LayoutBench: Performance Benchmarking of Cloud Storage Layouts for Multimedia Data

Debopam Sanyal[1], Hongjie Chen[2], Alexey Tumanov[1], Joshua Kimball[2]

[1]Georgia Institute of Technology, [2]Dolby Laboratories

The Problem: Why Storage Layout Matters

The Problem: Why Storage Layout Matters

- Modern ML = massive unstructured datasets (images, audio, video)

The Problem: Why Storage Layout Matters

- Modern ML = massive unstructured datasets (images, audio, video)
- Stored in cloud object storage (AWS S3)

The Problem: Why Storage Layout Matters

- Modern ML = massive unstructured datasets (images, audio, video)
- Stored in cloud object storage (AWS S3)
- Before training: samples must be retrieved → compute nodes

The Problem: Why Storage Layout Matters

- Modern ML = massive unstructured datasets (images, audio, video)
- Stored in cloud object storage (AWS S3)
- Before training: samples must be retrieved → compute nodes
- Retrieval directly affects: GPU utilization, throughput, cost

The Problem: Why Storage Layout Matters

- Modern ML = massive unstructured datasets (images, audio, video)
- Stored in cloud object storage (AWS S3)
- Before training: samples must be retrieved → compute nodes
- Retrieval directly affects: GPU utilization, throughput, cost

The Gap in Existing Benchmarks

The Gap in Existing Benchmarks

- OLTP[1]/OLAP[2]/time-series benchmarks → assume fixed storage abstraction
- COSBench[3], CNSBench[4] → evaluate services, not layouts

[1] Zhang et al. "Cdsben: Benchmarking the performance of storage services in cloud-native database system at bytedance." VLDB 2023.
[2] Lutsch et al. "Benchmarking Analytical Query Processing in Intel SGXv2." EDBT 2025.
[3] Zheng et al. "Cosbench: Cloud object storage benchmark." ACM ICPE 2013.
[4] Merenstein, et al. "CNSBench: A cloud native storage benchmark." Usenix FAST 2021.

The Gap in Existing Benchmarks

- OLTP[1]/OLAP[2]/time-series benchmarks → assume fixed storage abstraction
- COSBench[3], CNSBench[4] → evaluate services, not layouts
- Columnar format studies → focus on relational analytics
- ML-side (WebDataset[5], Petastorm[6], Lance[7]) → no controlled comparison

[1] Zhang et al. "Cdsben: Benchmarking the performance of storage services in cloud-native database system at bytedance." VLDB 2023.
[2] Lutsch et al. "Benchmarking Analytical Query Processing in Intel SGXv2." EDBT 2025.
[3] Zheng et al. "Cosbench: Cloud object storage benchmark." ACM ICPE 2013.
[4] Merenstein, et al. "CNSBench: A cloud native storage benchmark." Usenix FAST 2021.
[5] Aizman et al. "High performance I/O for large scale deep learning." IEEE BigData 2019.
[6] Petastorm Documentation. https://petastorm.readthedocs.io/en/latest/ index.html. 2022.
[7] Pace et al. "Lance: Efficient random access in columnar storage through adaptive structural encodings." arXiv 2025.

The Gap in Existing Benchmarks

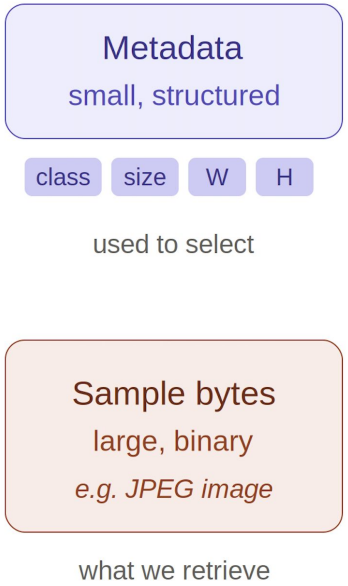
- OLTP[1]/OLAP[2]/time-series benchmarks → assume fixed storage abstraction
- COSBench[3], CNSBench[4] → evaluate services, not layouts
- Columnar format studies → focus on relational analytics
- ML-side (WebDataset[5], Petastorm[6], Lance[7]) → no controlled comparison

No benchmark for multimedia retrieval on cloud storage!

[1] Zhang et al. "Cdsben: Benchmarking the performance of storage services in cloud-native database system at bytedance." VLDB 2023.
[2] Lutsch et al. "Benchmarking Analytical Query Processing in Intel SGXv2." EDBT 2025.
[3] Zheng et al. "Cosbench: Cloud object storage benchmark." ACM ICPE 2013.
[4] Merenstein, et al. "CNSBench: A cloud native storage benchmark." Usenix FAST 2021.
[5] Aizman et al. "High performance I/O for large scale deep learning." IEEE BigData 2019.
[6] Petastorm Documentation. https://petastorm.readthedocs.io/en/latest/ index.html. 2022.
[7] Pace et al. "Lance: Efficient random access in columnar storage through adaptive structural encodings." arXiv 2025.

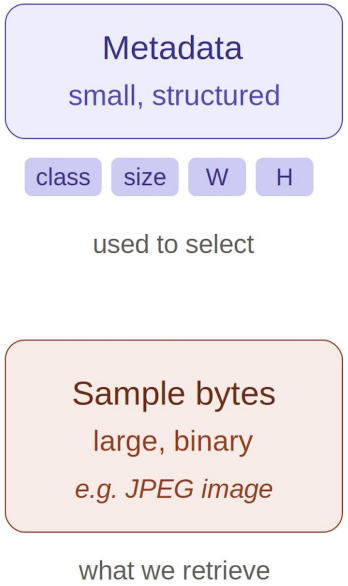
What is a "Storage Layout"?

What is a "Storage Layout"?



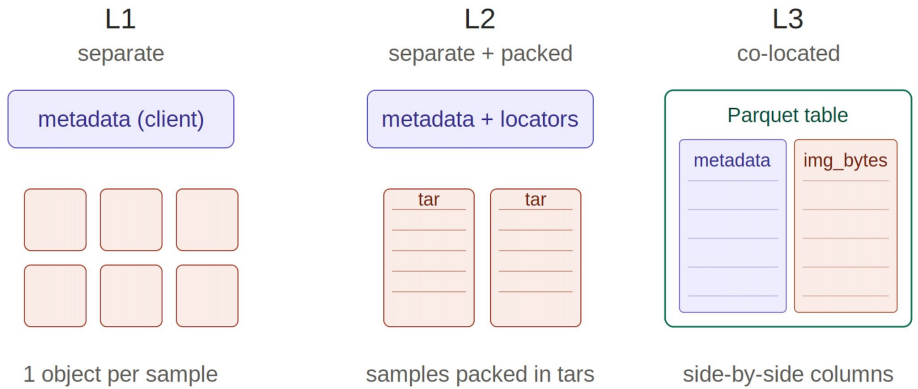
What is a "Storage Layout"?

- Every layout manages two things:
  - Metadata — class, size, width, height (used to select)
  - Sample data — raw image bytes (what we retrieve)



What is a "Storage Layout"?

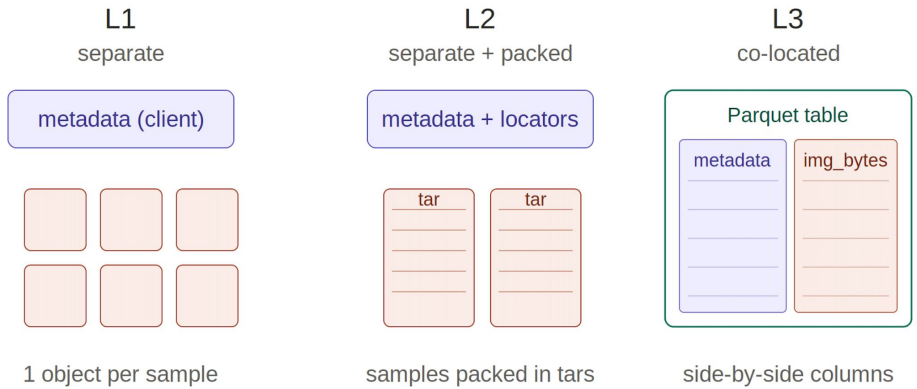
- Every layout manages two things:
  - Metadata — class, size, width, height (used to select)
  - Sample data — raw image bytes (what we retrieve)



What is a "Storage Layout"?

