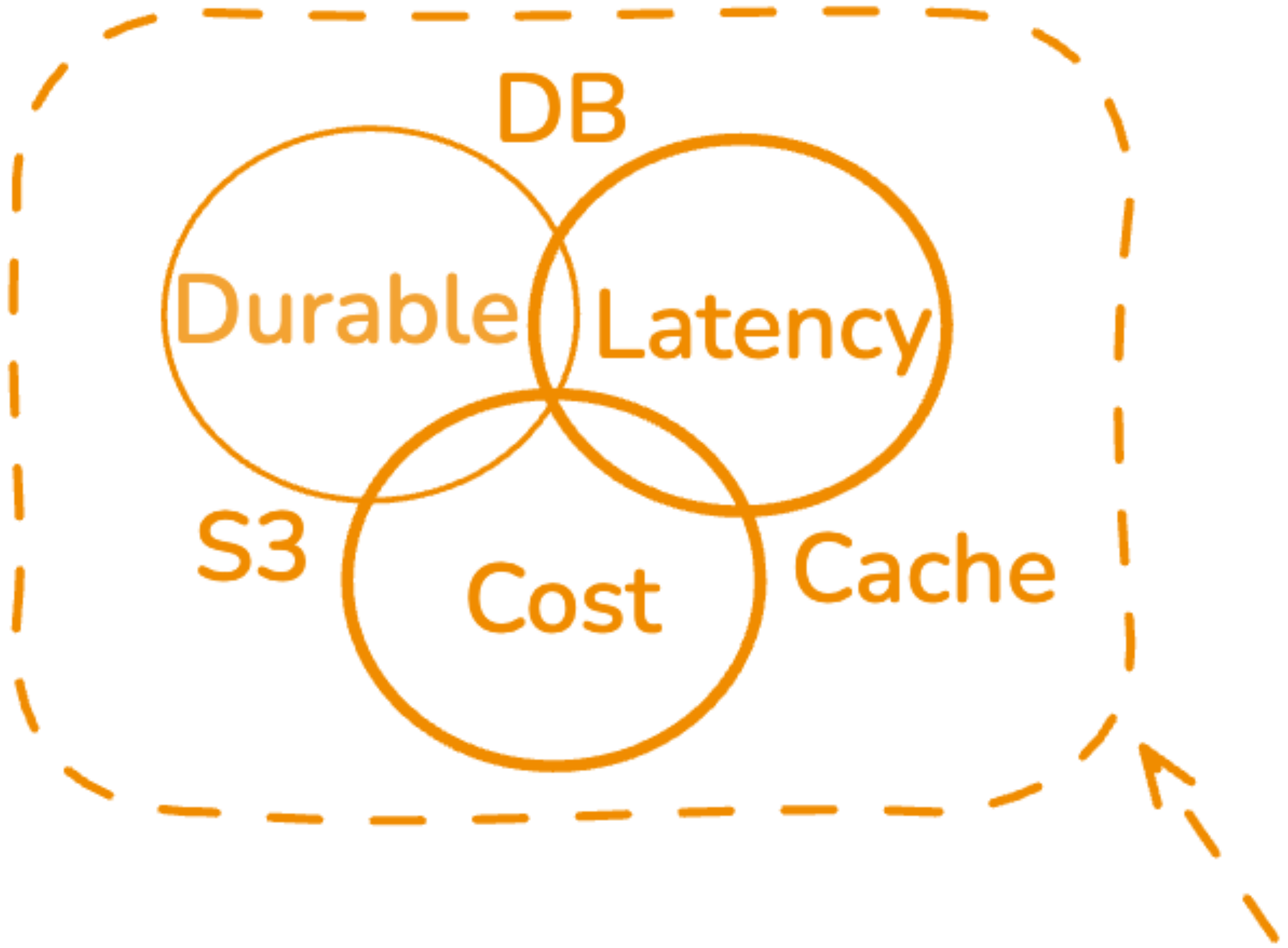
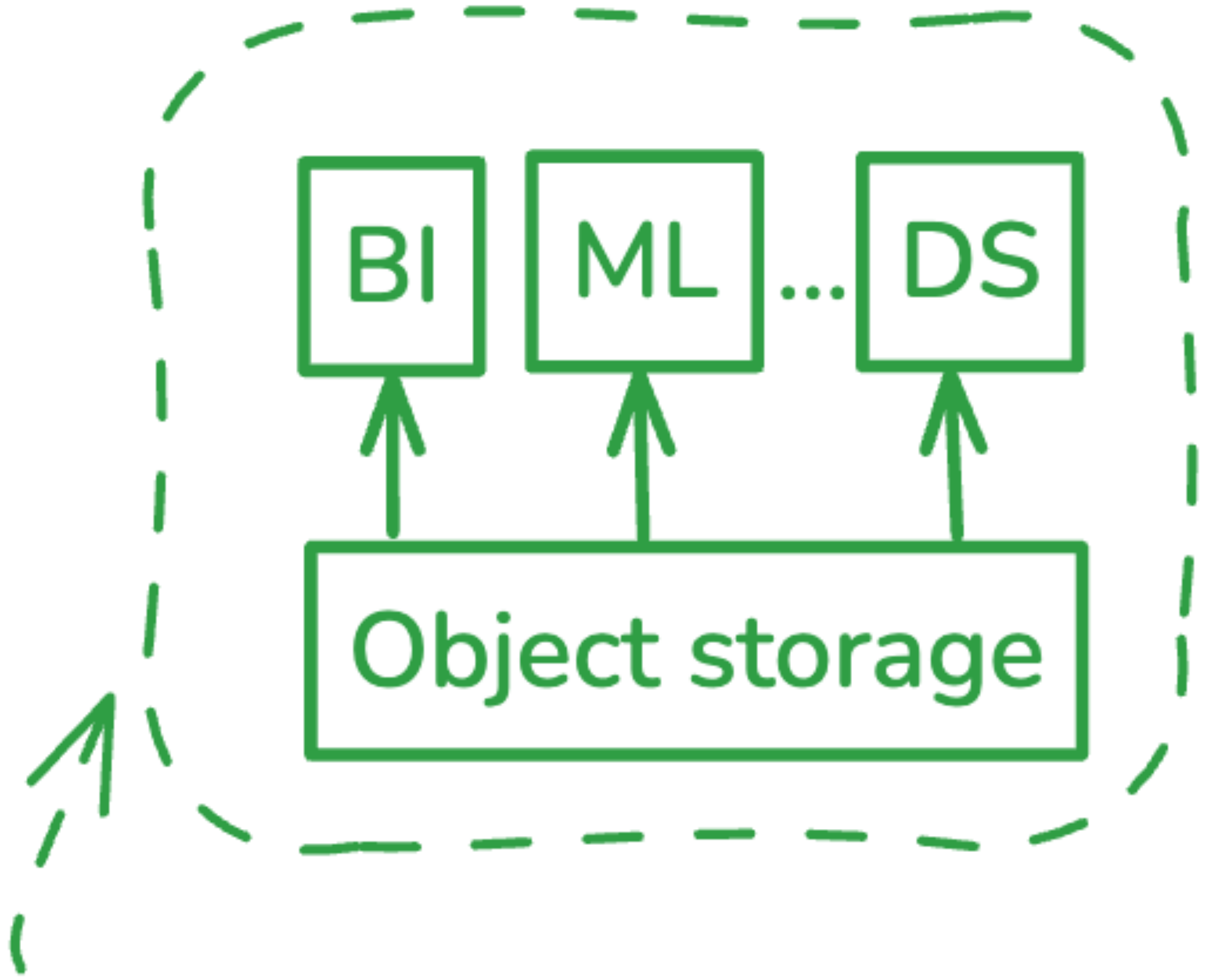
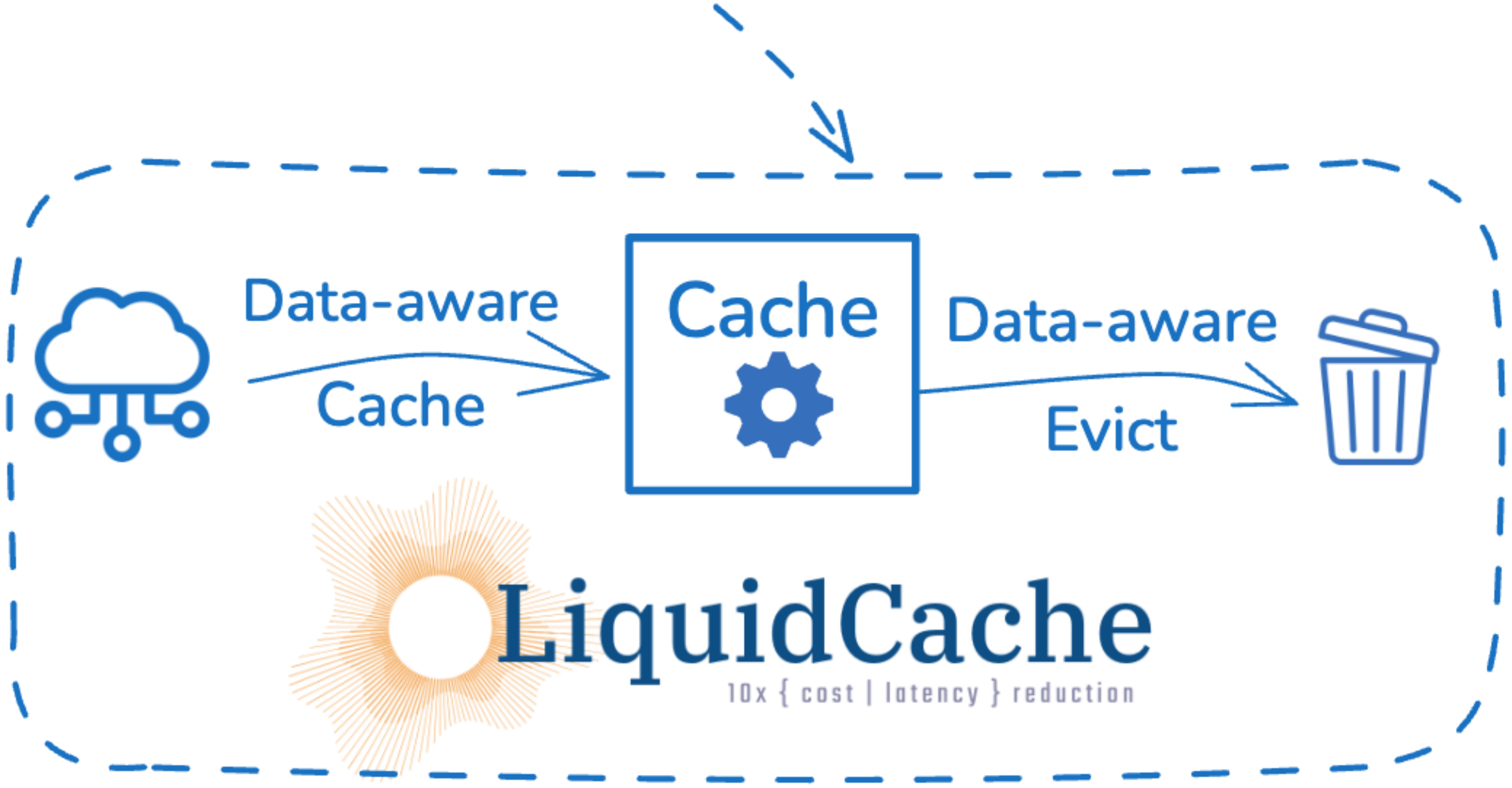




Data-Aware Caching for Cloud Analytics

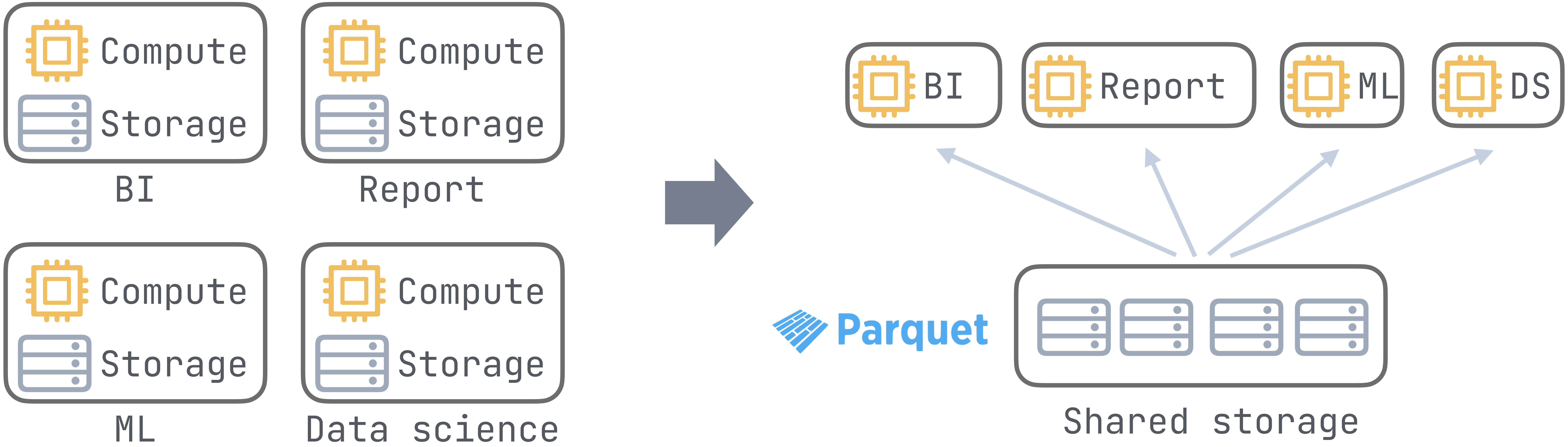


Xiangpeng Hao

Department of Computer Science
University of Wisconsin-Madison

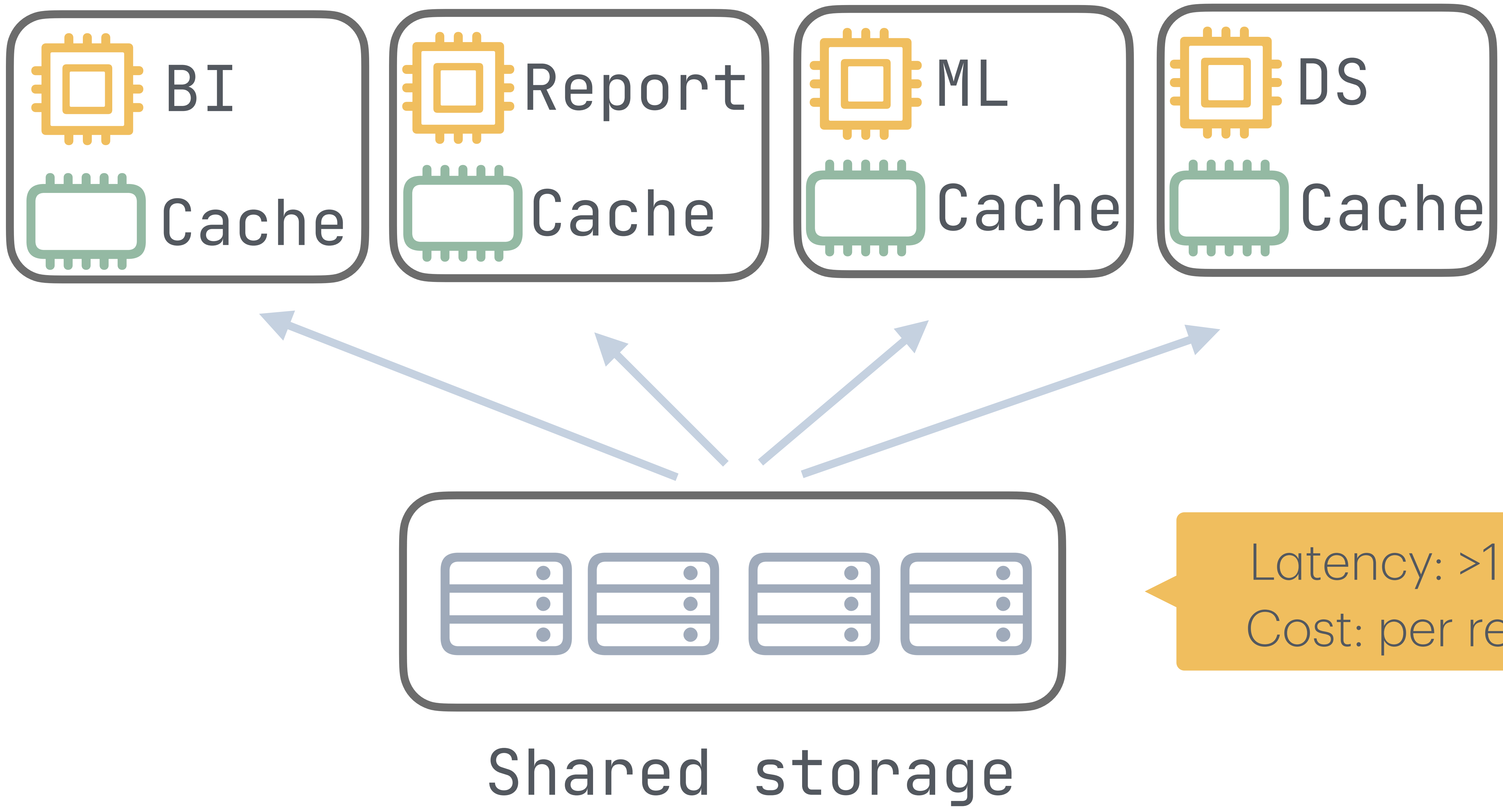
May 19th, 1 PM CDT
CS2310 or Zoom

On premise → cloud (2010-2020)



