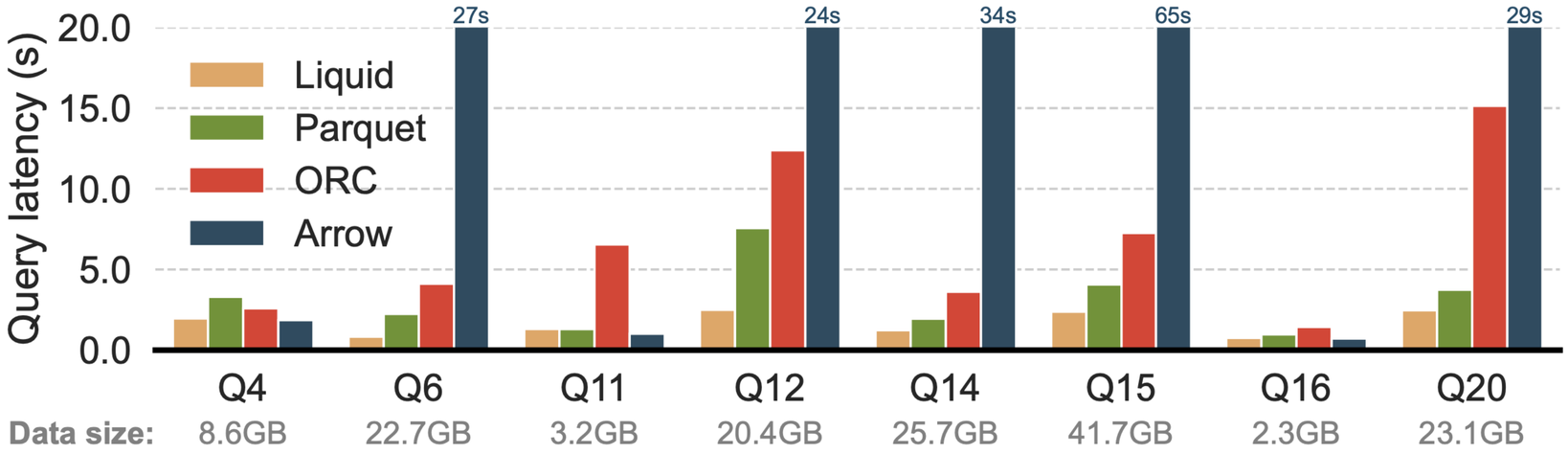
Asynchronous:
Transcode when less busy

LiquidCache

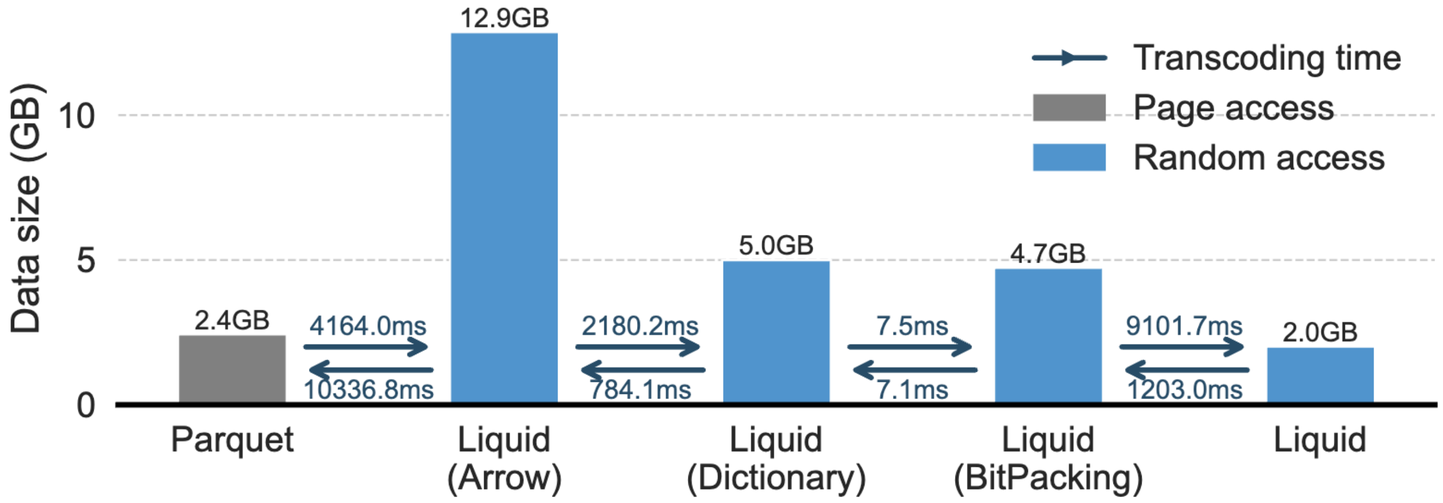
10x { cost | latency } reduction



Transcoding cost: hide by IO



With same memory: 10x lower latency



Compression ratio: comparable to Parquet

LiquidCache

10x { cost | latency } reduction

Predicate pushdown to reduce network traffic

Cache-specific format to stop the file-format war

LiquidCache: the missing cloud-native cache

https://github.com/XiangpengHao/liquid-cache