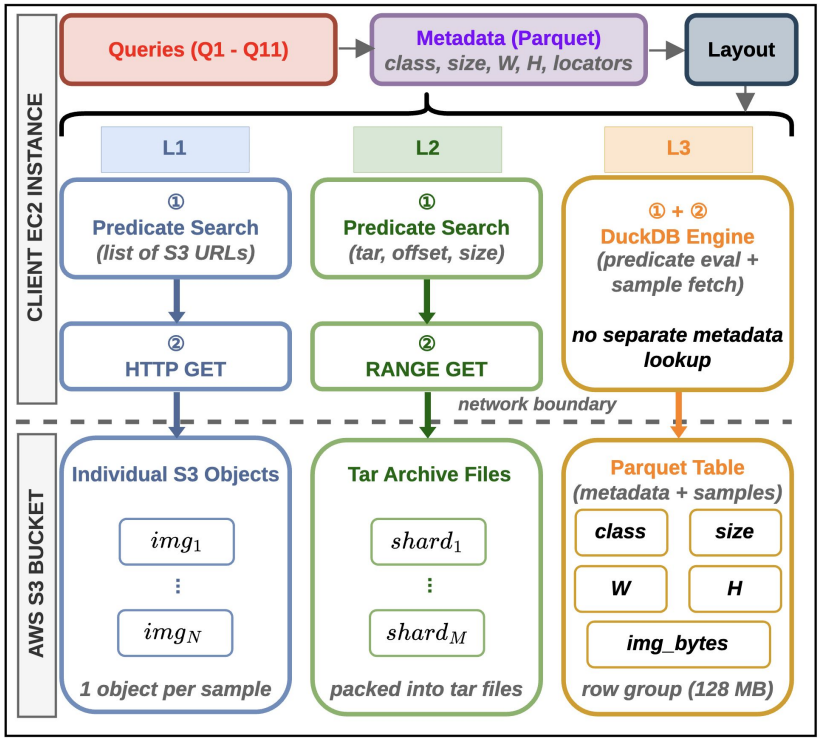
- Every layout manages two things:
  - Metadata — class, size, width, height (used to select)
  - Sample data — raw image bytes (what we retrieve)

- Layouts differ in how they co-locate or separate these



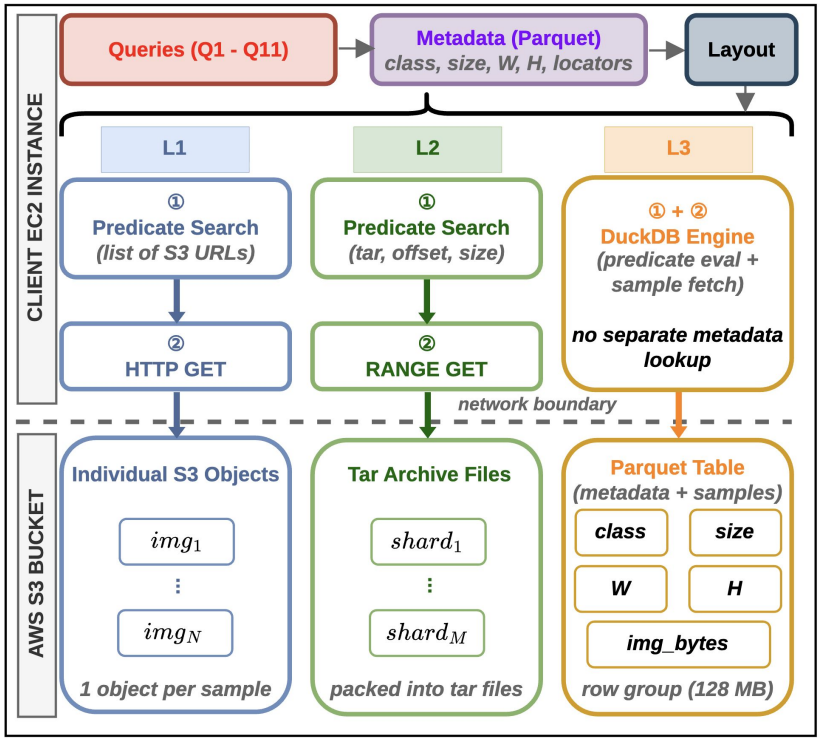
The Three Layouts (Architecture Overview)

The Three Layouts (Architecture Overview)



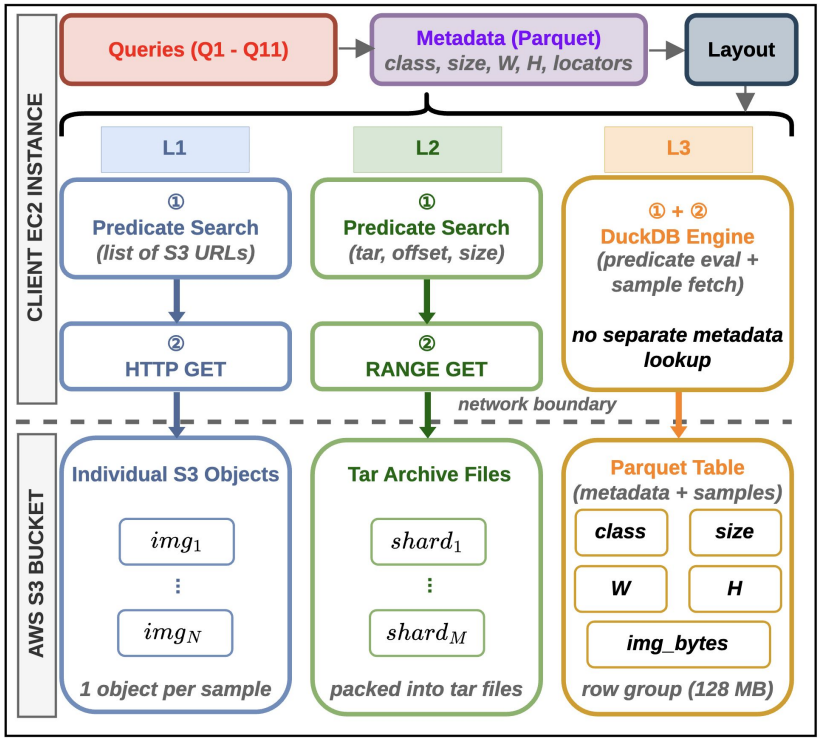
The Three Layouts (Architecture Overview)

- L1: One sample → one S3 object



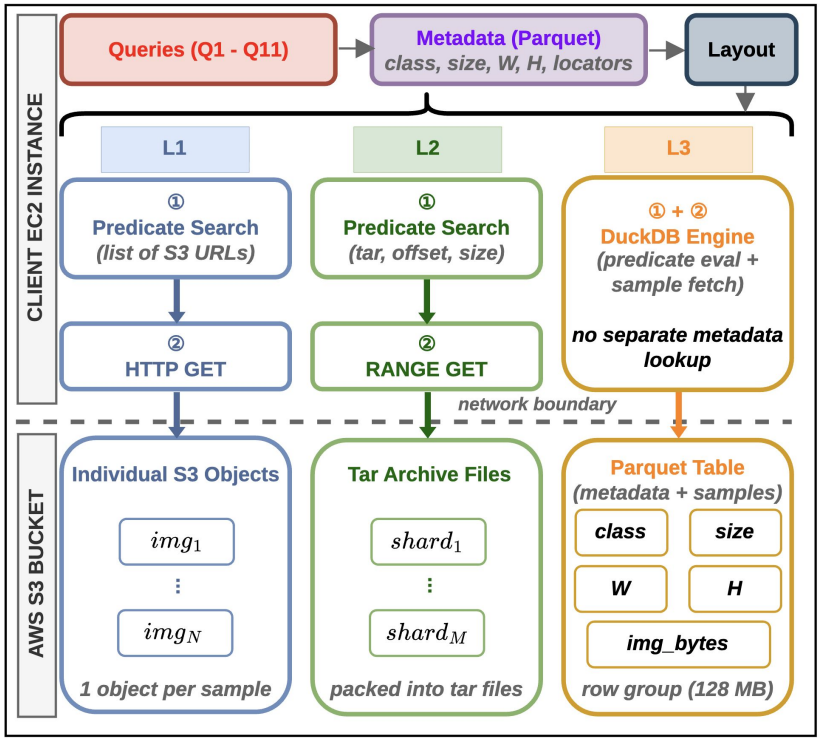
The Three Layouts (Architecture Overview)

- L1: One sample → one S3 object
- L2: Samples packed into tar archives (byte-range GETs)



The Three Layouts (Architecture Overview)

- L1: One sample → one S3 object
- L2: Samples packed into tar archives (byte-range GETs)
- L3: Samples as BLOBs in Parquet columns (DuckDB queries)



Experimental Setup

Experimental Setup

- Dataset: ImageNet-1K (mini / medium / full — 1%, 10%, 100%)

Experimental Setup

- Dataset: ImageNet-1K (mini / medium / full — 1%, 10%, 100%)
- 11 queries: atomic (Q1–Q6) + composite (Q7–Q11)

Experimental Setup

- Dataset: ImageNet-1K (mini / medium / full — 1%, 10%, 100%)
- 11 queries: atomic (Q1–Q6) + composite (Q7–Q11)
- 6 EC2 instances: 3 bandwidth tiers + 3 memory tiers

Experimental Setup

- Dataset: ImageNet-1K (mini / medium / full — 1%, 10%, 100%)
- 11 queries: atomic (Q1–Q6) + composite (Q7–Q11)
- 6 EC2 instances: 3 bandwidth tiers + 3 memory tiers
- Metrics: retrieval time T, data transferred D, cost C

Dataset statistics at the storage server and client

Table 1: Dataset statistics at the storage server and client.