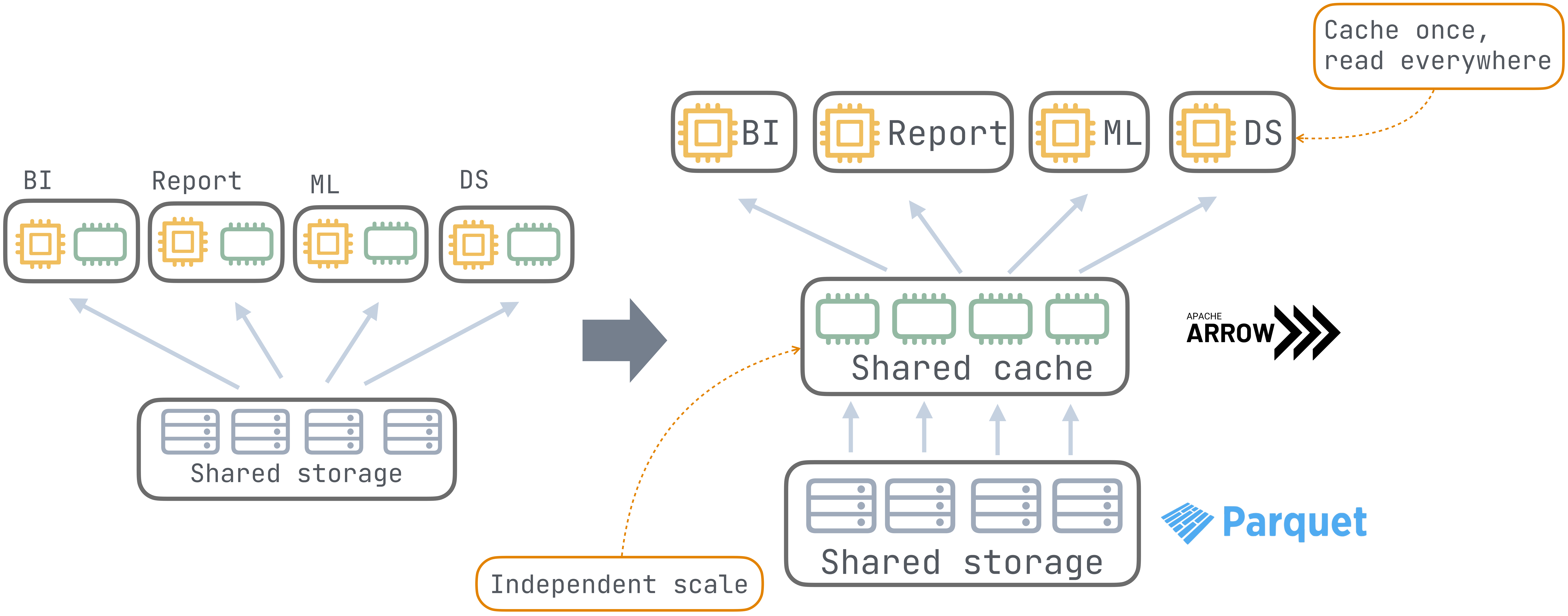
SAP databricks snowflake influxdata®

Every system has a cache (2020-2025)

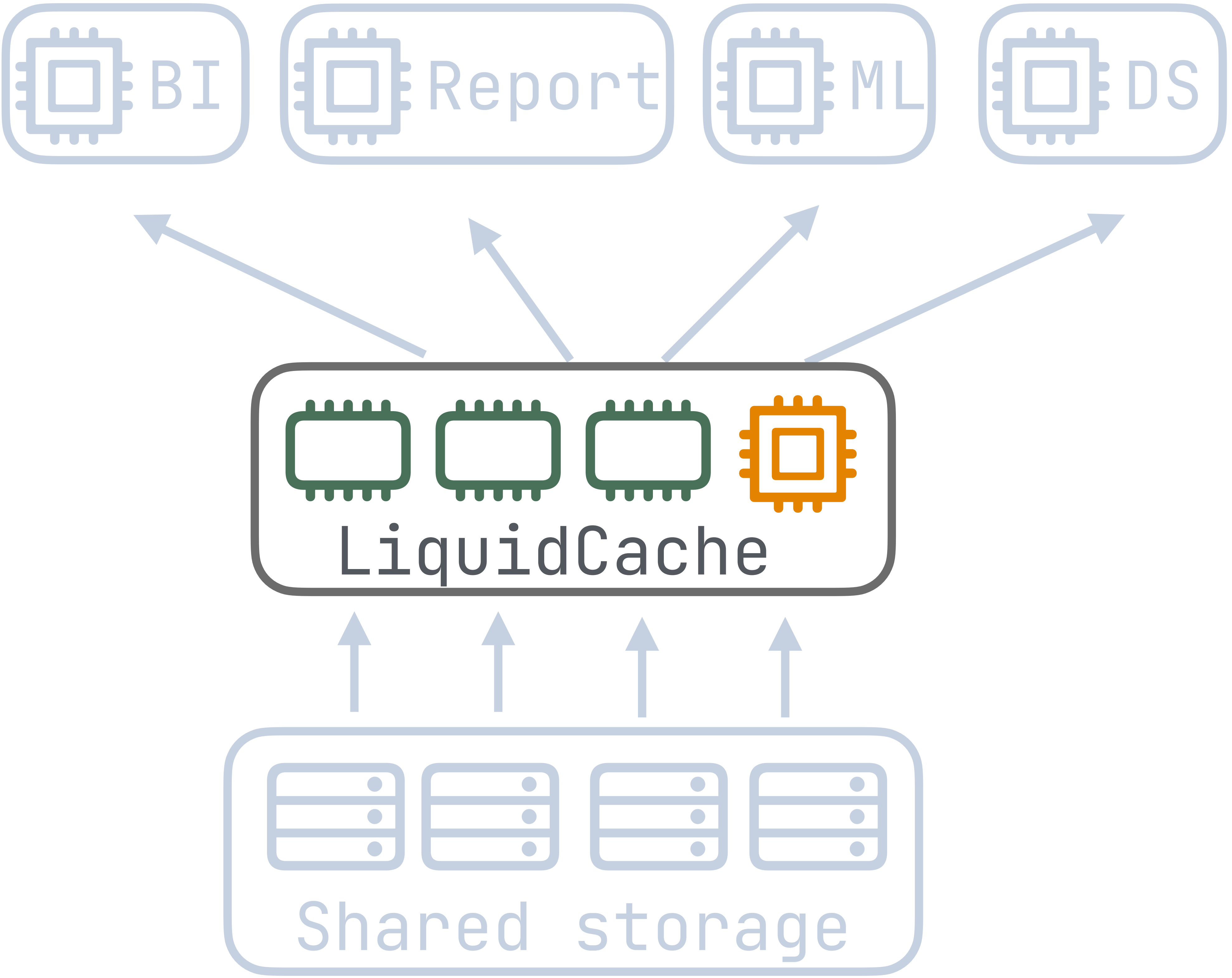


Latency: >100ms
Cost: per request

Vision: shared cache (2025+)



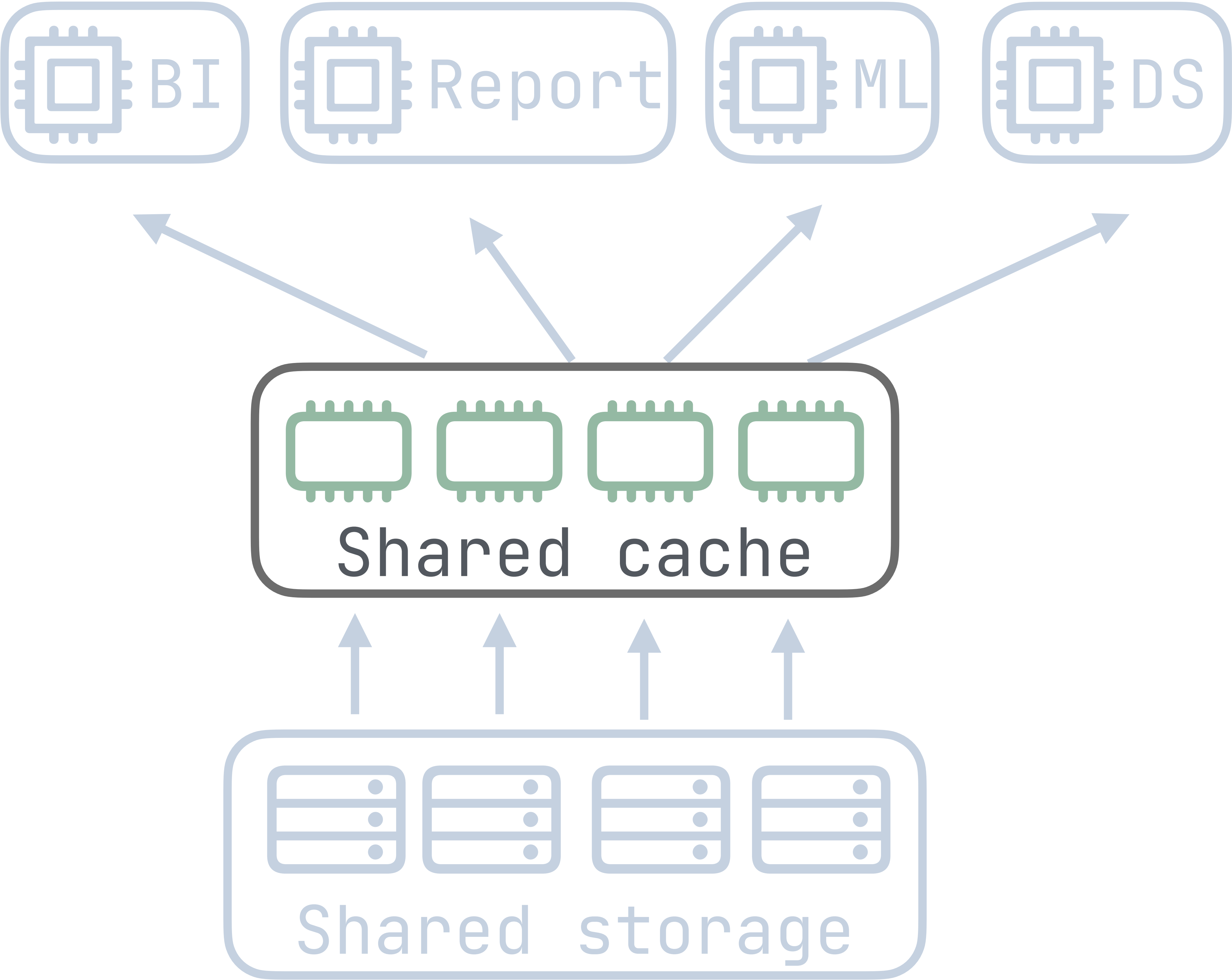
Thesis goal: LiquidCache



Thesis goal:

Design a cost-effective shared cache system by combining compute and data, while preserving ecosystem compatibility.

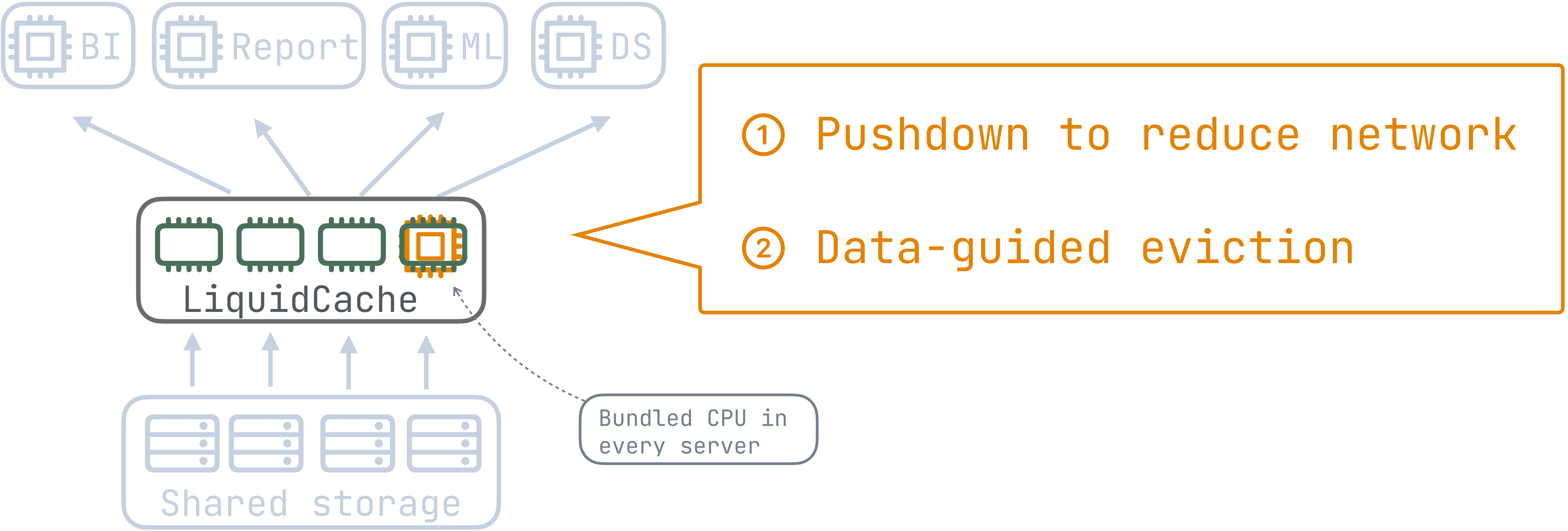
First attempt: byte cache



Challenges:

