CloudJump: Optimizing Cloud Database For Cloud Storage

Zongzhi Chen, Xinjun Yang, Feifei Li, Xuntao Cheng, Qingda Hu, Zheyu Miao, Rongbiao Xie, Xiaofei Wu, Kang Wang, Zhao Song, Haiqing Sun, Zechao Zhuang, Yuming Yang, Jie Xu, Liang Yin, Wenchao Zhou, Sheng Wang

1 **Background and Motivation**

2 Design Considerations

3 Case Study: PolarDB

4 Case Study: RocksDB

Background and Motivation

VLDB 2022

Alibaba Group 阿里巴巴集团

Cloud-native database

Massive amounts of data

Elasticity

High availability and durability

High performance

Serverless、pay-as-you-go

…

Cloud-storage

Large storage capability

Data persistence

High availability

High aggregated I/O bandwidth

On-demand pricing

Reduce maintenance costs

…

Target: Can we build a “more” cloud-native database through migrating an on-premise database kernel onto the cloud using a cloud storage?

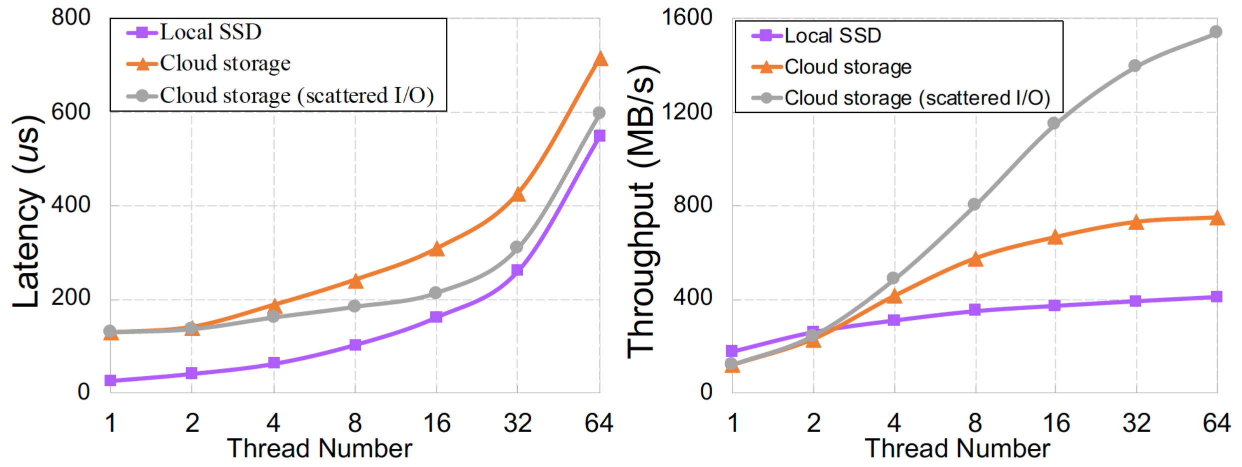
Background and Motivation

- Experience from our online service
  - Slow SQL with cloud storage
  - Low bandwidth utilization
  - Bad log performance when flushing dirty pages