Dataset	Location	Attribute	Storage Layout Plan		
			L1	L2	L3
mini	Server	Number of Files	12,756	23	22
		File Size	1.39 GB	1.38 GB	1.36 GB
	Client	Index File Size	553 KB	810 KB	0
medium	Server	Number of Files	128,066	28	28
		File Size	13.91 GB	13.76 GB	13.57 GB
	Client	Index File Size	4.4 MB	6.8 MB	0
full	Server	Number of Files	1,281,167	276	274
		File Size	139.25 GB	137.73 GB	135.85 GB
	Client	Index File Size	35.6 MB	54.7 MB	0

Summary of the queries

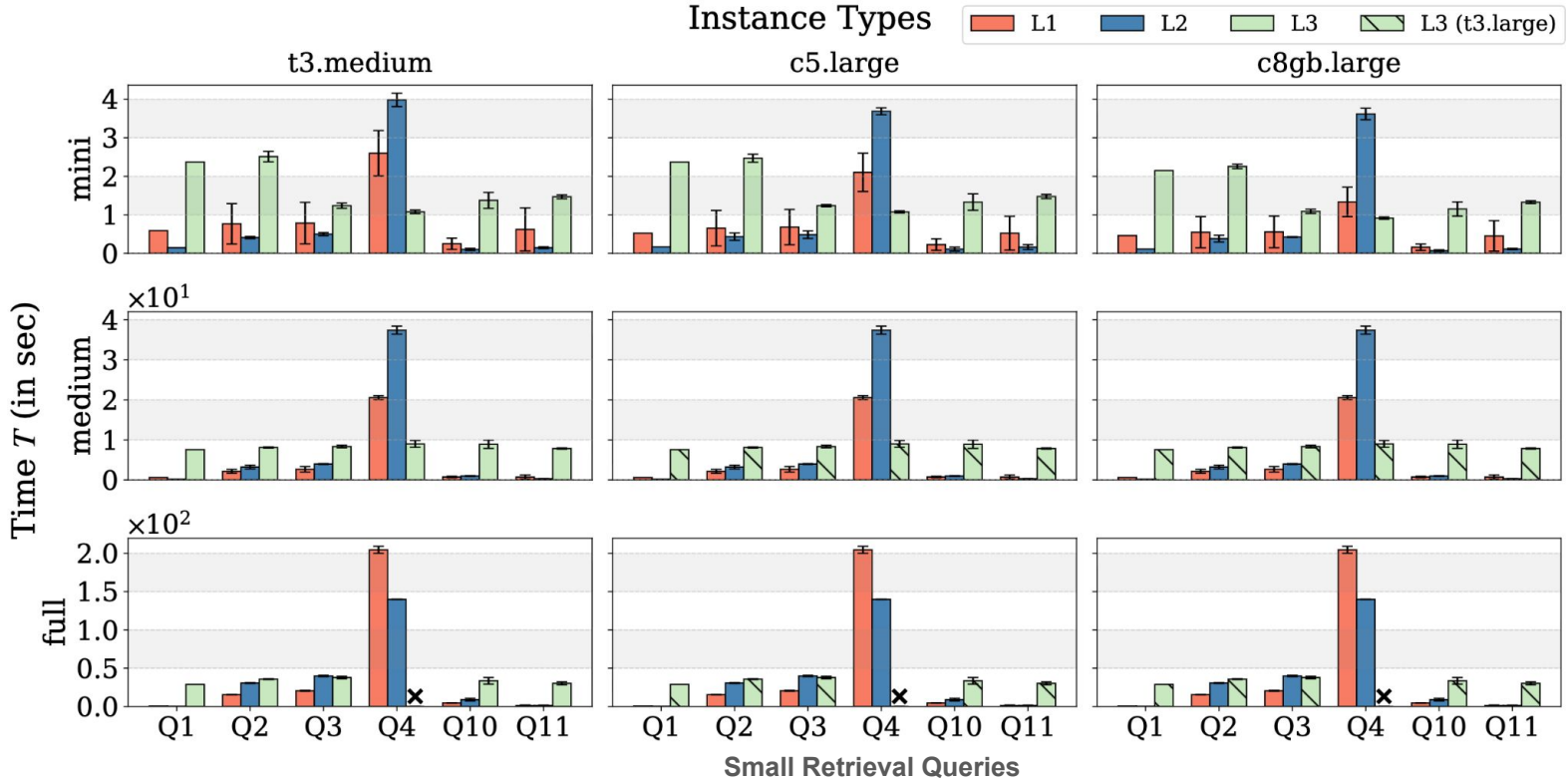
Q.	Type	Predicate				Query Description	Retrieval – Num. of Files		
		C	S	W	H		mini	medium	full
Q1	One Sample	✓				one image of *streetcar*	1	1	1
Q2	Batch Fetch	✓				N images of *lemon* ($N = 10(mini), 100(medium), 1000(full)$)	10	100	1,000
Q3	One Class	✓				images of *cello*	13	130	1,300
Q4	Regex Match	✓				images whose labels are prefixed with snake, i.e., LIKE *%snake*	130	1,300	13,000
Q5	Fan-out	✓				images of *Old_English_sheepdog*, *giant_panda*, and *water_buffalo*	39	390	3,900
Q6	File size Filter		✓			images that are at least 500 KB	86	801	7,837
Q7	Resolution Filter			✓	✓	images whose width and height are both at least 1024 pixels	133	1,309	12,635
Q8	Cross-column			✓	✓	images whose width / height ratio is greater than 1.5	2,311	23,006	229,710
Q9	Multi-label Filter	✓	✓			images of *canoe* and *starfish* that are smaller than 100 KB	6	47	575
Q10	Selective Filter	✓	✓			images of *tiger_cat* that are smaller than 100 KB	1	27	294
Q11	All-column Filter	✓	✓	✓	✓	images of *bagel* under 200 KB with width and height at least 512 pixels	1	6	32

Comparison of client EC2 instance types

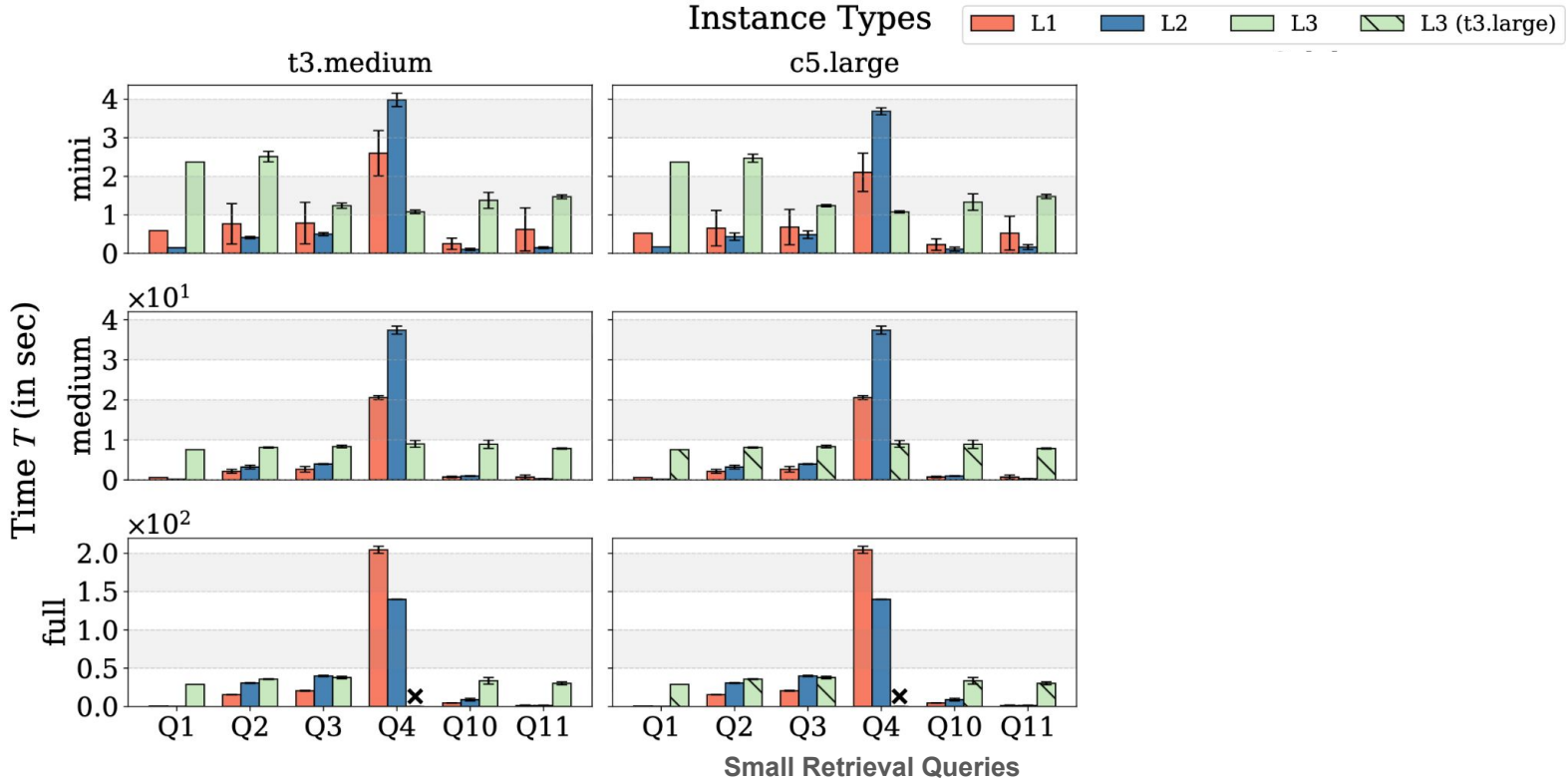
API Name	Network	Memory	vCPUs	Hourly Cost
t3.medium	Up to 5 Gigabit	4 GiB	2 vCPUs	$0.0416
c5.large	Up to 10 Gigabit	4 GiB	2 vCPUs	$0.0850
c8gb.large	Up to 20 Gigabit	4 GiB	2 vCPUs	$0.1185
t3.large	Up to 5 Gigabit	8 GiB	2 vCPUs	$0.0832
t3.xlarge	Up to 5 Gigabit	16 GiB	4 vCPUs	$0.1664
t3.2xlarge	Up to 5 Gigabit	32 GiB	8 vCPUs	$0.3328