1. Network bottleneck
2. Inefficient cache eviction

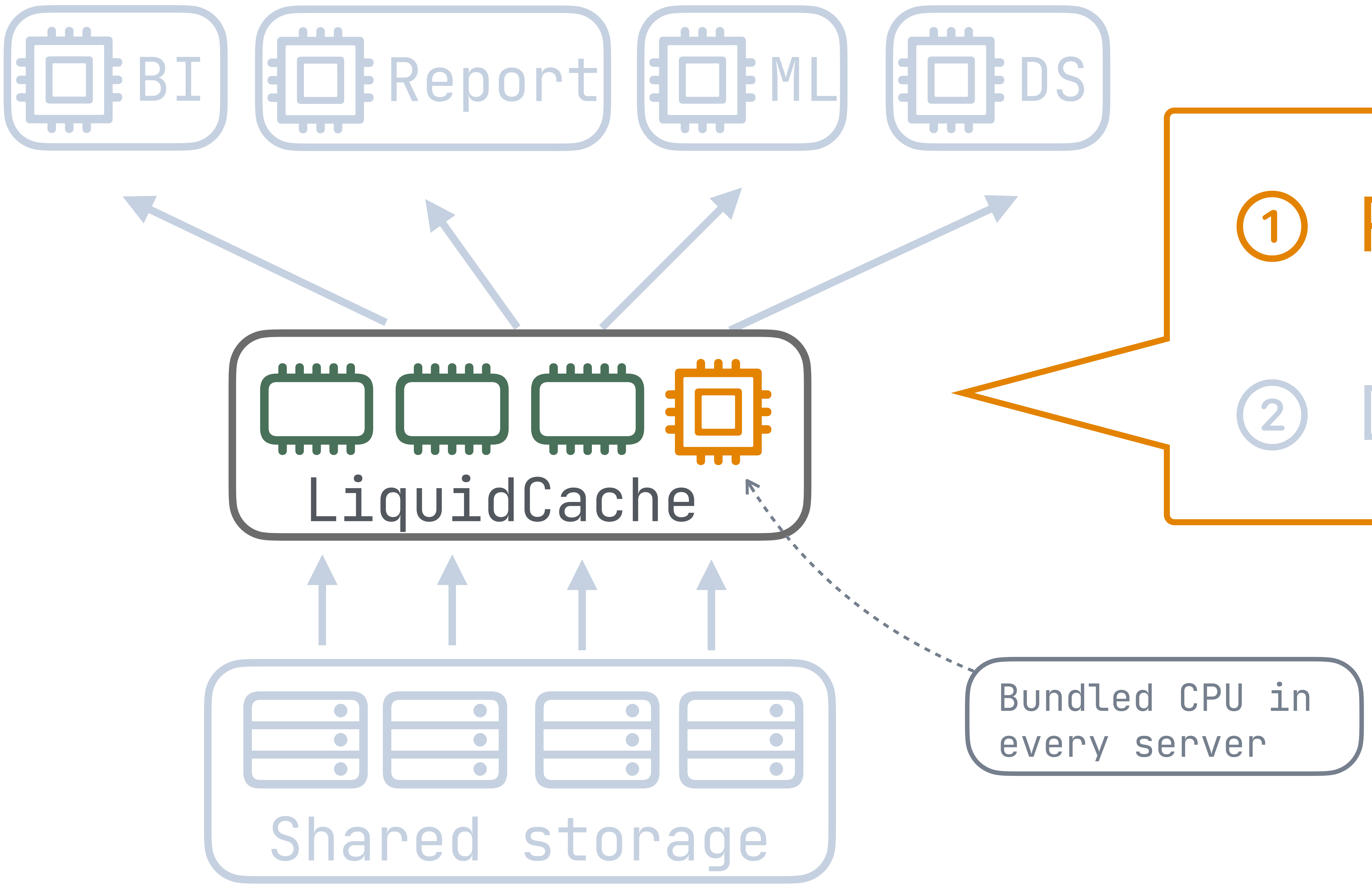
LiquidCache = compute + data



① Pushdown to reduce network

② Data-guided eviction

LiquidCache = compute + data



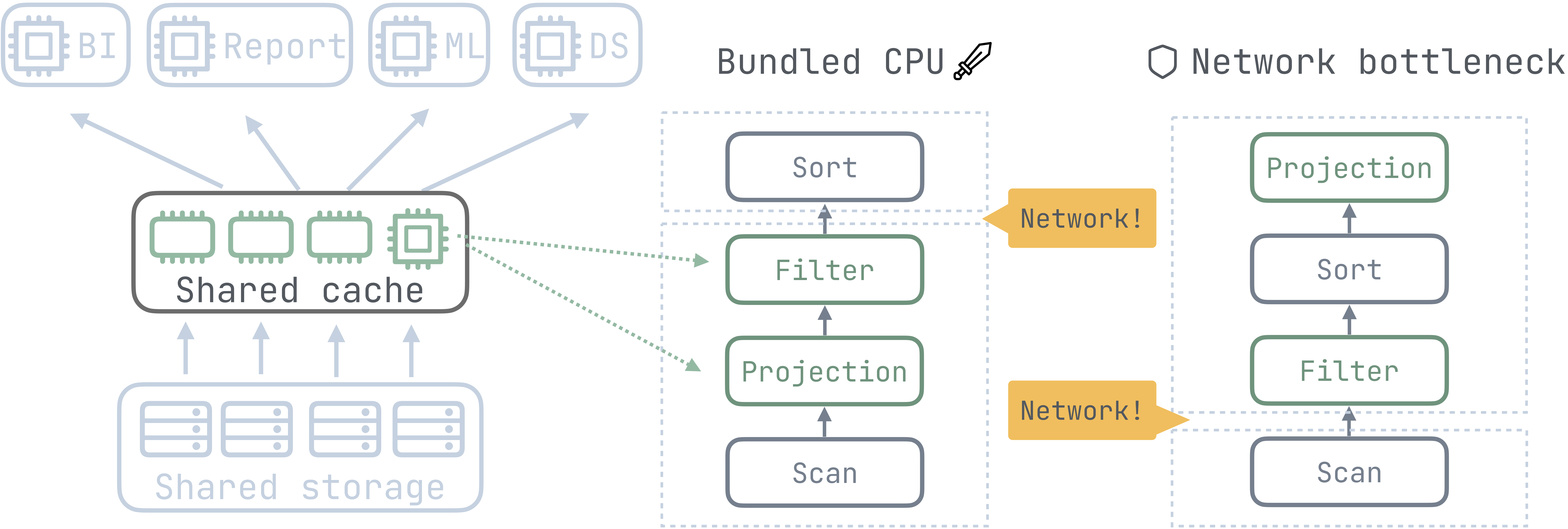
Bundled CPU in every server

VLDB 2025 (hopefully)

① Pushdown to reduce network

② Data-guided eviction

Pushdown to reduce network



BI

Report

ML

DS

Shared cache

Shared storage

Bundled CPU

Sort

Filter

Projection

Scan

Network!

Network!

Network bottleneck

Projection

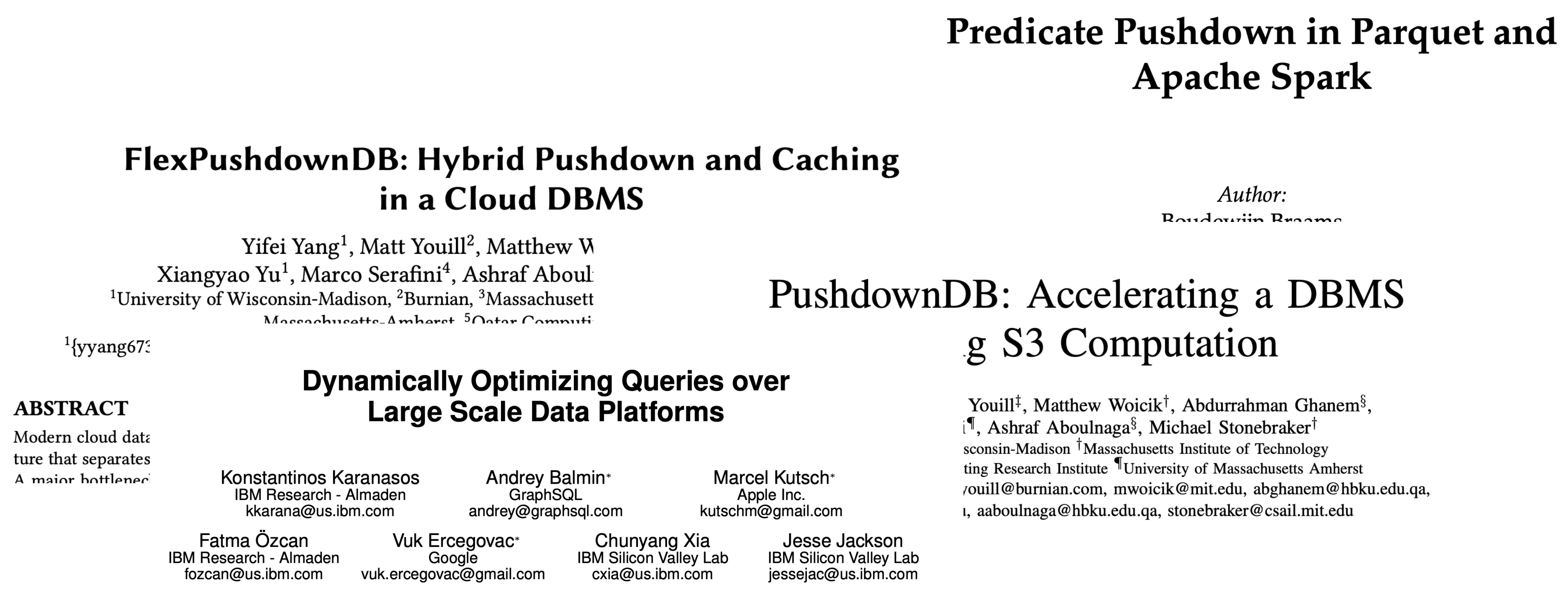
Sort

Filter

Scan

Pushdown overwhelms cache CPU

aws MINIO Azure databricks



Predicate Pushdown in Parquet and Apache Spark

Author:
Boudewijn Braams

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
Massachusetts-Amherst, [5]Qatar Computi

[1]{yyang67

ABSTRACT

Modern cloud data
ture that separates
A major bottlenec

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

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

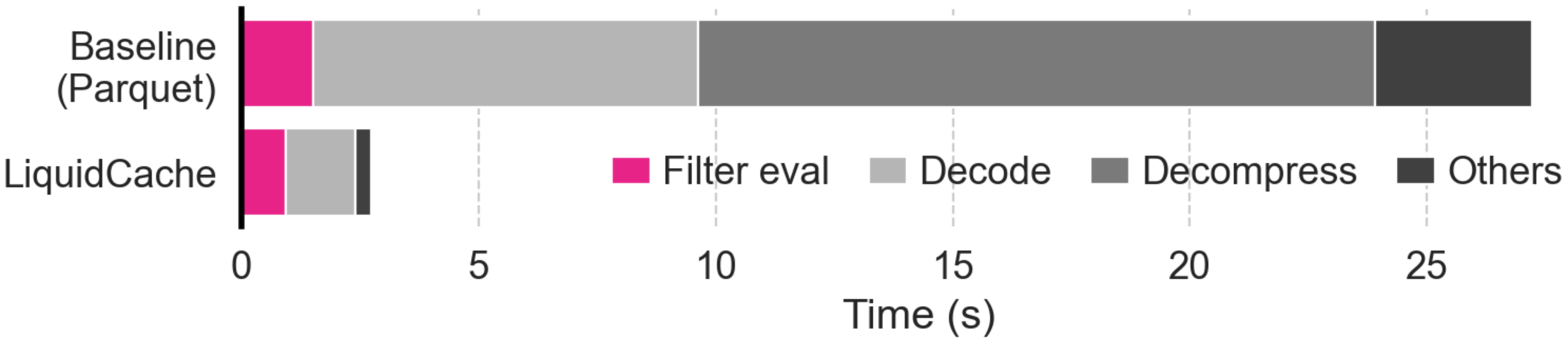
Vuk Ercegovac*
Google
vuk.ercegovac@gmail.com

Chunyang Xia
IBM Silicon Valley Lab
cxia@us.ibm.com

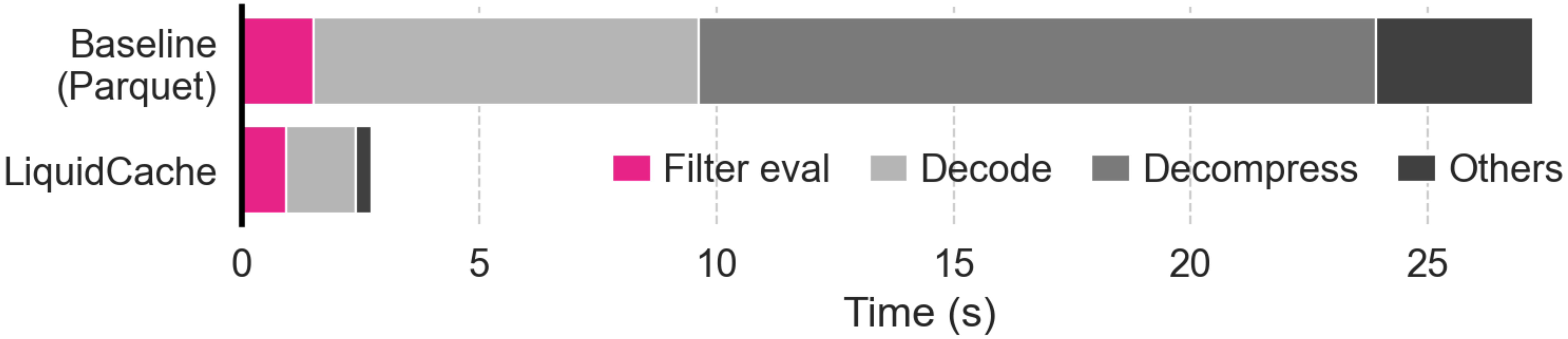
Jesse Jackson
IBM Silicon Valley Lab
jessejac@us.ibm.com