  …

- Micro-benchmark
  - High I/O lantency and bandwidth

  …



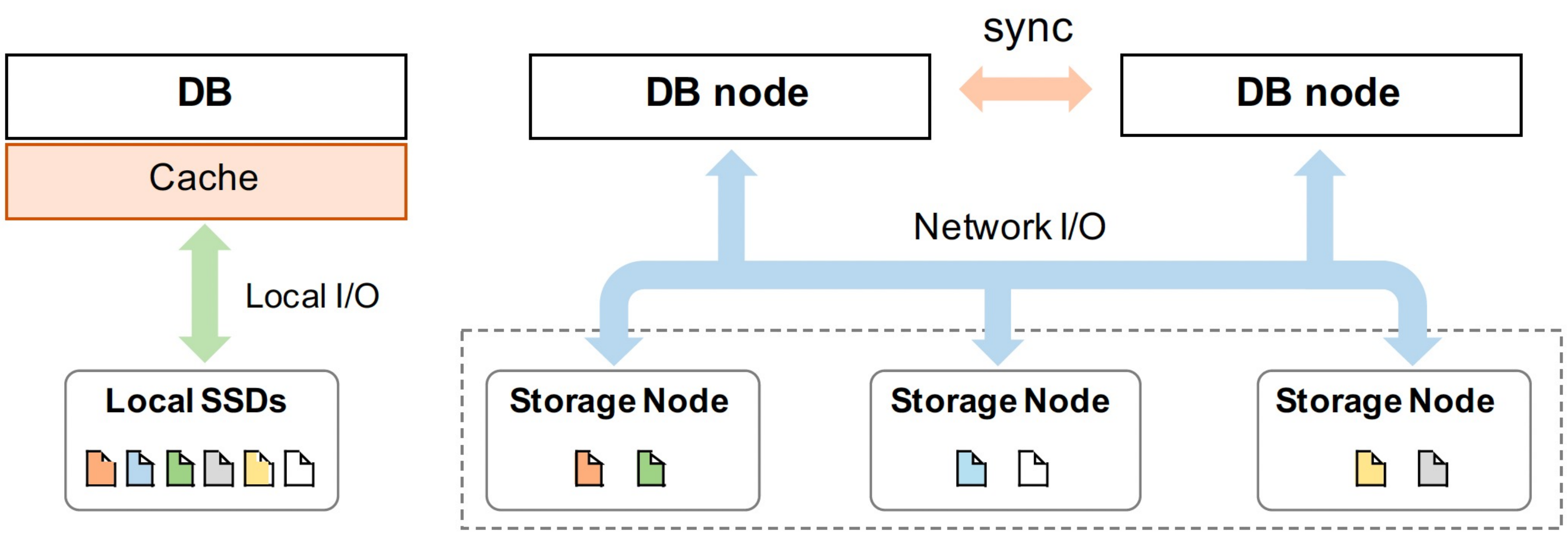
(a) I/O latency v.s. Thread Number (b) Throughput v.s. Thread Number

Figure 2: Single-file write performance of local and distributed storages (*both steady state*).

These hinder cloud storage from becoming an performance-satisfied service for cloud-native databases

Background and Motivation

- Architecture differences in on-premise and on-cloud-storage database

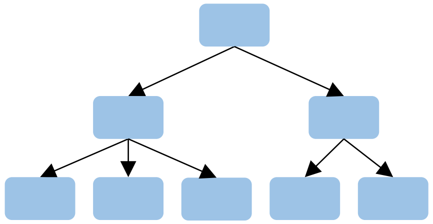


Challenges:

- Local accesses v.s. remote accesses
- Local bandwidth v.s. aggregated bandwidth
- Consistency among multiple database nodes
- I/O isolation
- Big table further worsen the performance

Background and Motivation

B-tree Based
(Update-in-place)



LSM-tree Based
(Append-only)

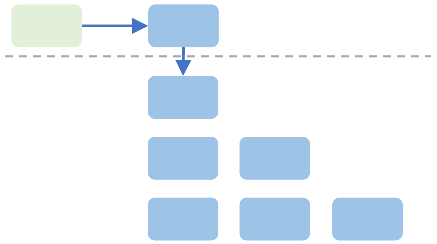


Table : impacts on the design knobs of databases

Challenges	Design knobs	Problems	
		Update-in-place	Append-only
Remote accesses	WAL	Slow serial logging	
	Log replay	Applying logs to multiple pages	Bulk writing of memtables
	Data read	Loading dependent remote pages	Read amplifications
	Synchronization	Blocking updates while writing pages	Compactions with amplified writes & low aggregated utilization
Aggregated bandwidth	Data write	Low bandwidth (accessing a small single page)	
	Data read		amplified reads
Consistency among nodes	Page cache	With cache: high consistency overhead; Without cache: amplified I/O with no buffers	
I/O isolation	I/O scheduling	Concurrent and extensive log and data I/Os cause unpredictable performance	

VLDB 2022

Alibaba Group 阿里巴巴集团

1 Background and Motivation

2 **Design Considerations**

3 Case Study: PolarDB

4 Case Study: RocksDB

Design consideration : optimize on cloud storage

- Thread-level Parallelism

  eg. Adopt multiple logging and data I/O threads, use asynchronous I/O models to fully scatter data across multiple storage nodes

- Task-level Parallelism

  eg. Partitioned log on page-space and written in parallel to multiple tasks. Concurrent Recovery based on partition.