Finding #1: Latency Crossover (L1 vs L2); L3's Baseline Overhead

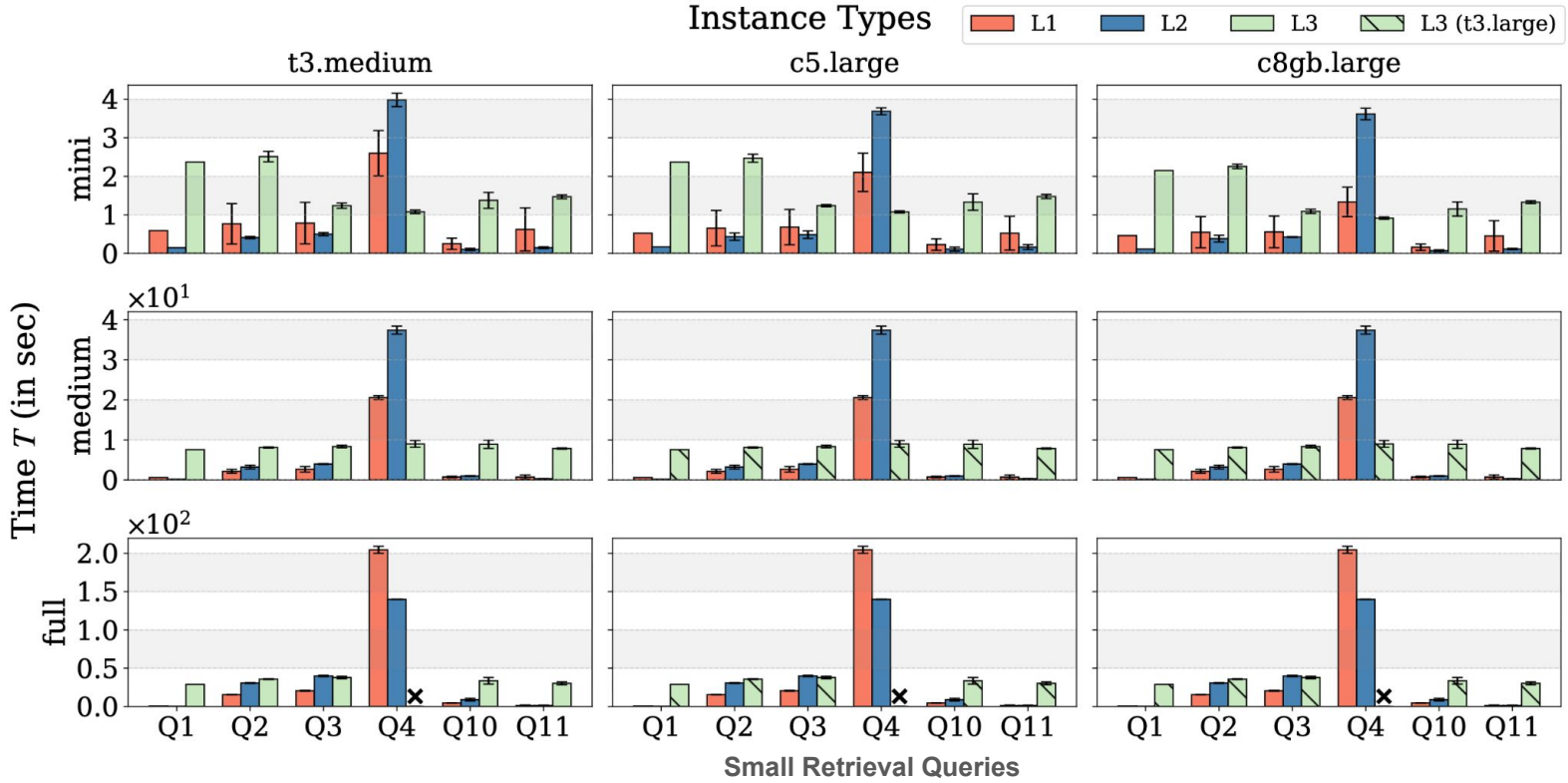
End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #1]



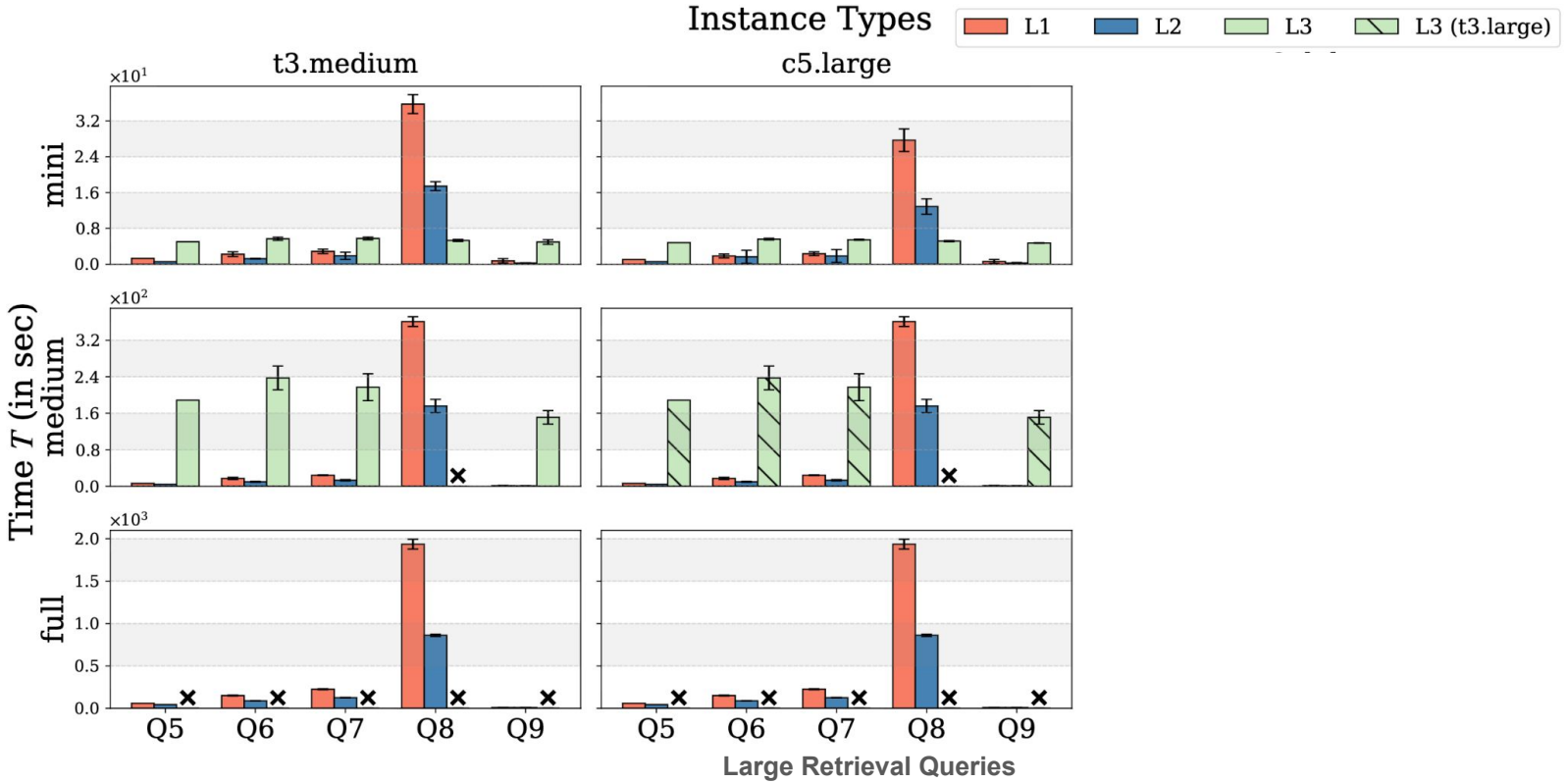
End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #1]