Previously believed bottleneck:
Filter evaluation

Our findings:
Data decoding

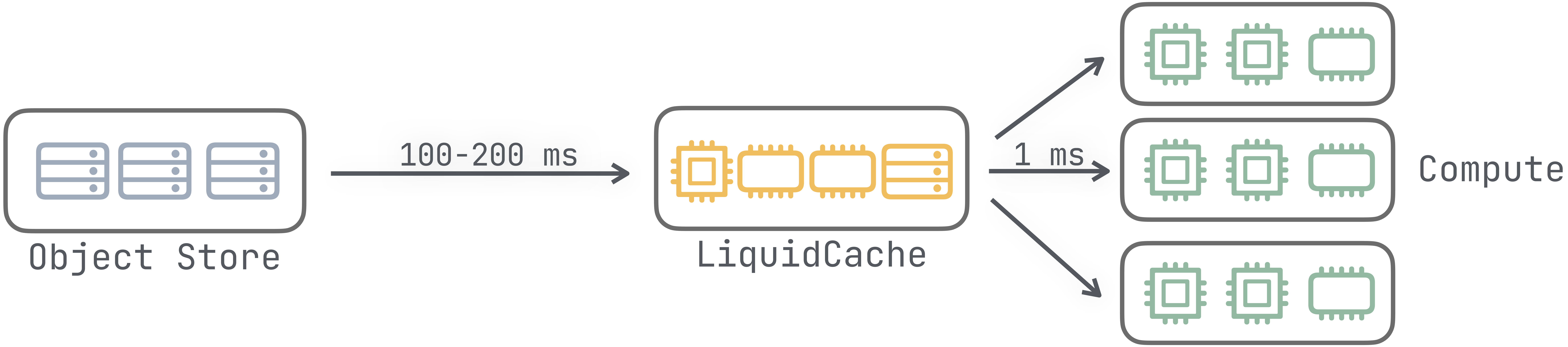


LiquidCache

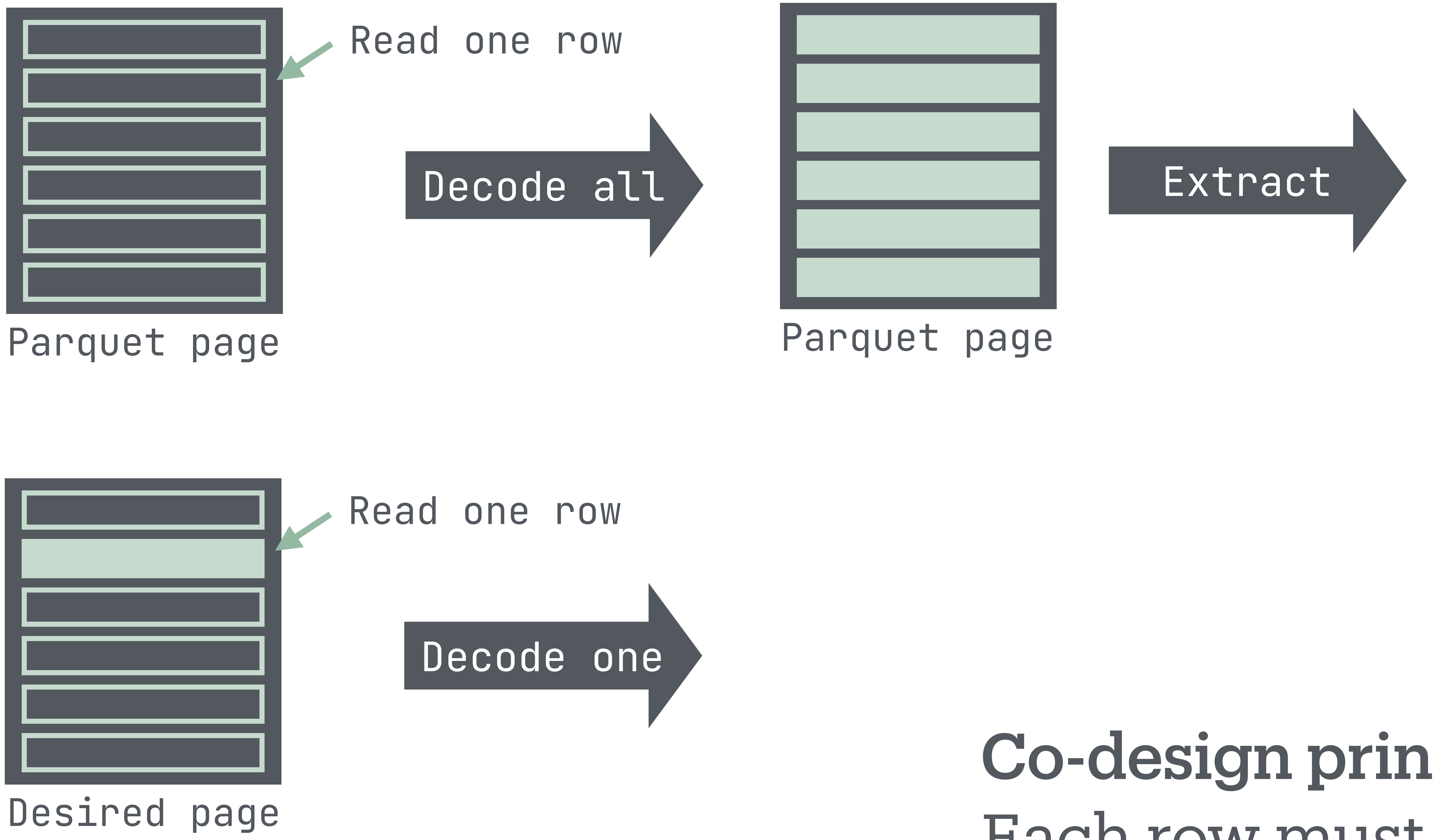


Part 1:
co-designed format to skip decoding

Part 2:
progressive, selective, asynchronous transcoding

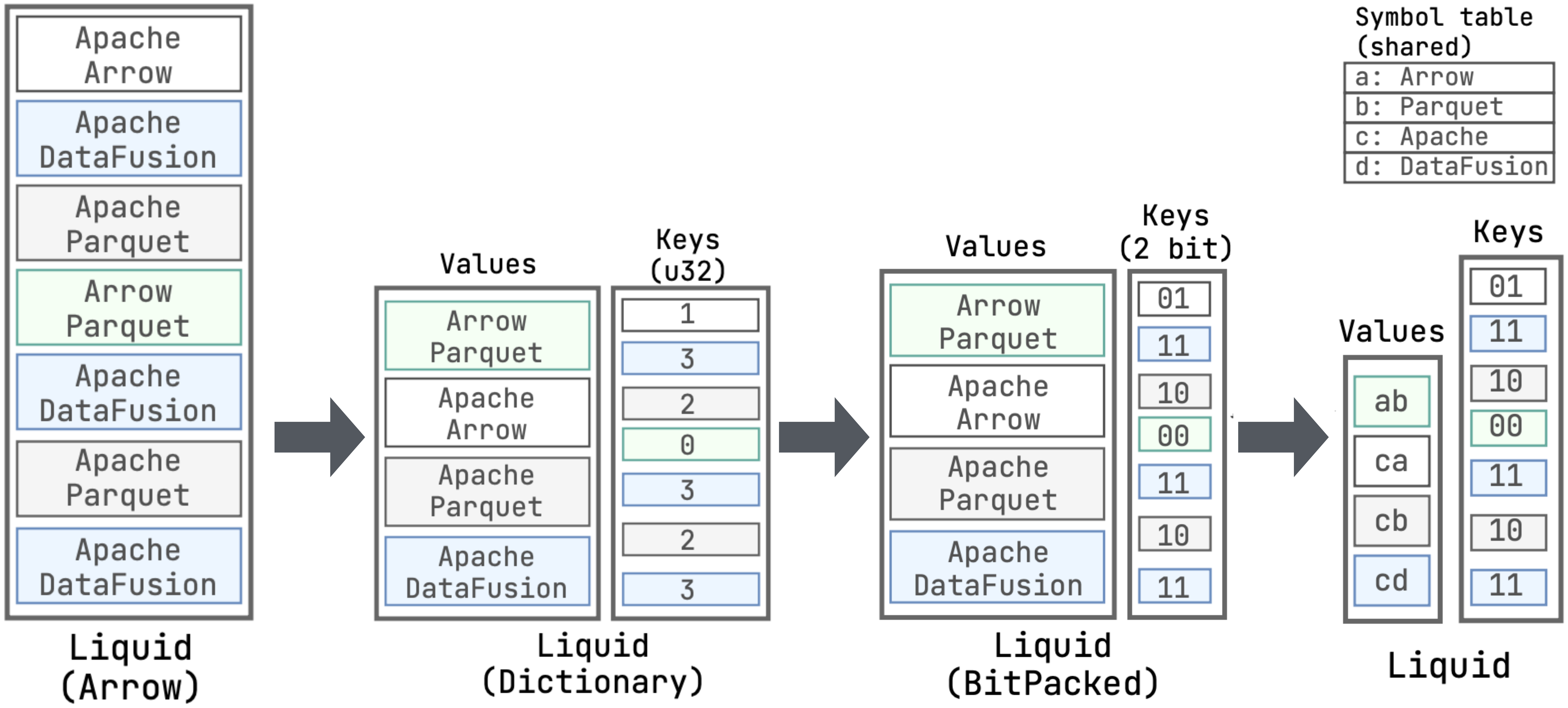


Part 1:
co-designed format to skip decoding



Co-design principle:
Each row must be independently decodable

Each row must be independently decodable (string example)



Liquid (Arrow)

Apache Arrow

Apache DataFusion

Apache Parquet

Arrow Parquet

Apache DataFusion

Apache Parquet

Apache DataFusion

Liquid (Dictionary)

Values	Keys (u32)
Arrow Parquet	1
Apache Arrow	3
Apache Parquet	2
Apache DataFusion	0
	3
	2
	3

Liquid (BitPacked)

Values	Keys (2 bit)
Arrow Parquet	01
Apache Arrow	11
Apache Parquet	10
Apache DataFusion	00
	11
	10
	11

Liquid

Symbol table (shared)

a: Arrow
b: Parquet
c: Apache
d: DataFusion

Values	Keys
ab	01
ca	11
cb	10
cd	00
	11
	10
	11

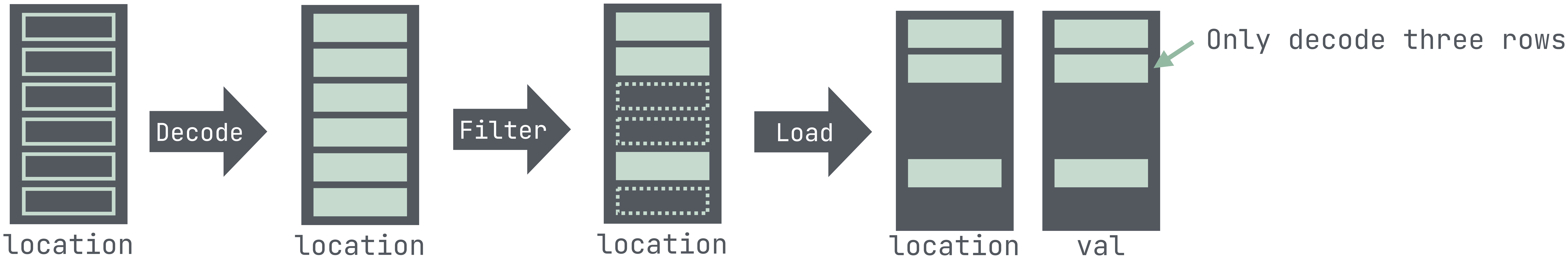
No general purpose compression

Leverages state-of-the-art encoding schemes

Carefully designed encoding/layout for each data types

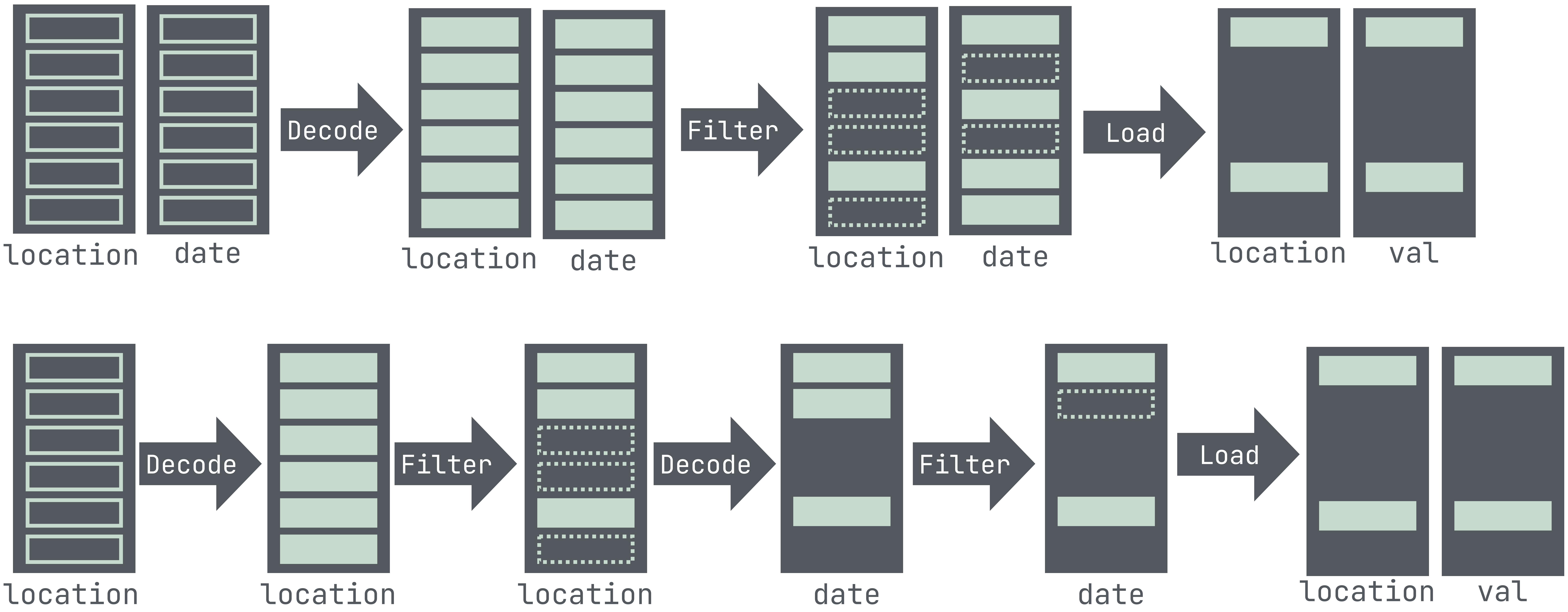
Co-design with filter pushdown (selective decoding)

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



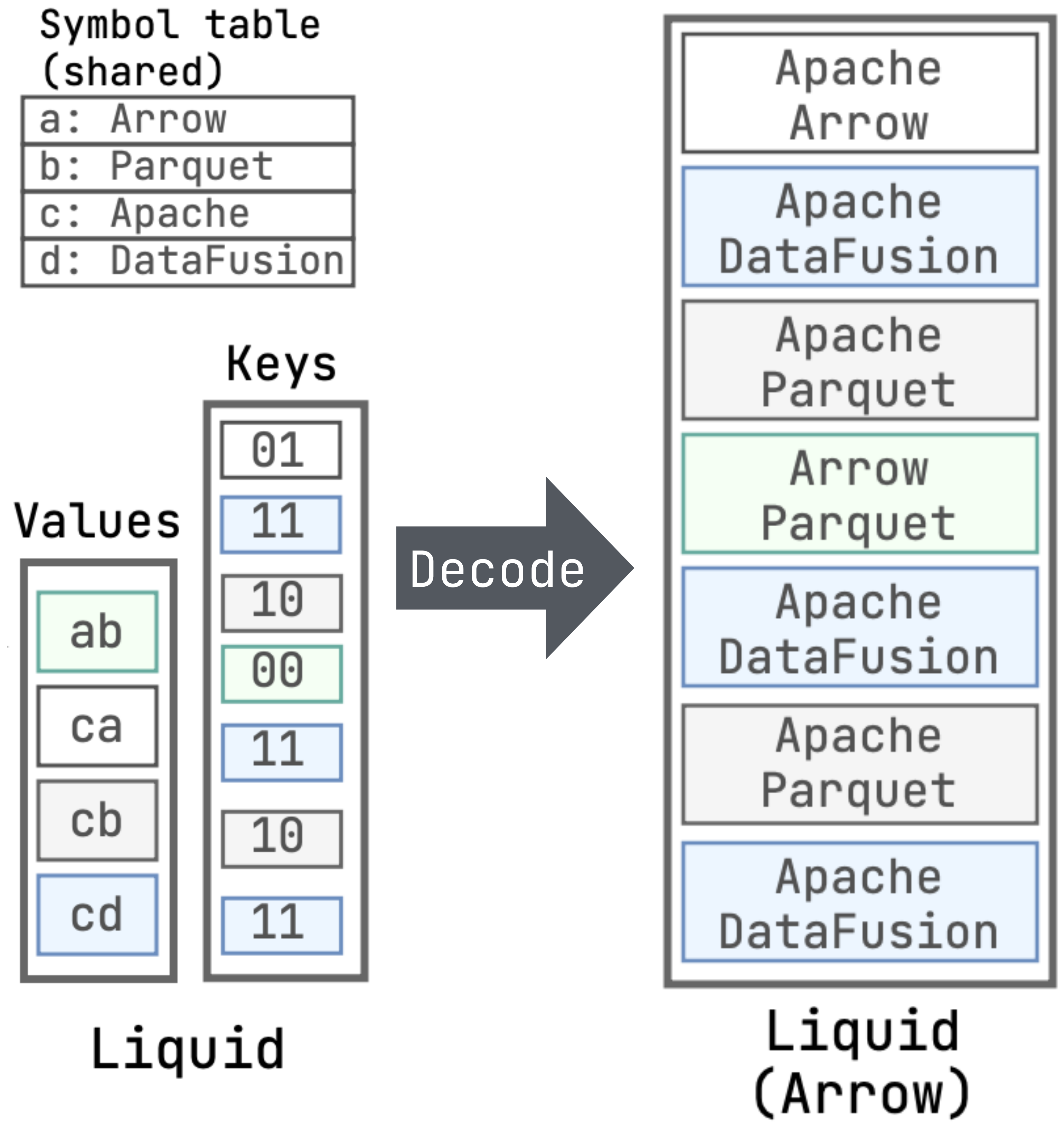
Co-design with filter pushdown (filter late materialization)

```sql
SELECT val, location
FROM sensor_data
WHERE location = 'office', date > '2025-03-21';
```

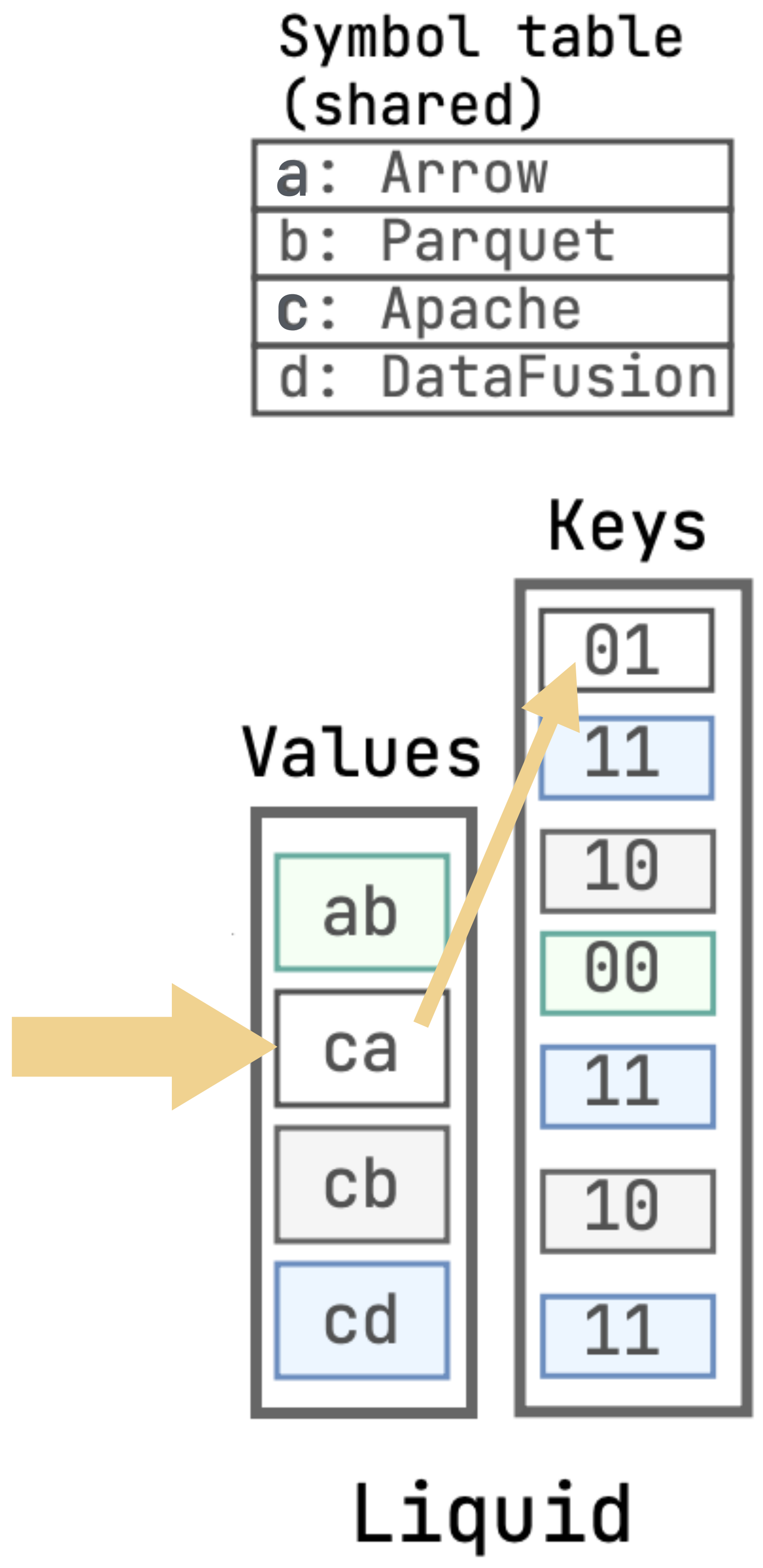


Co-design with filter pushdown (evaluate on encoded data)

```sql
SELECT val, location
FROM sensor_data
WHERE name = ‘Apache Arrow’
```