- Reduce remote read and Prefetching

  eg. Prefetching potentially achieves larger performance gains on the cloud storage compared with those on local SSDs,

Design Considerations

Seven design consideration : optimize on cloud storage

- Fine-grained Locking and Lock-free Data Structures

  eg. To minimize the chances of contention during prolonged I/O.

- Scattering among Distributed Nodes

  eg. Distribute large log I/Os to different storage nodes to make full use of the aggregated bandwidth.

- Bypassing Caches

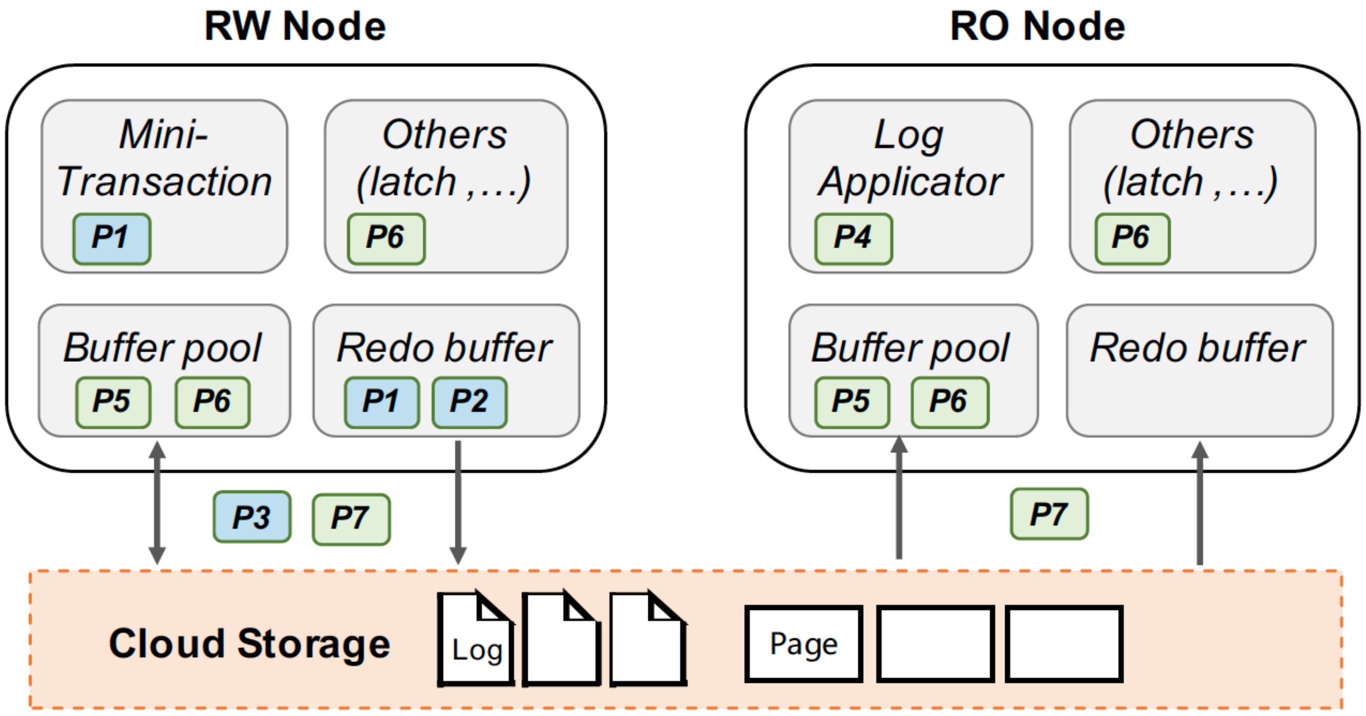
  eg. Avoid the coherence issue and optimize I/O formats on database layer.

- Scheduling Prioritized I/O Tasks

  eg. Marking and scheduling priorities for different I/Os on database layer.