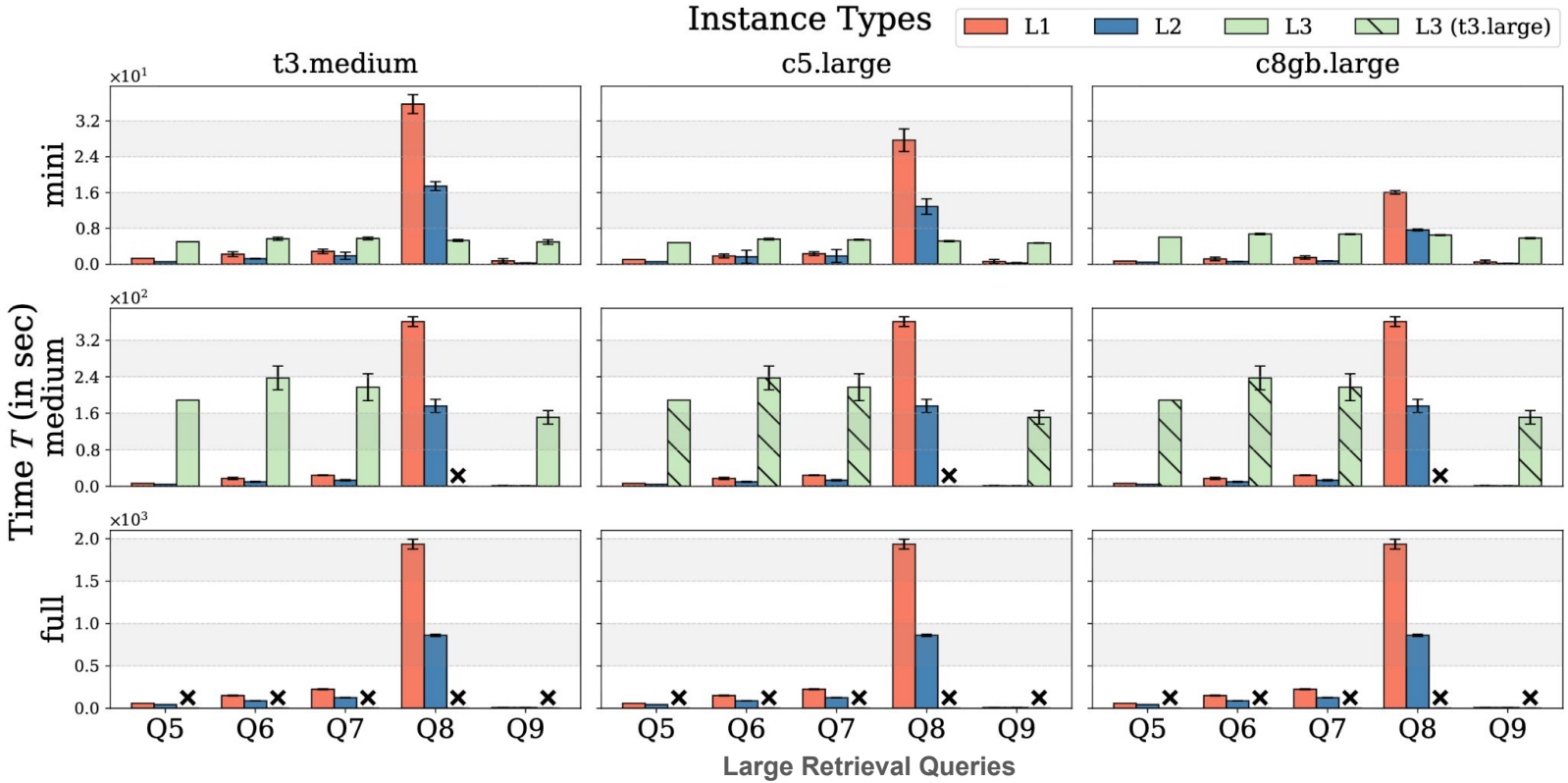
End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #1]



End-to-end retrieval time (Q5-Q9) [Finding #1]



End-to-end retrieval time (Q5-Q9) [Finding #1]



Finding #1: Latency Crossover (L1 vs L2); L3's Baseline Overhead

- Small retrievals: L1 ≈ L2 (L2 slightly faster)
- Mid retrievals: L1 wins (fewer connections dominate)
- Large retrievals: L2 wins via connection reuse

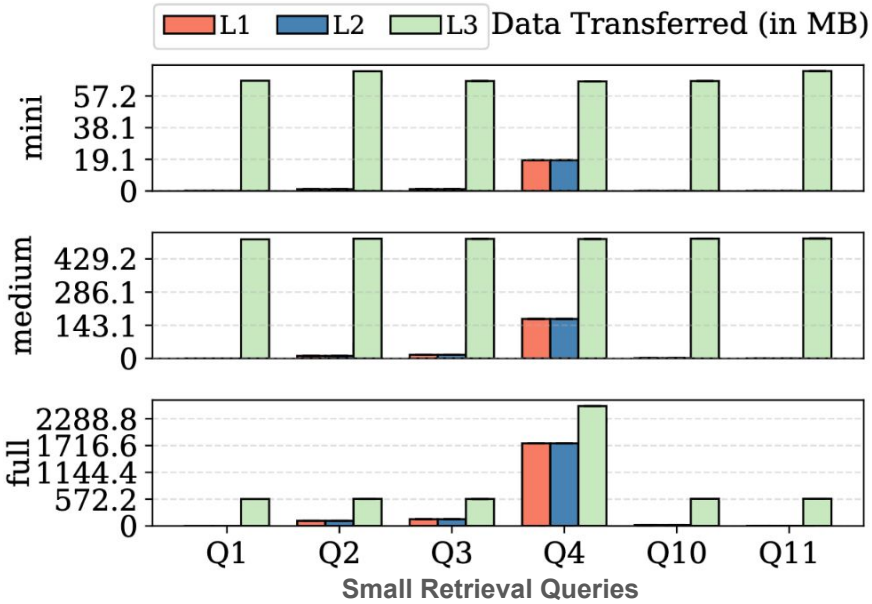
Finding #1: Latency Crossover (L1 vs L2); L3's Baseline Overhead

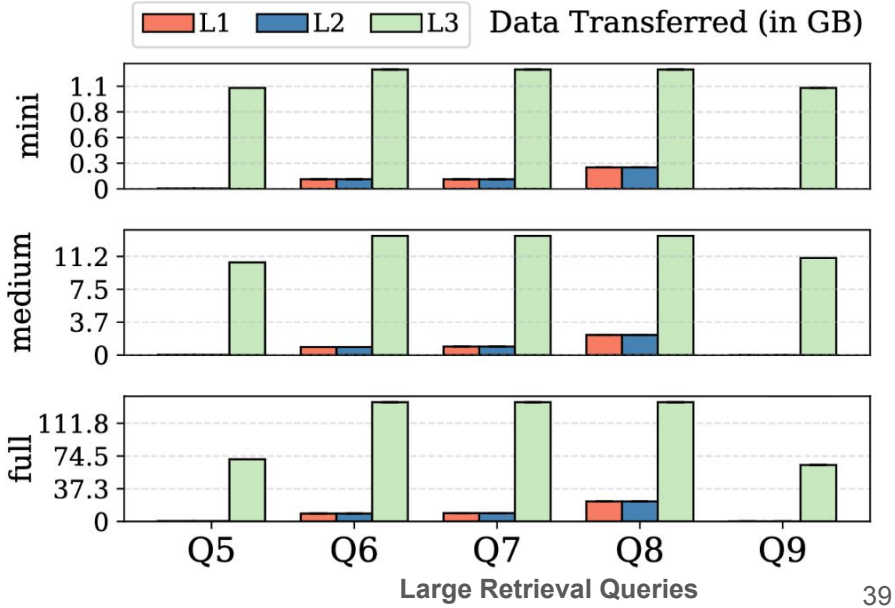
- Small retrievals: L1 ≈ L2 (L2 slightly faster)
- Mid retrievals: L1 wins (fewer connections dominate)
- Large retrievals: L2 wins via connection reuse

- L3 is slow for small & mid retrievals (DuckDB init overhead)
- L3 is fastest for very large retrievals (row-group reads win)
- L3 cannot run on small instances at full scale (✗ marks)