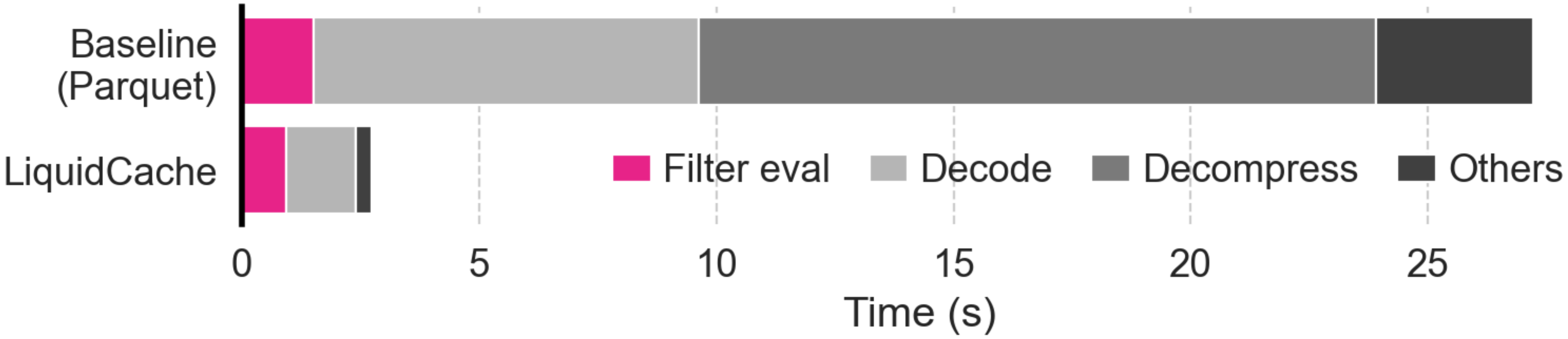
Baseline



Evaluate on encoded

In paper:
Evaluate on **partially** encoded data

LiquidCache

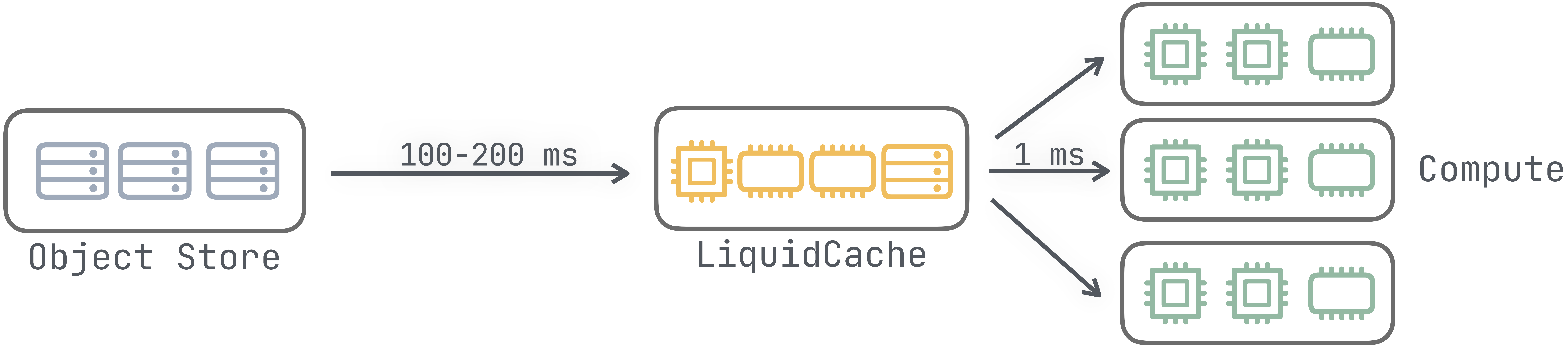


Part 1:

co-designed format to skip decoding

Part 2:

progressive, selective, asynchronous transcoding



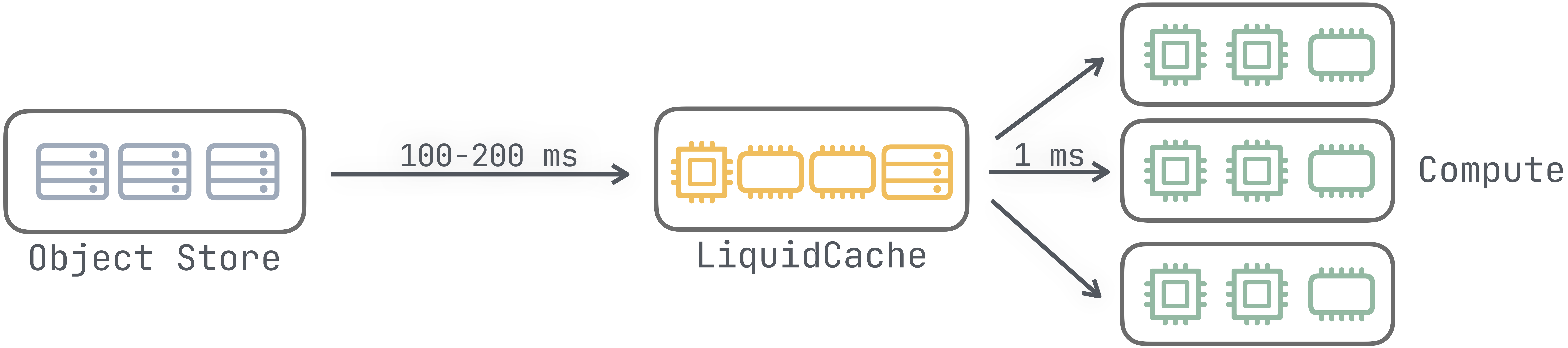
Part 2:
progressive, selective, asynchronous transcoding

Yet another file format?

Nimble LanceDB
Vortex ClickHouse
DuckDB

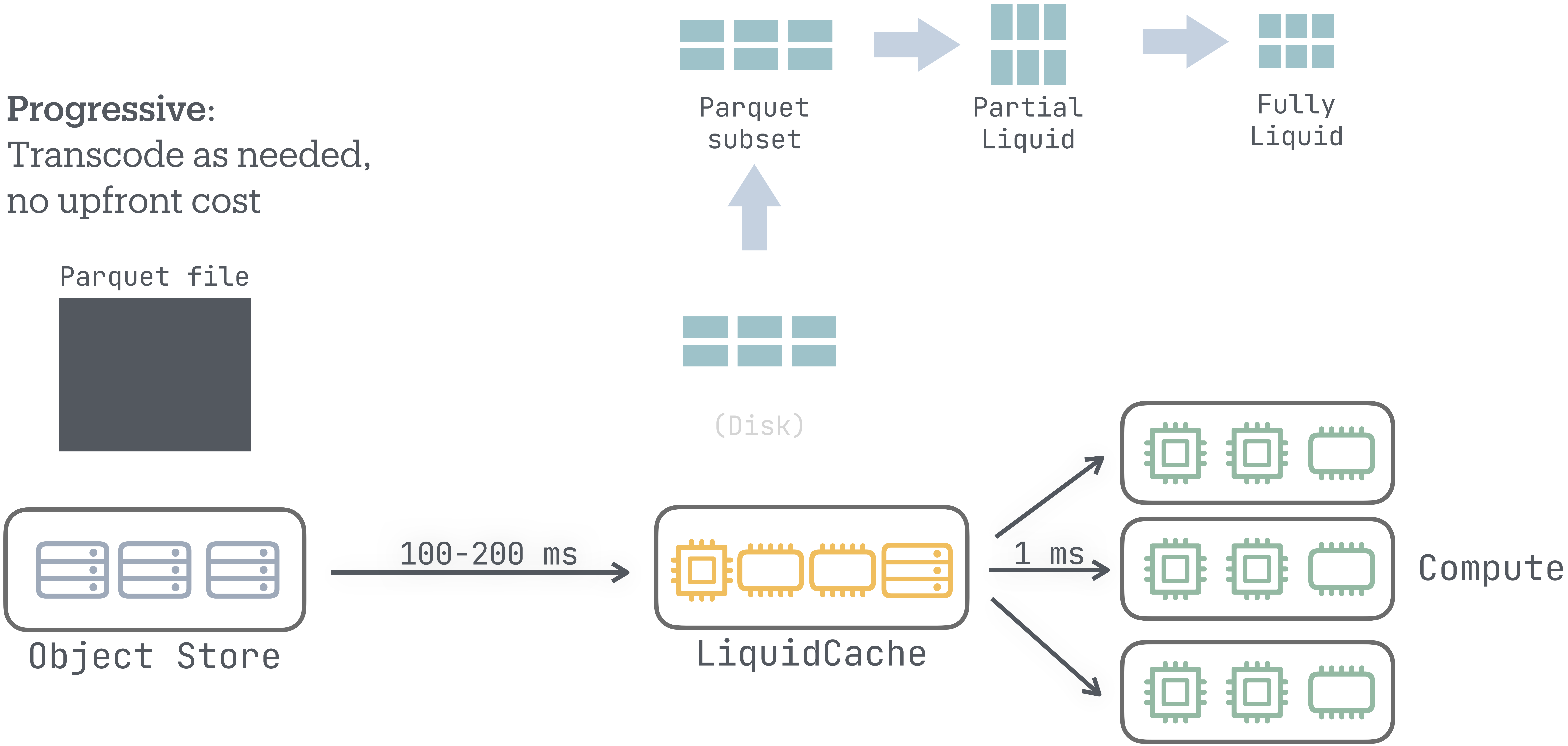
No!

Technical: they are not much different
Organizational: license, governance, ecosystem
LiquidCache: progressively bend the world