Finding #2: Data Transfer Story

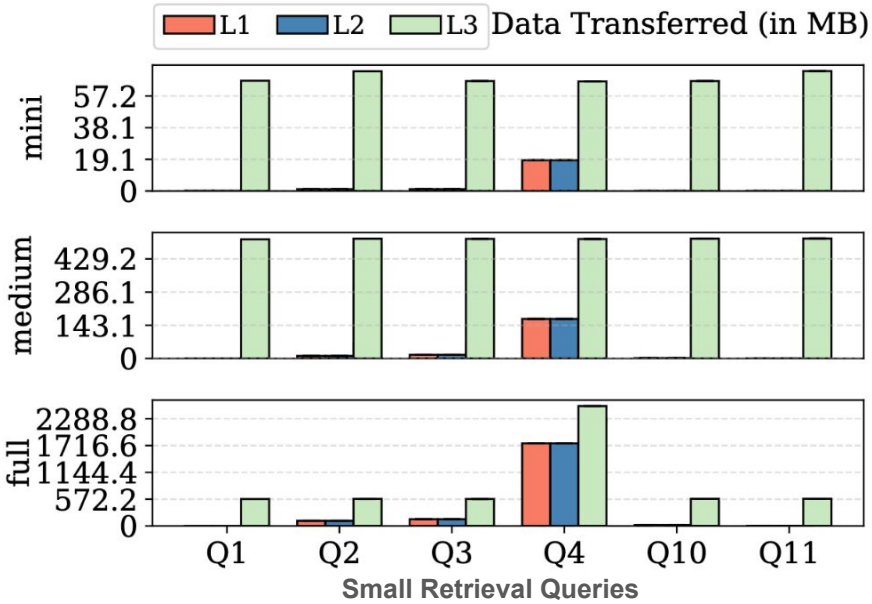
Finding #2: Data Transfer Story

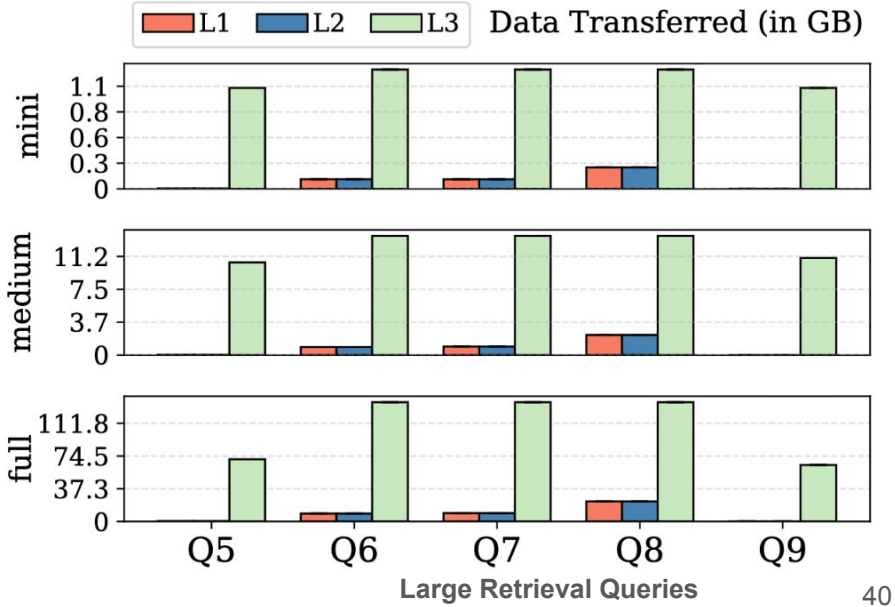




Finding #2: Data Transfer Story

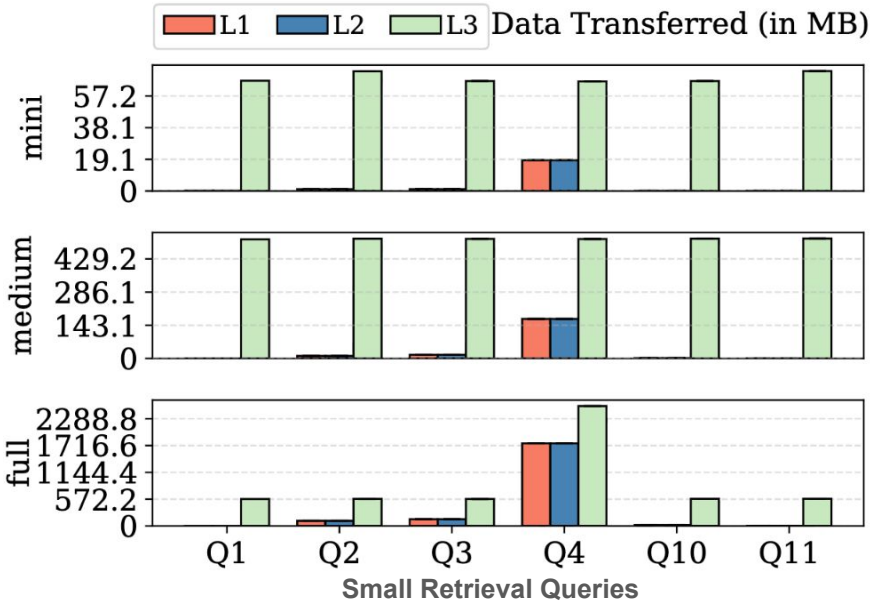
- L1 ≈ L2: fetch close to minimum bytes needed

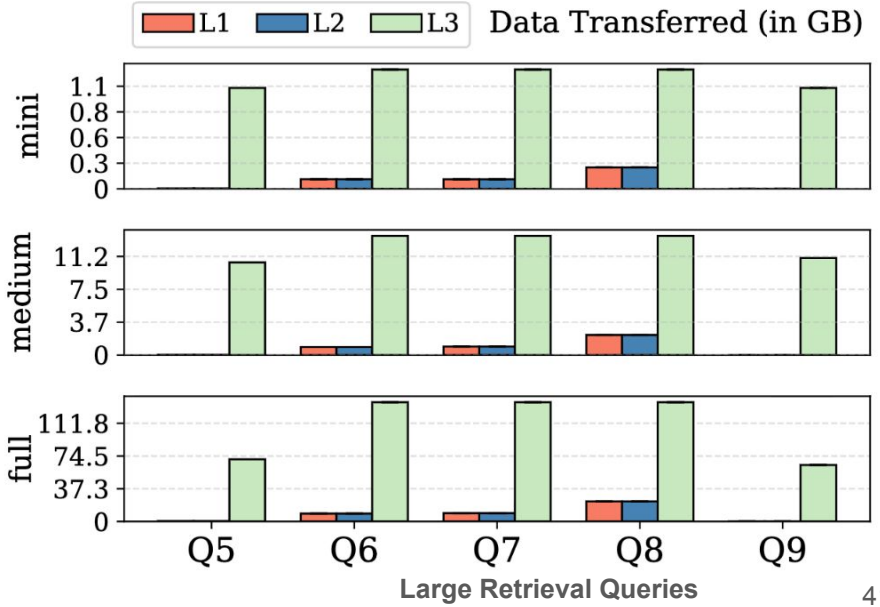




Finding #2: Data Transfer Story

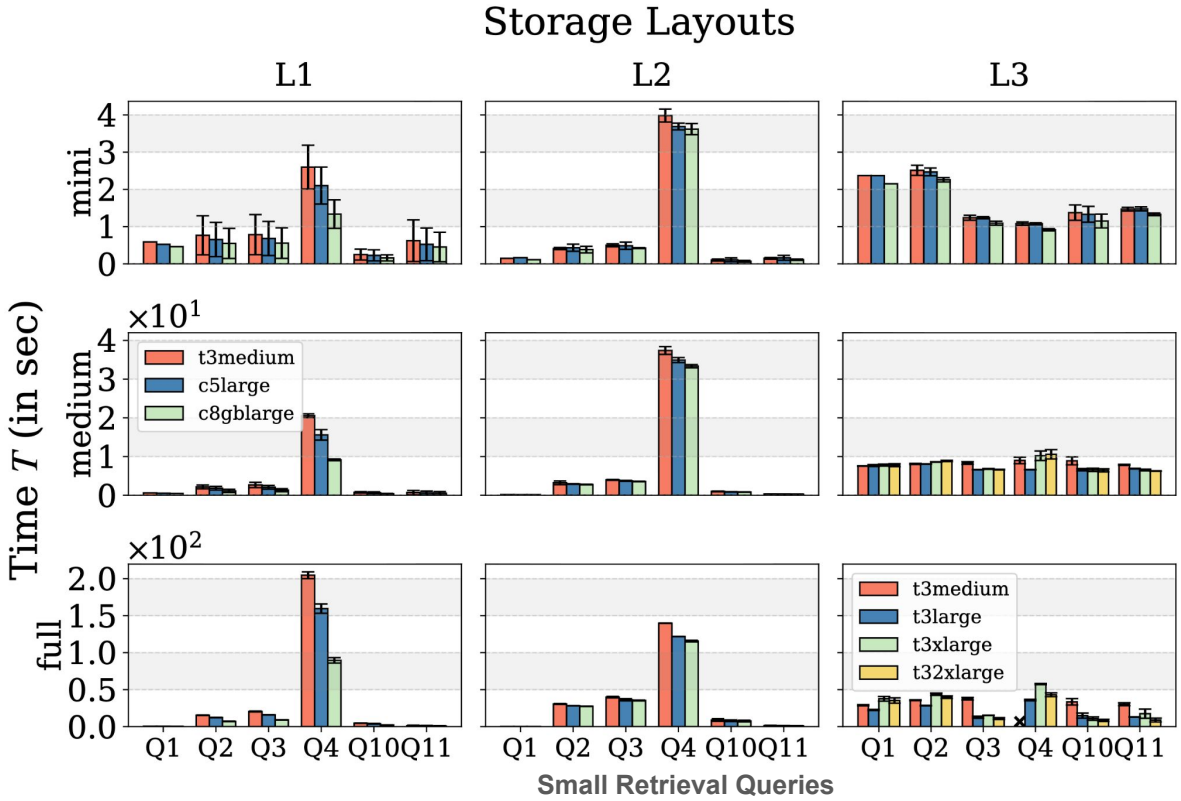
- L1 ≈ L2: fetch close to minimum bytes needed
- L3: fetches entire row groups → big overhead on selective queries



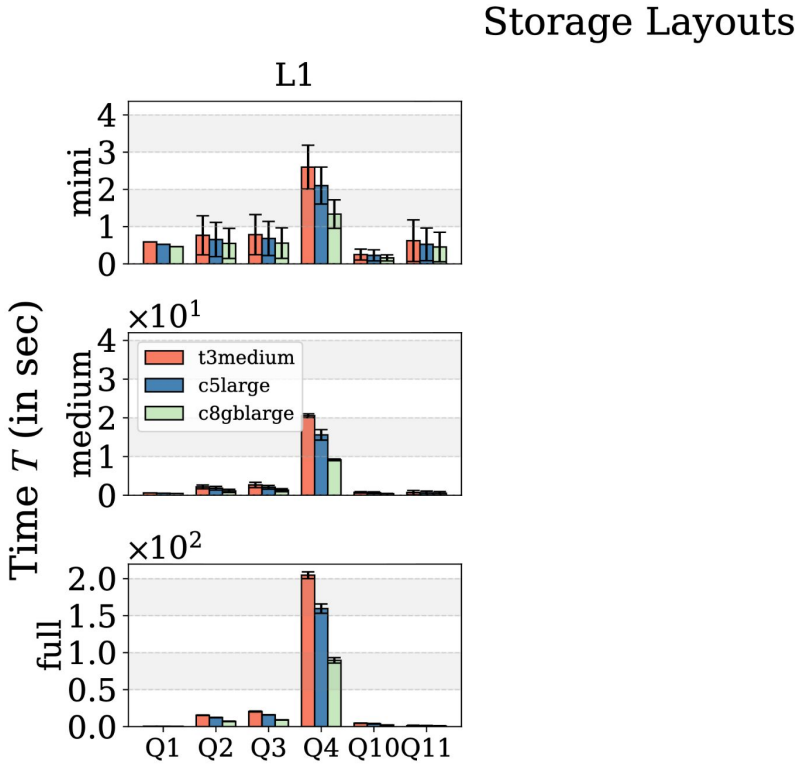


Finding #3: Each Layout Has a Different Bottleneck

End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #3]

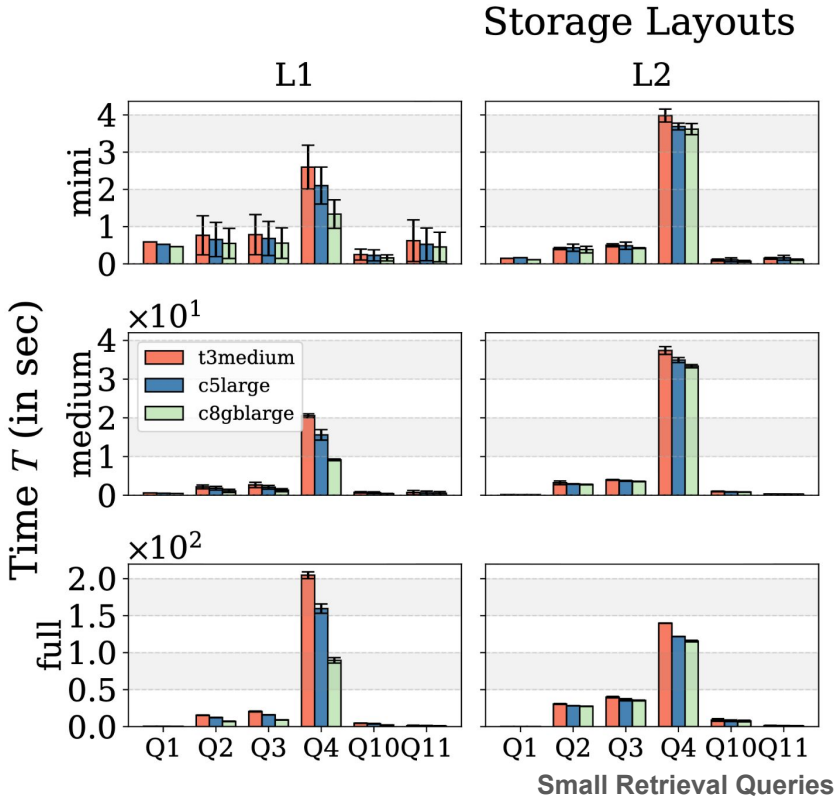


End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #3]

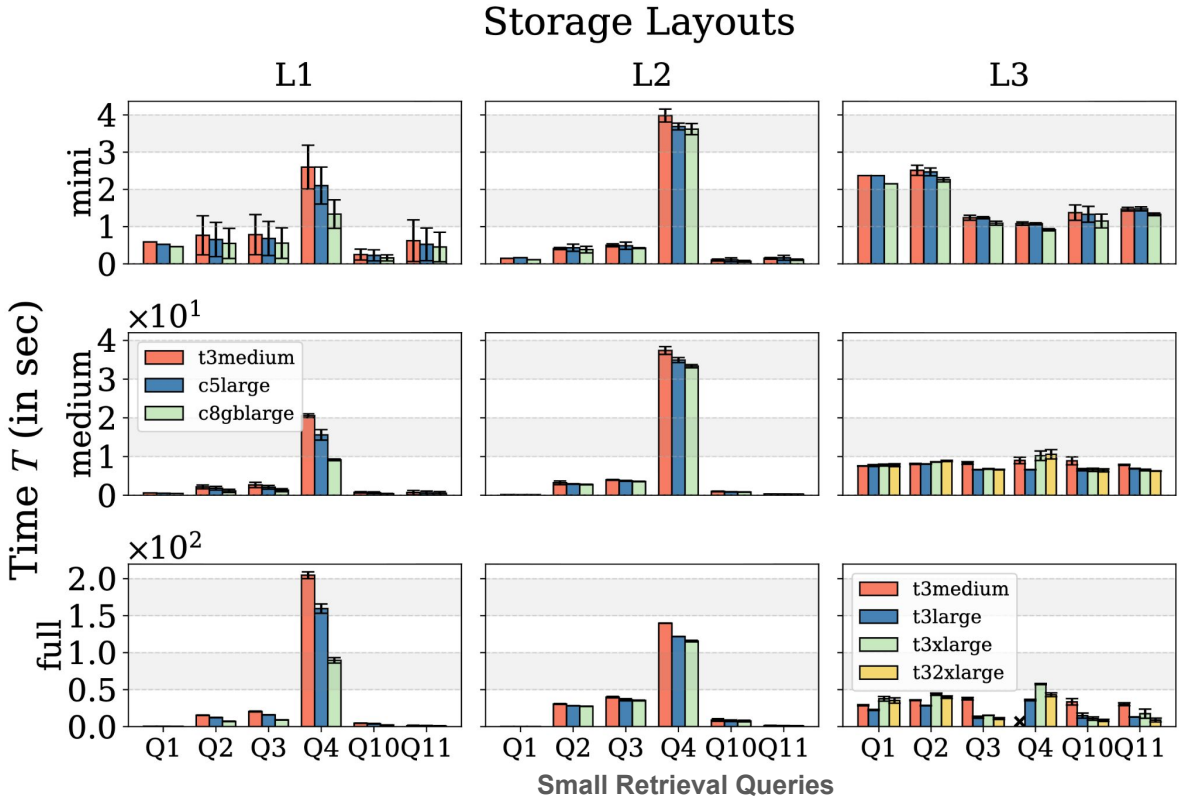


Small Retrieval Queries

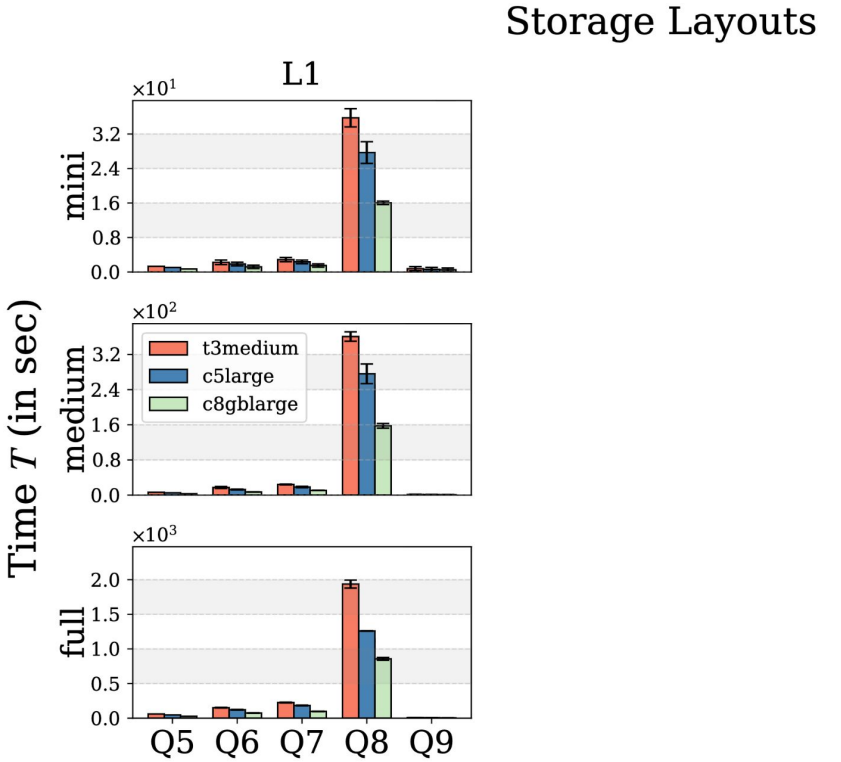
End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #3]