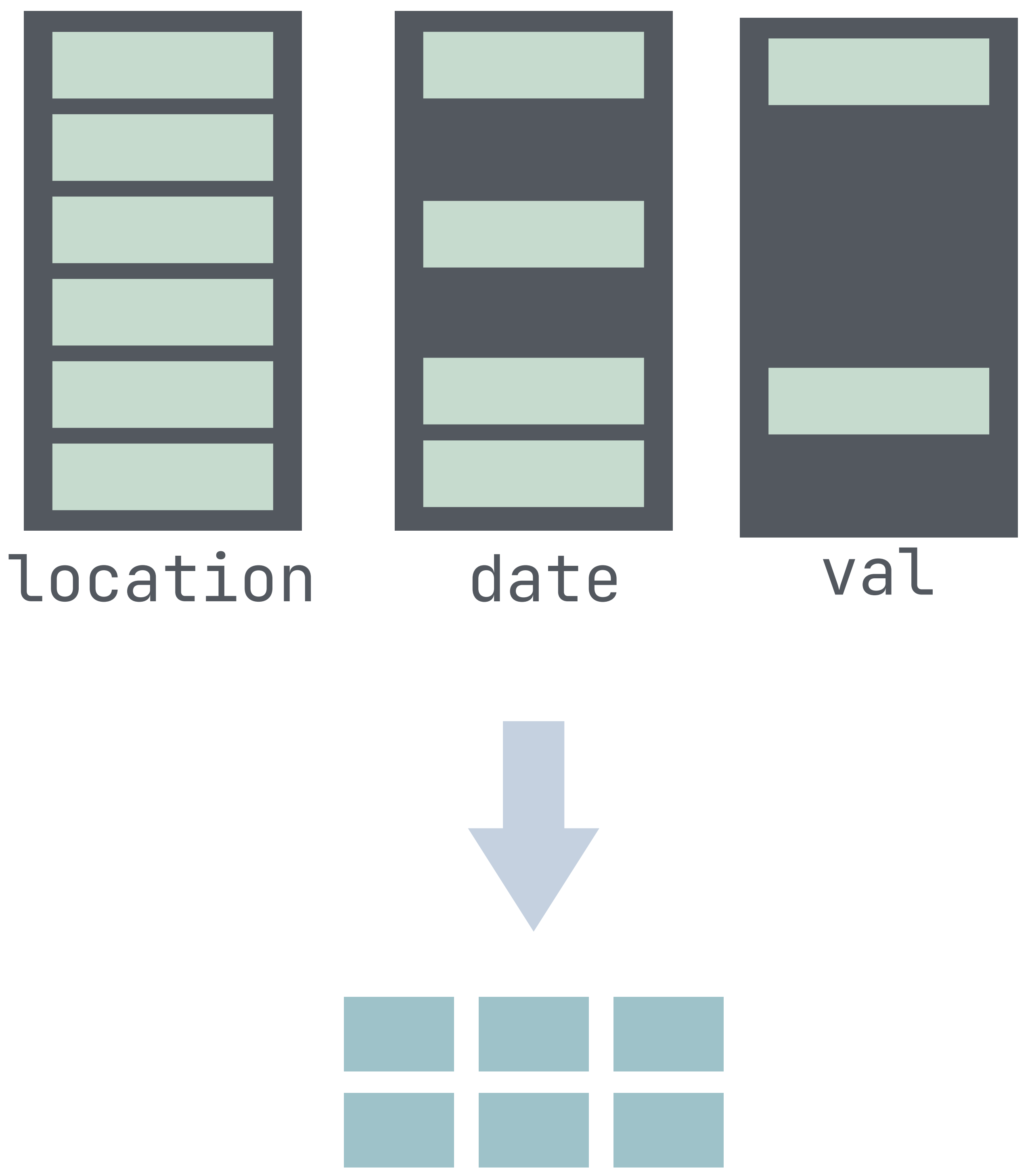


Progressive transcoding

Progressive:
Transcode as needed,
no upfront cost

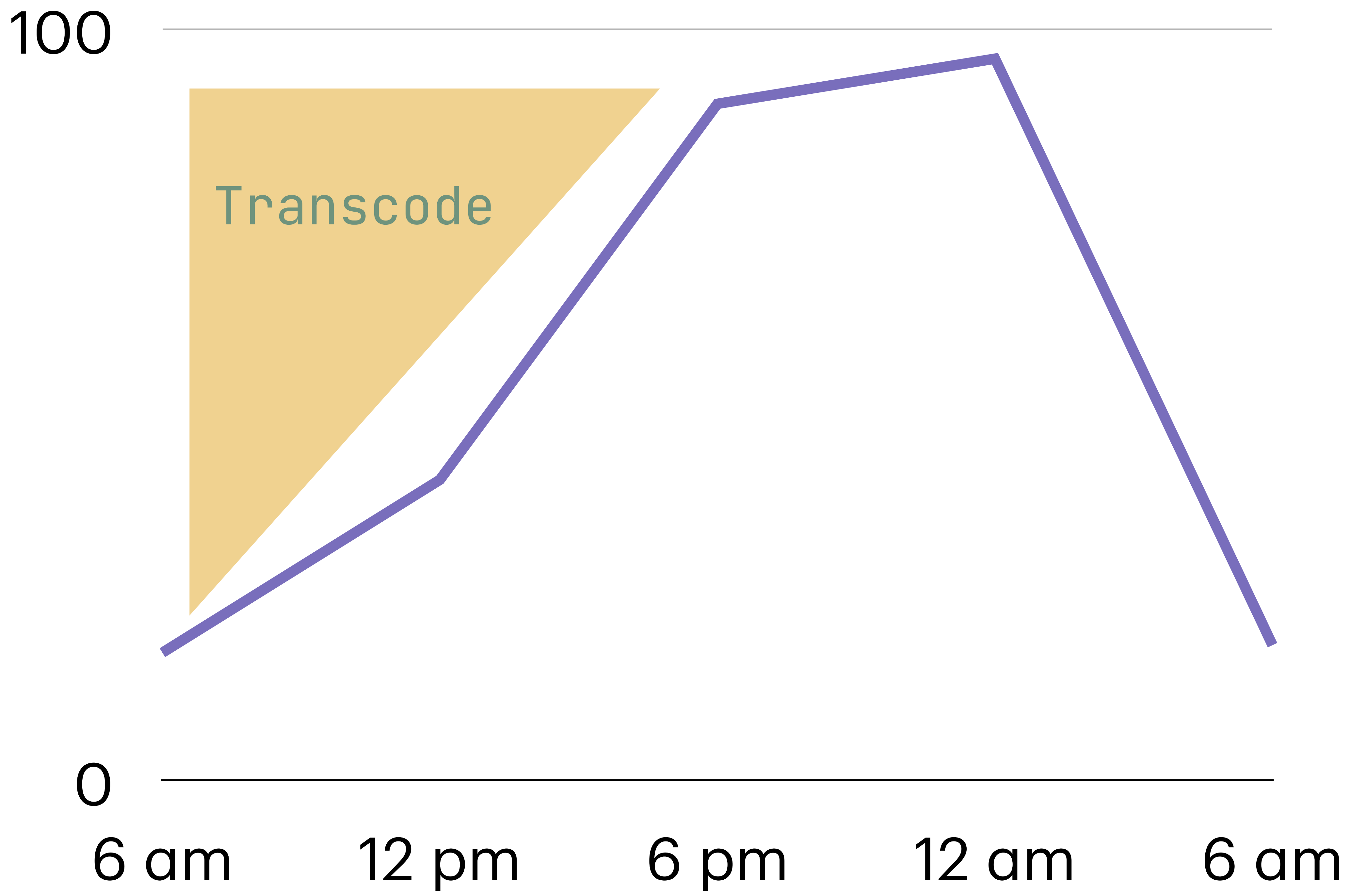


Selective transcoding



Selective:
Transcode only touched data

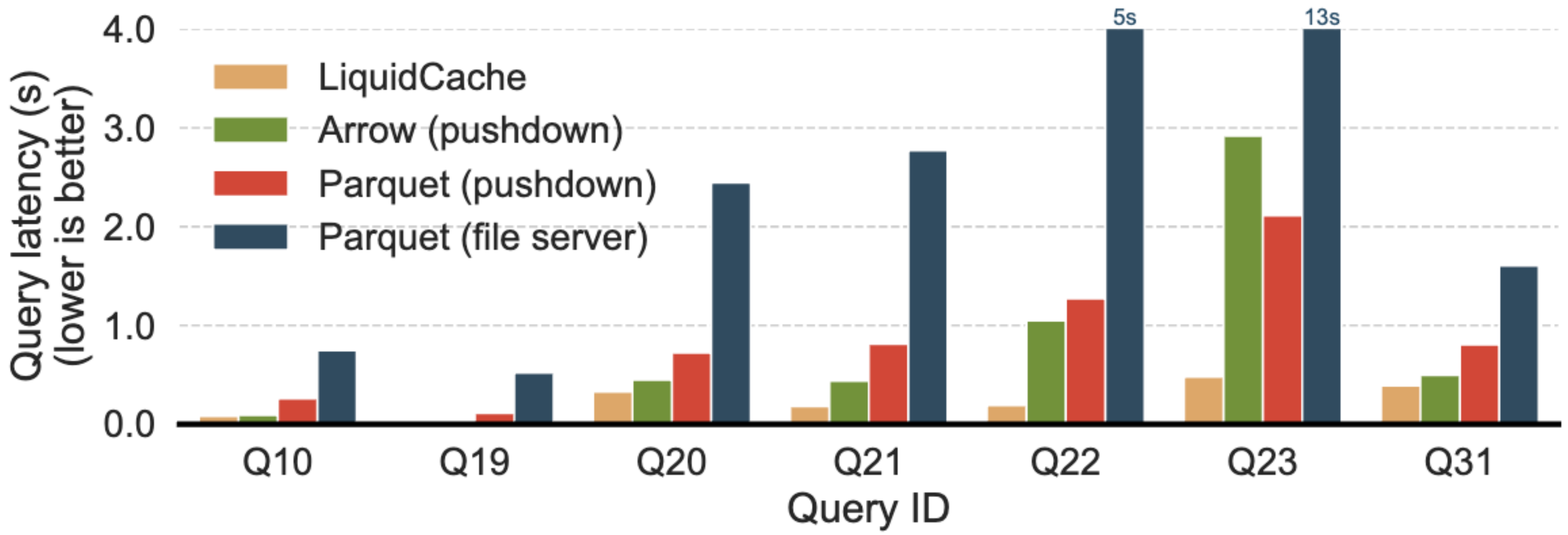
Asynchronous transcoding



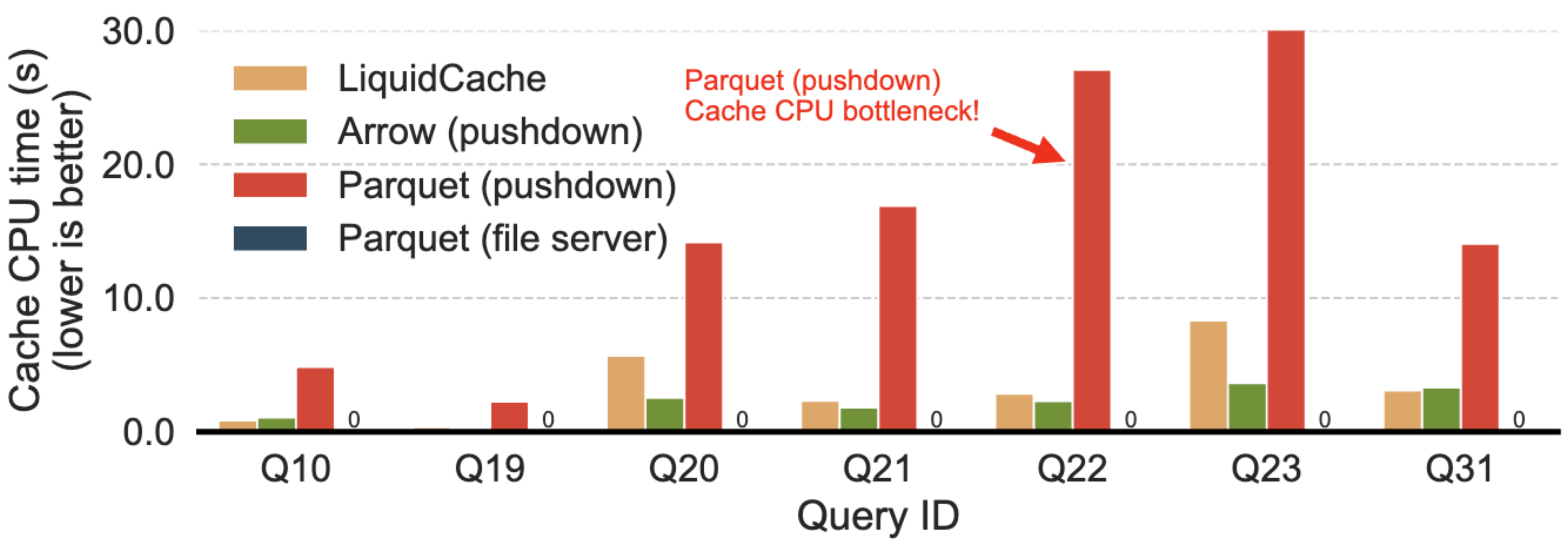
Asynchronous:
Transcode when less busy

LiquidCache

10x { cost | latency } reduction



With same memory: 10x lower latency

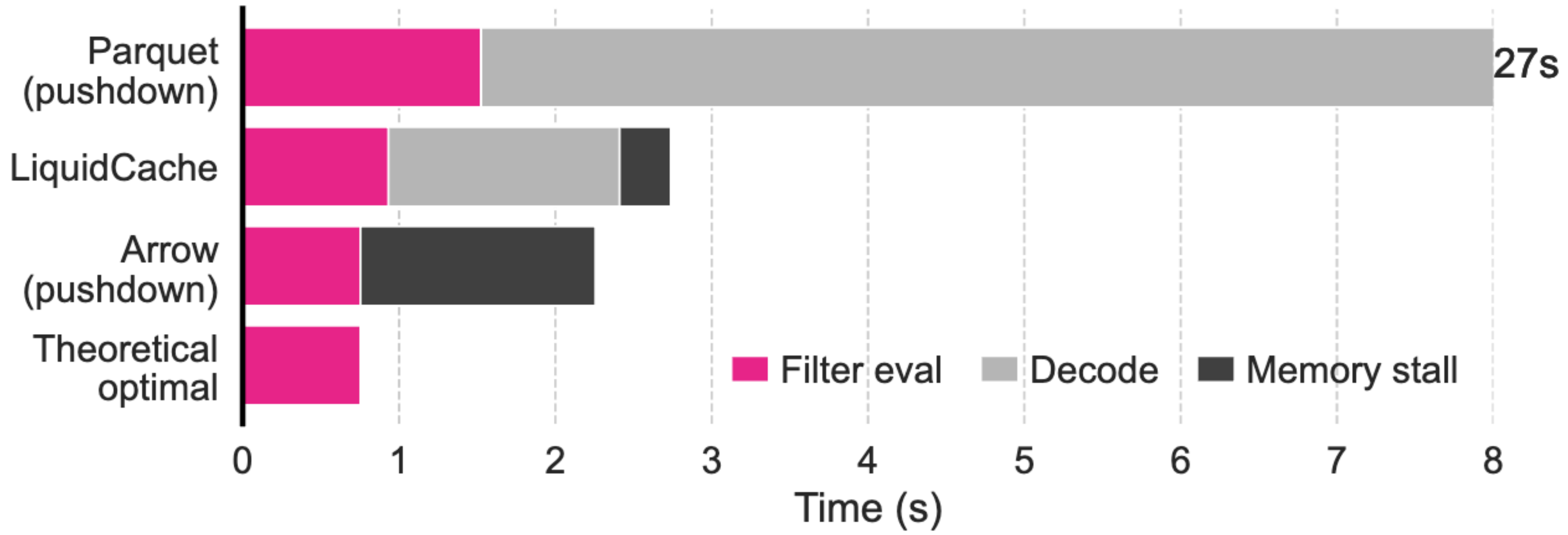


Parquet (pushdown)
Cache CPU bottleneck!

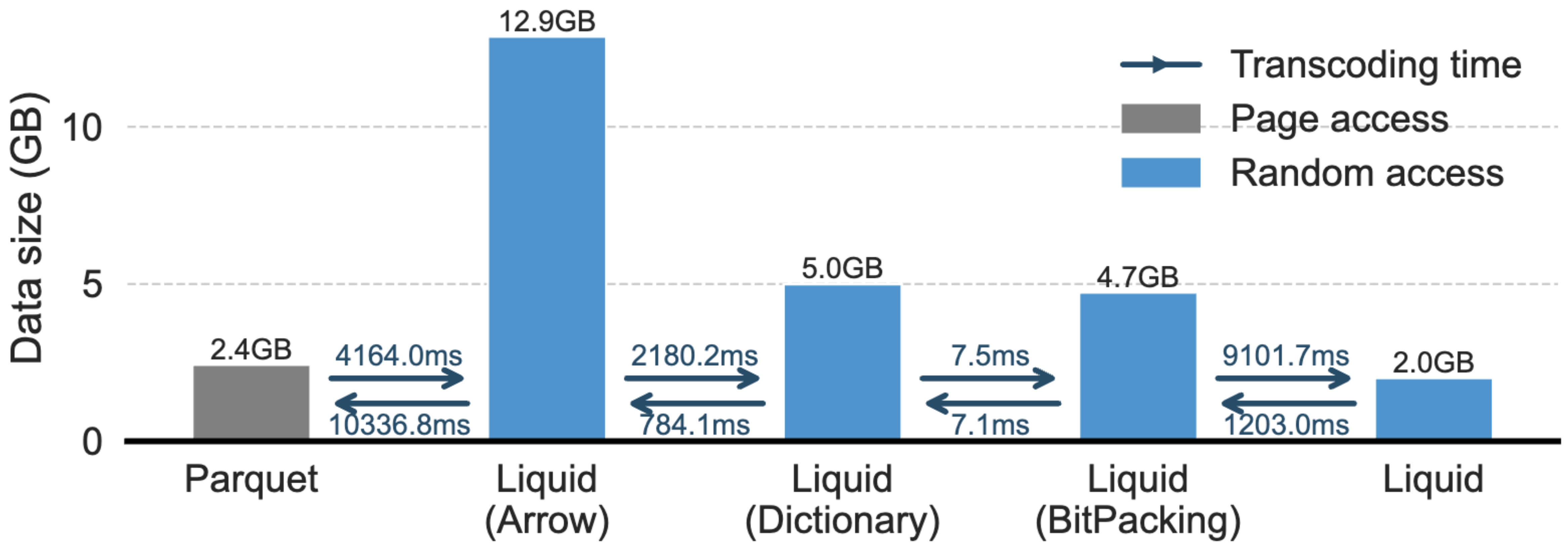
With same CPU: 10x lower CPU time

LiquidCache

10x { cost | latency } reduction



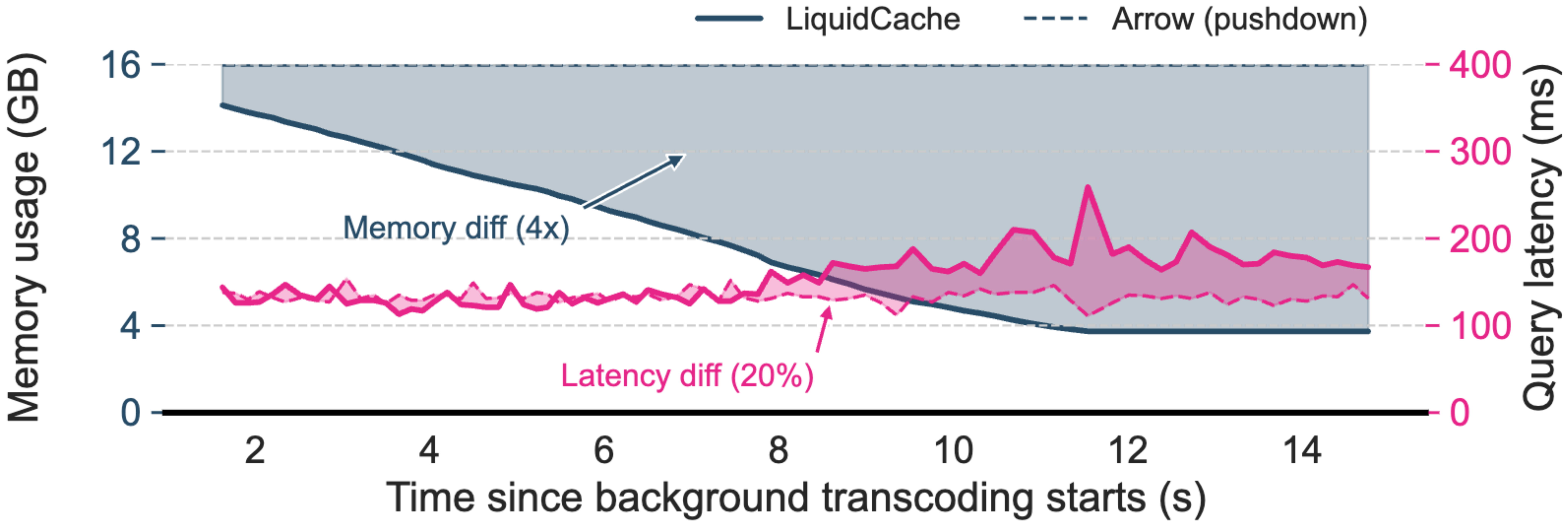
Decoding cost: close to theoretical optimal



Compression ratio: comparable to Parquet

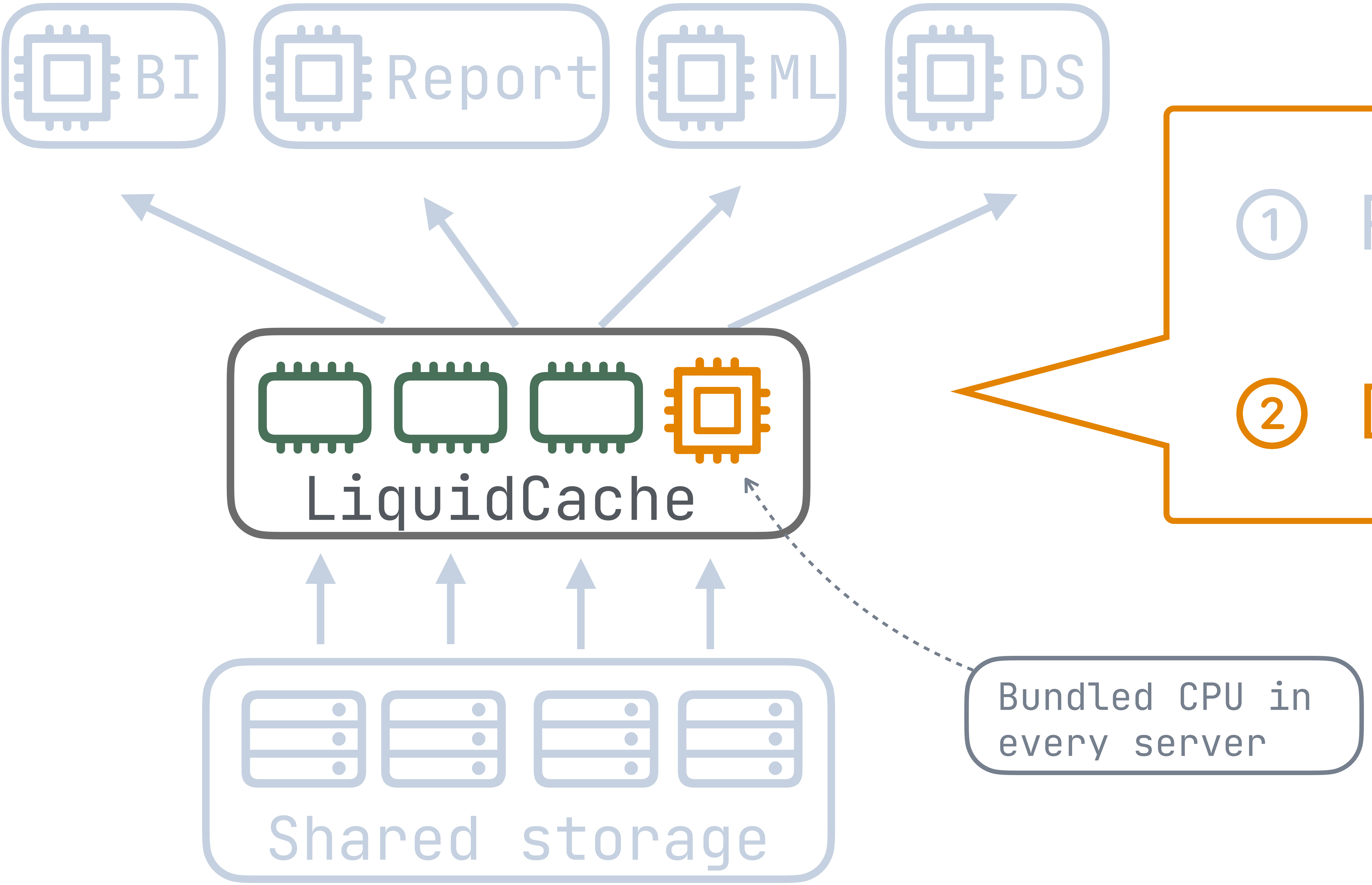
LiquidCache

10x { cost | latency } reduction



Transcoding cost: negligible, no latency spike

LiquidCache = compute + data

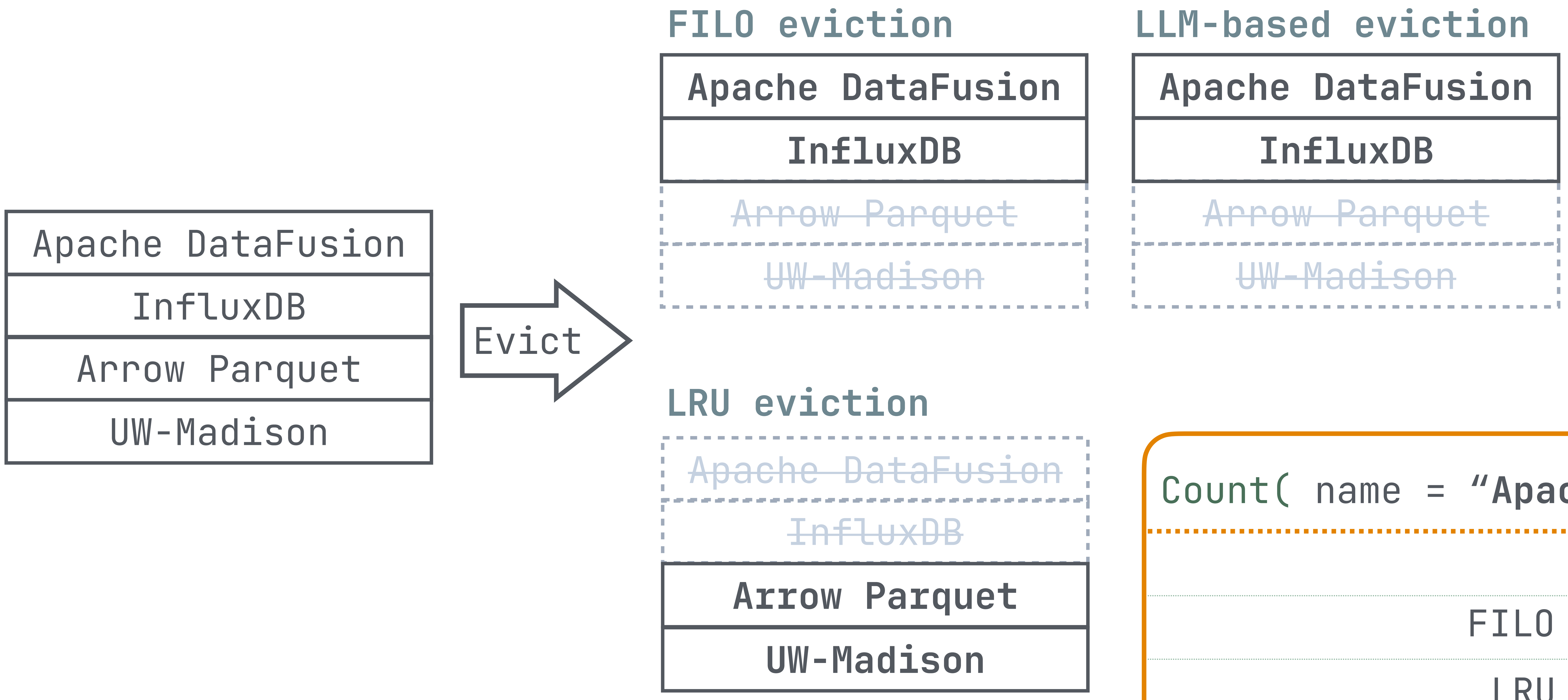


① Pushdown to reduce network

② Data-guided eviction

Proposed work

Motivating example



Apache DataFusion

InfluxDB

Arrow Parquet

UW-Madison

Evict

FILO eviction

Apache DataFusion

InfluxDB

~~Arrow Parquet~~

~~UW-Madison~~

LLM-based eviction

Apache DataFusion

InfluxDB

~~Arrow Parquet~~

~~UW-Madison~~

LRU eviction

~~Apache DataFusion~~

~~InfluxDB~~

Arrow Parquet

UW-Madison

~~Evicted~~

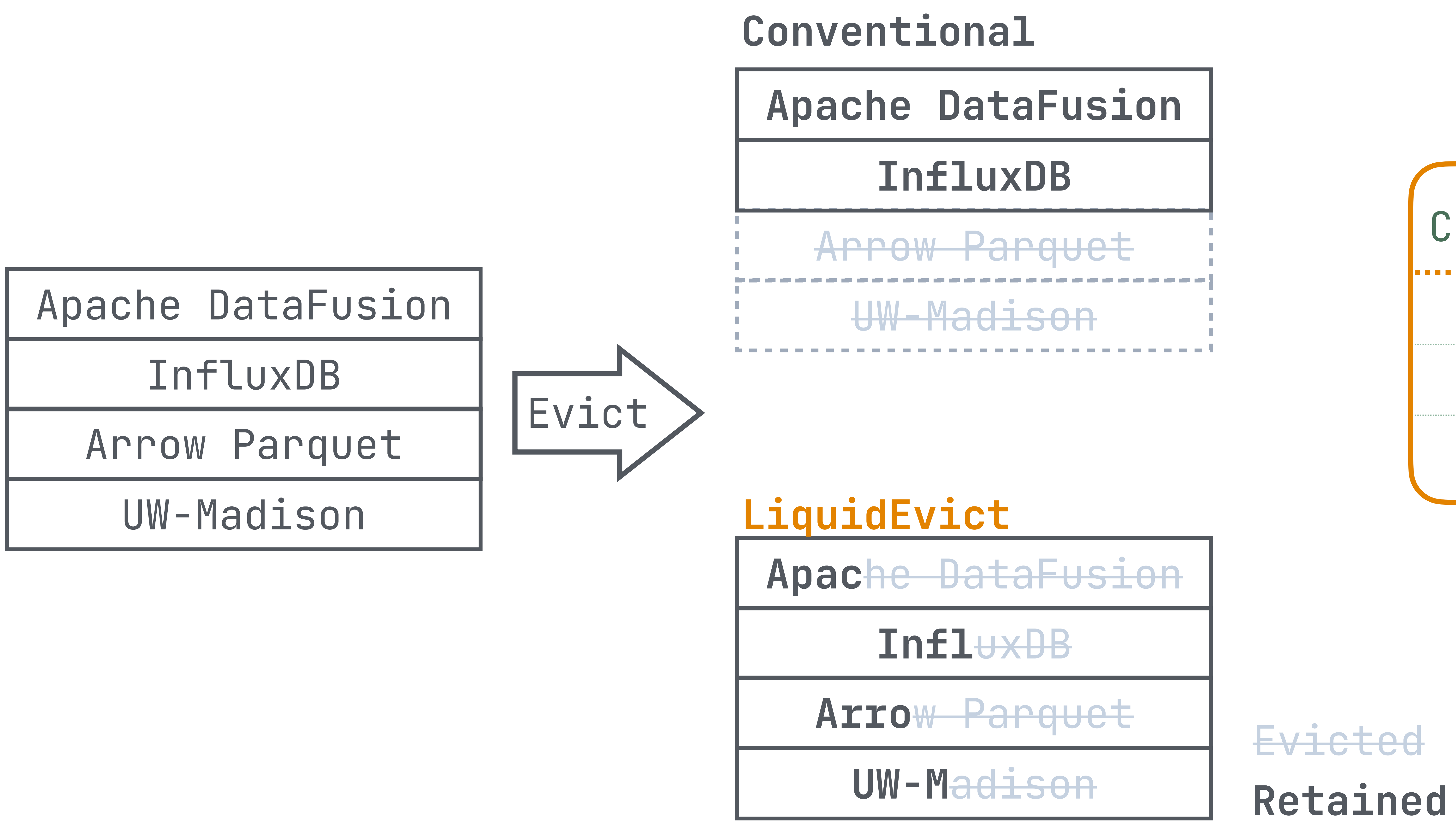
Retained

Can we do better?

Count(name = “**Apache DataFusion**”)

	Cache miss
FILO	2
LRU	2
LLM-based	2
Theoretical opt.	2

Data-guided eviction



Apache DataFusion

InfluxDB

Arrow Parquet

UW-Madison

Evict

Conventional

Apache DataFusion

InfluxDB

~~Arrow Parquet~~

~~UW-Madison~~

LiquidEvict

Apac~~he DataFusion~~

Infl~~uxDB~~

Arro~~w Parquet~~

UW-M~~adison~~

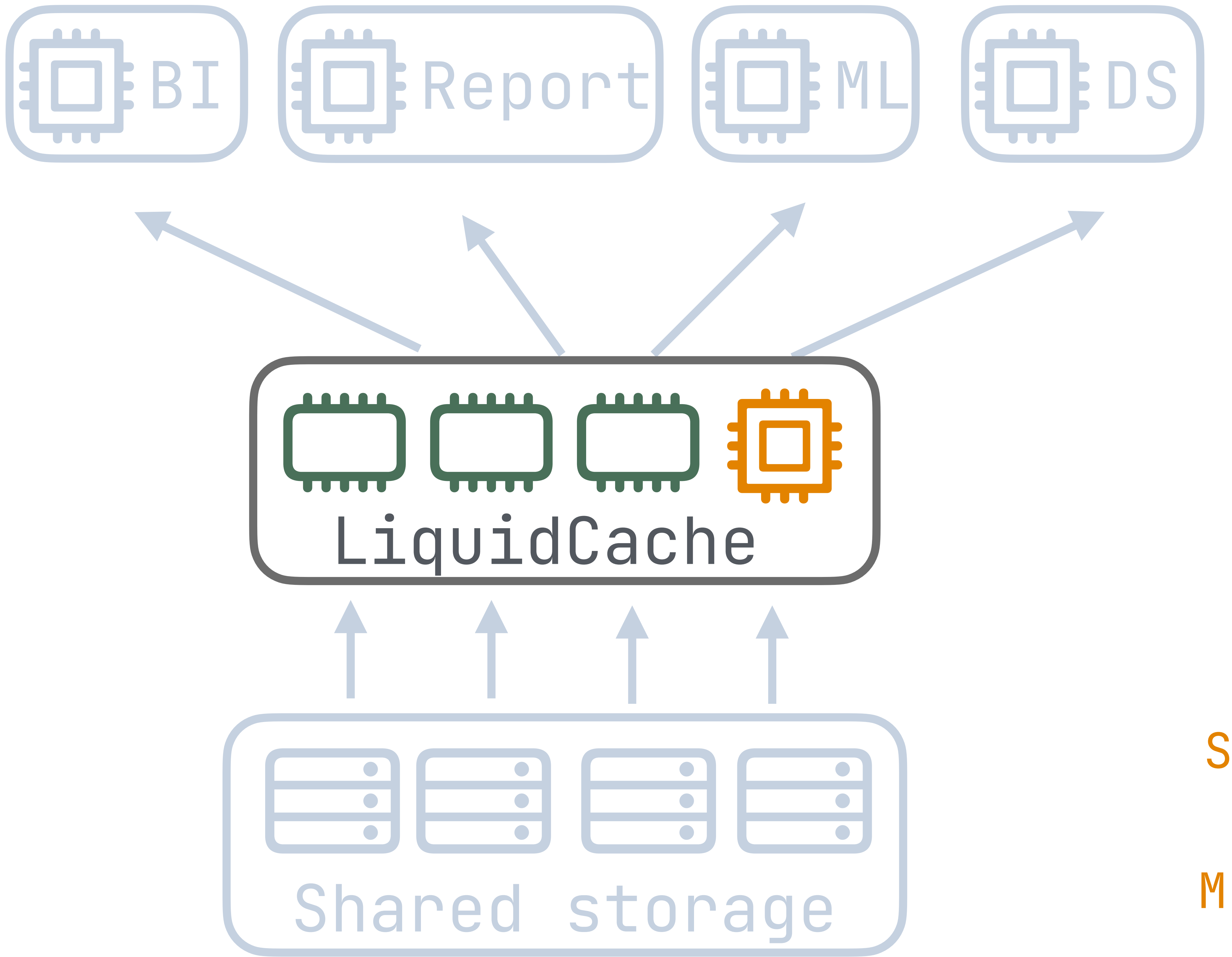
~~Evicted~~

Retained

Count(name = “**Apache DataFusion**”)

	Cache miss
Conventional	2
LiquidEvict	**1**

Insight: evict partial data



Some parts of data are more important than others

String prefix

Bit width

Dictionary

String length

Min/max/avg

Nullable mask

r_name == ‘EUROPE’

SearchPharse ≠ ‘’

p_brand == ‘Brand#23’

MIN(“URL”)

MIN(“TITLE”)