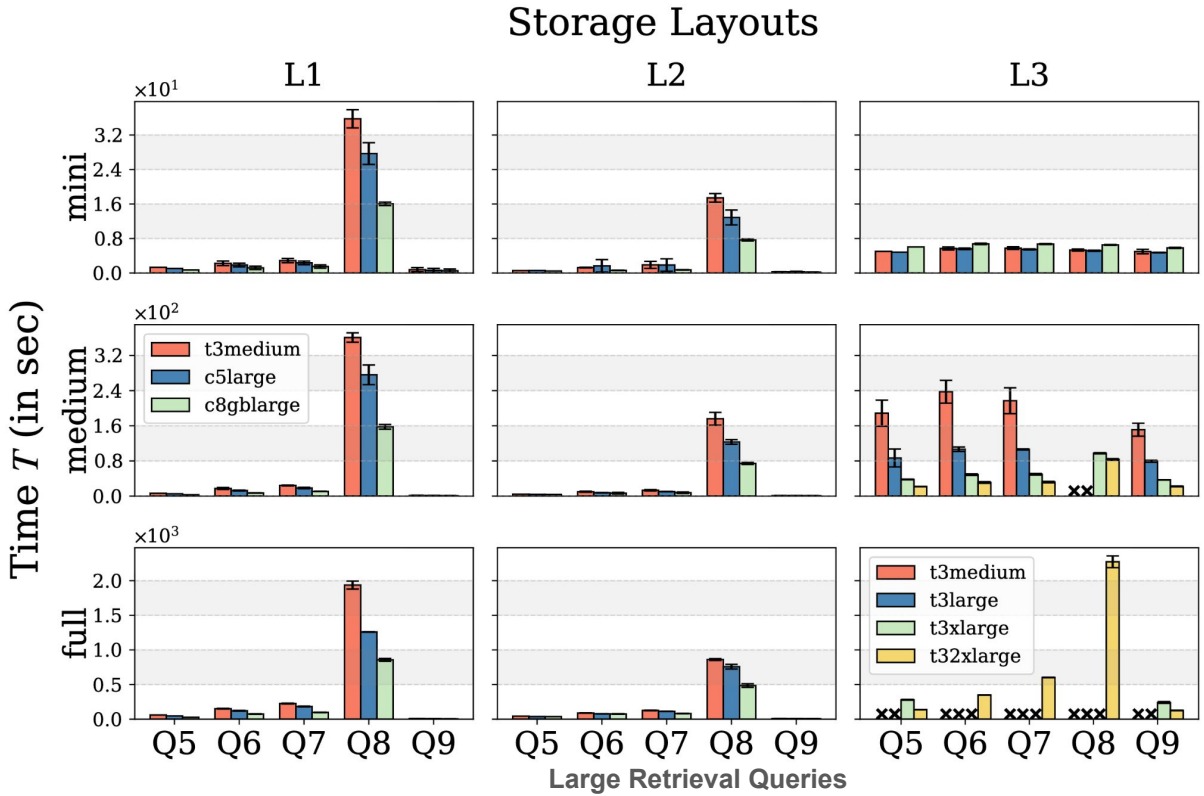
End-to-end retrieval time (Q1-Q4 & Q10-Q11) [Finding #3]



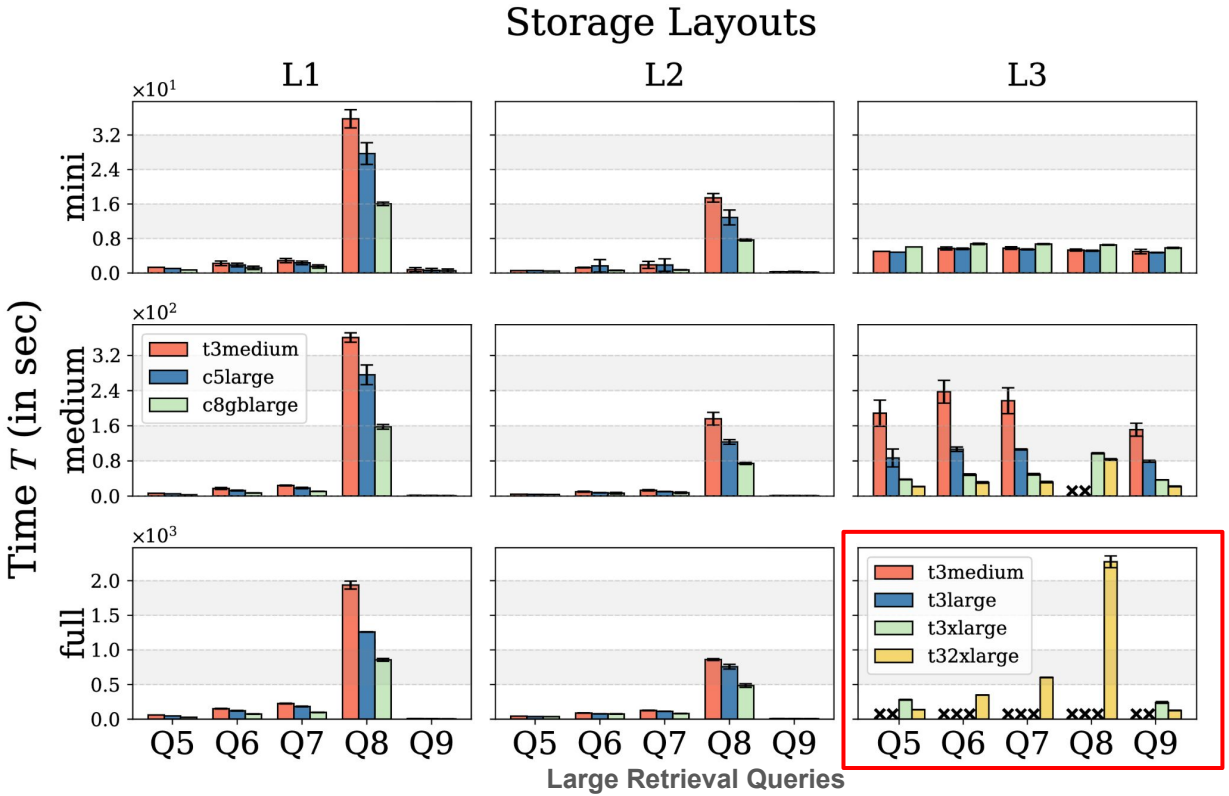
End-to-end retrieval time (Q5-Q9) [Finding #3]



End-to-end retrieval time (Q5-Q9) [Finding #3]



End-to-end retrieval time (Q5-Q9) [Finding #3]



Finding #3: Each Layout Has a Different Bottleneck

Layout	**Bottleneck**	**Fixed by**
L1	Per-request latency and network bandwidth	Can't easily fix
L2	Network bandwidth	More bandwidth
L3	Memory	Bigger instance (💰)

Finding #4: The Cost Story

Finding #4: The Cost Story

- Cost breakdown on *t3.medium*

	L1	L2	L3
C_{network}	\$0.795	\$0.795	\$9.183
C_{compute}	\$0.010	\$0.005	\$0.019
$C(total)$	\$0.805	**\$0.800**	\$9.202

Summary: Characterization Table

Summary: Characterization Table

Characteristic	L1	L2	L3
Primary bottleneck	Latency	Bandwidth	Memory
Connection overhead	High	Low	N/A
Predicate pushdown	✗	✗	✓
Min. instance (full)	t3.medium	t3.medium	t3.2xlarge
Scales with bandwidth	✓	✓	✗

Summary: Characterization Table

Characteristic	L1	L2	L3
Primary bottleneck	Latency	Bandwidth	Memory
Connection overhead	High	Low	N/A
Predicate pushdown	✗	✗	✓
Min. instance (full)	t3.medium	t3.medium	t3.2xlarge
Scales with bandwidth	✓	✓	✗

Recommendation: L2 for images – best latency-cost balance for image workloads