Real-world queries

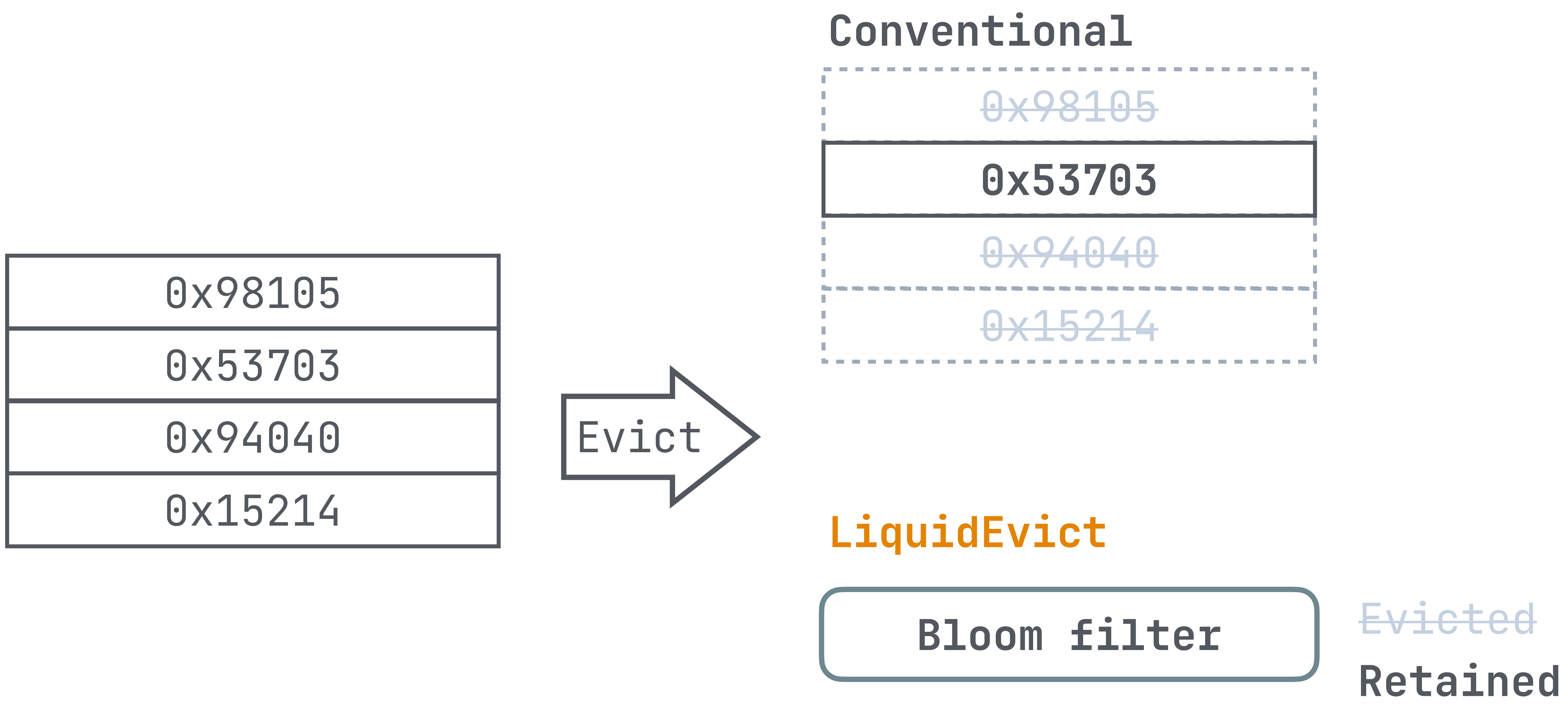
MobilePhoneModel ≠ ‘’

n_name == ‘SAUDI ARABIA’

UserID == 435090932899640449

l_quantity < 24

Evicts data, keep summaries



0x98105
0x53703
0x94040
0x15214

Evict

Conventional

~~0x98105~~
0x53703
~~0x94040~~
~~0x15214~~

LiquidEvict

Bloom filter

~~Evicted~~
Retained

Find “0x80309”?

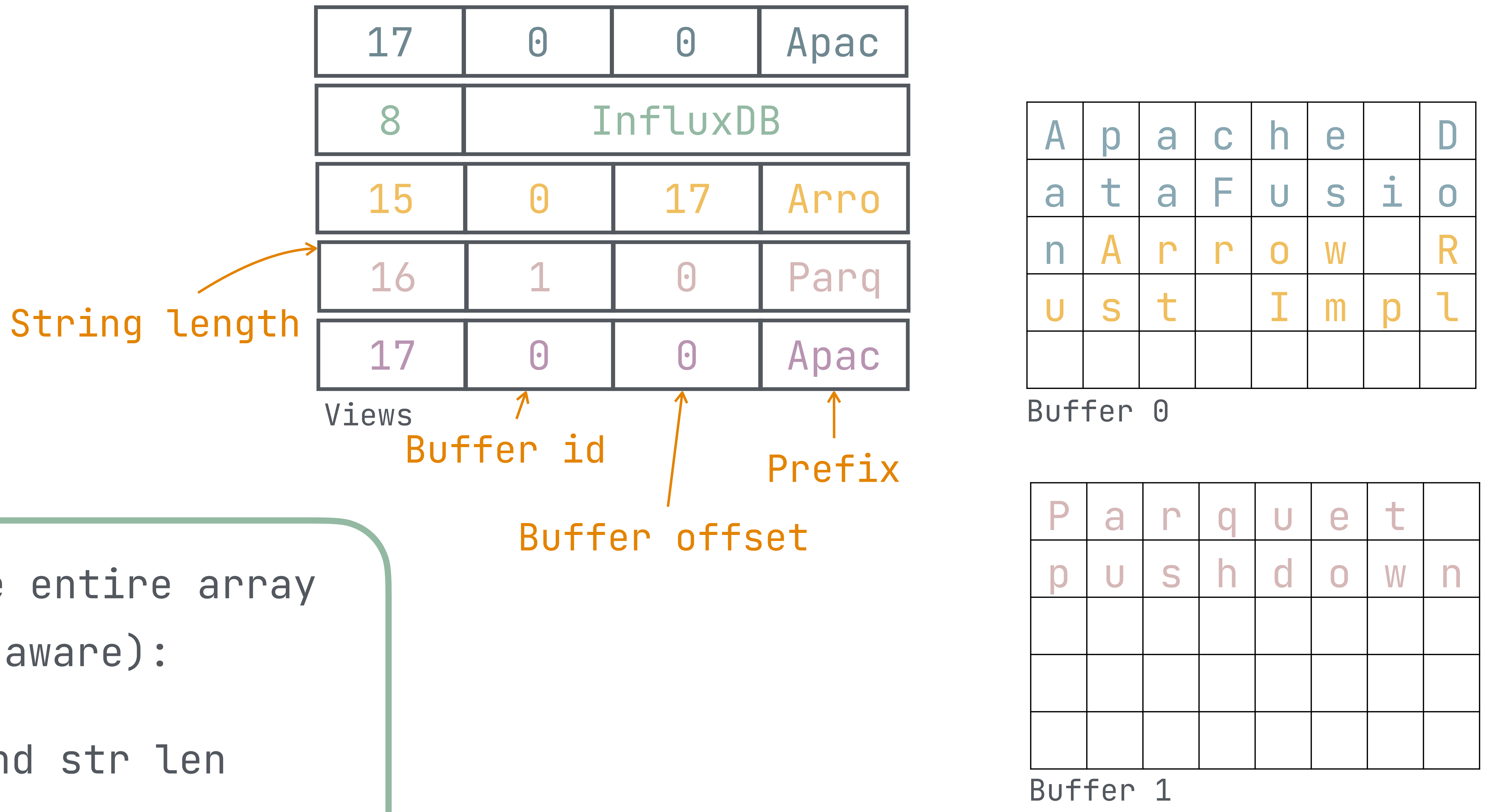
	Cache miss
Conventional	3
LiquidEvict	0?

Real world example: StringView eviction

Logical content

Apache DataFusion
InfluxDB
Arrow Rust Impl
Parquet pushdown
Apache DataFusion

Physical representation



17	0	0	Apac
8	InfluxDB		
15	0	17	Arro
16	1	0	Parq
17	0	0	Apac

Views

String length

Buffer id

Buffer offset

Prefix

A	p	a	c	h	e		D
a	t	a	F	u	s	i	o
n	A	r	r	o	w		R
u	s	t		I	m	p	l

Buffer 0

P	a	r	q	u	e	t	
p	u	s	h	d	o	w	n

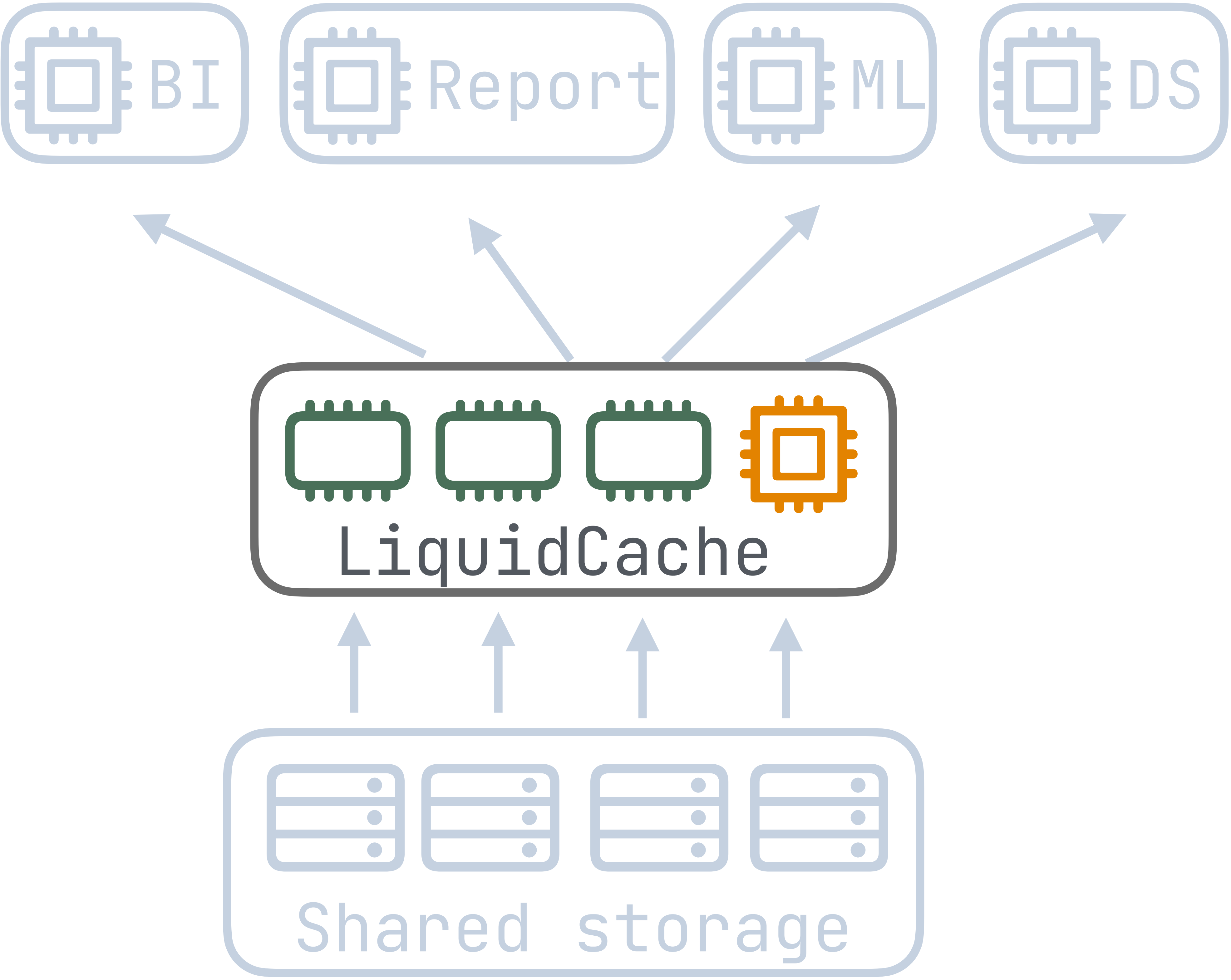
Buffer 1

Conventional: evict the entire array

LiquidEvict (structure-aware):

1. Evict buffers
2. Retain only prefix and str len
3. Retain only prefix

LiquidCache + LiquidEvict



Data-aware cache

1. Pushdown to reduce traffic
2. Efficient decoding

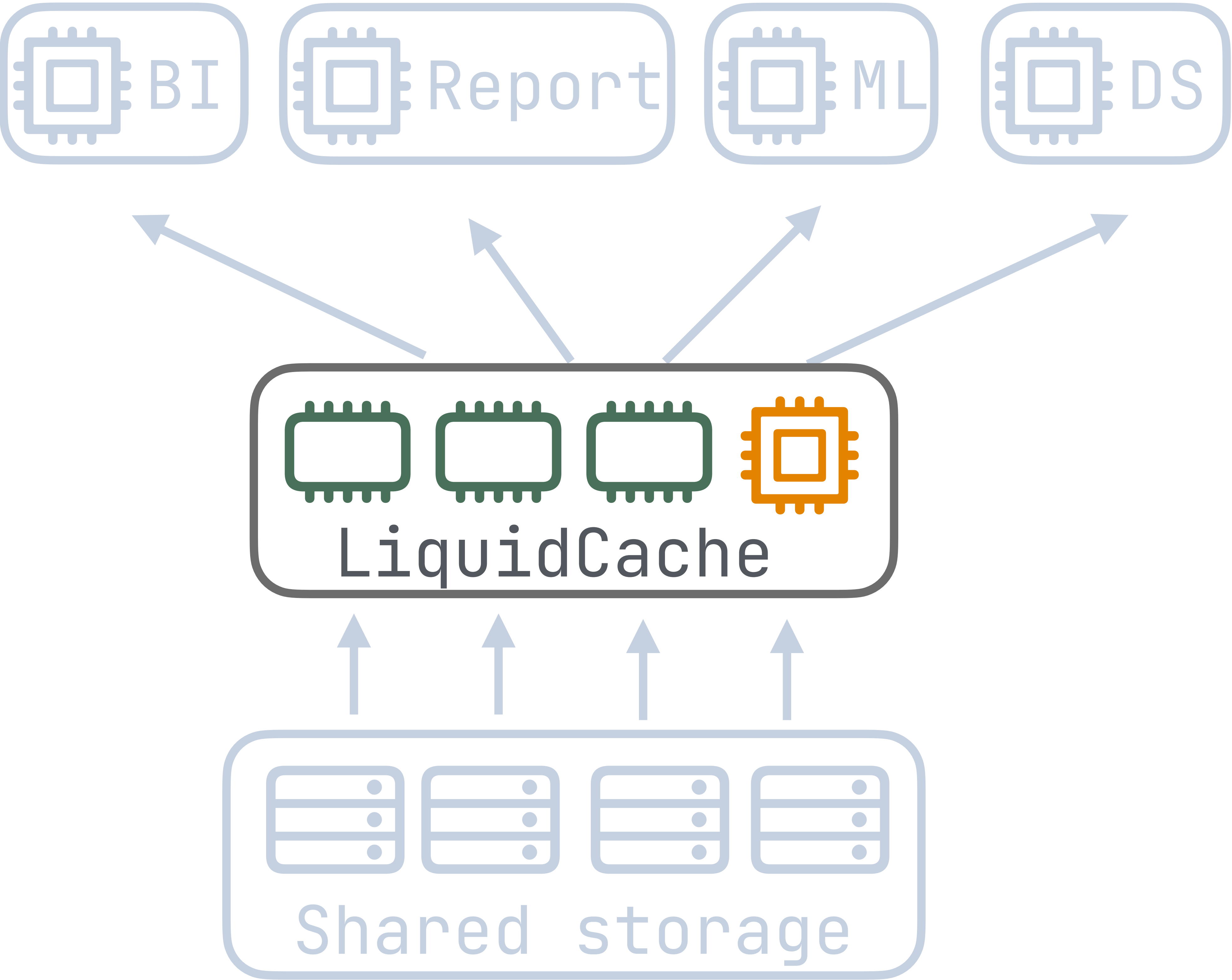
Data-aware eviction

1. Evicts unimportant parts of data
2. Evicts data, keep summaries

Shared, pushdown cache system

10x lower latency, 10x lower cost

Timeline



Jun. 1 - LiquidCache VLDB revision

July 1 - LiquidEvict prototype

Aug 1 - LiquidEvict bench & polish

Sep 1 - LiquidEvict VLDB submission

Dec 1 - LiquidEvict revision

Jan - Mar 2026 - Polish & present

May - Defense + graduate

Other first-author papers:

Bf-Tree: A Modern Read-Write-Optimized Concurrent Larger-Than-Memory Range Index.
Xiangpeng Hao, Badrish Chandramouli. (VLDB 2024)

Towards Buffer Management with Tiered Main Memory.
Xiangpeng Hao, Xinjing Zhou, Xiangyao Yu, Michael Stonebraker. (SIGMOD 2024)

LiquidCache

10x { cost | latency } reduction

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

Huge thanks to InfluxData for supporting the project in 2024-25

influxdata®

Need your help:

1. Make LiquidEvict real
2. Make LiquidCache ready for your company
3. Advance science and serve the public good

$50k charitable giving supports 1 PhD/year

20% of a FTE cost!

Better than most chatbots!