Case Study：PolarDB

- B-tree based storage engine
- Multiple computation nodes



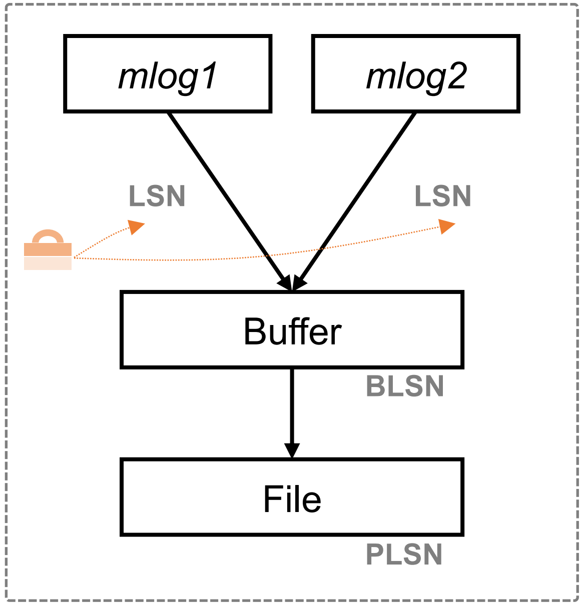
RW	P1 Partitioning Global Log Buffer	P2 Parallel Log Writer
	P3 Multiple I/O Queues and Scheduler	
RW & RO	P4 Fast Recovery	P5 Prefetching
	P6 Fine-grained Locking	P7 Aligned I/O

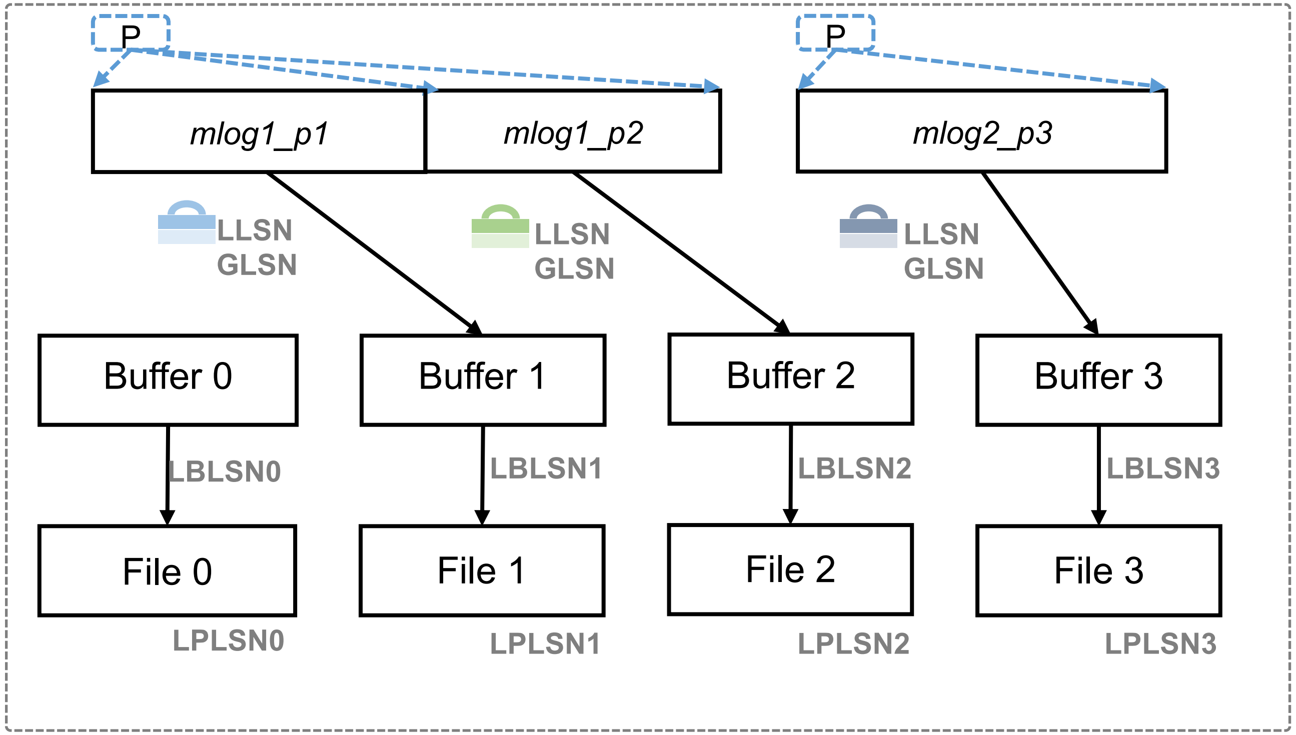
Case Study：PolarDB

Scattered & Partitioned Global Log

- ✗ High WAL I/O Latency
- ✗ Sequential WAL I/O
- ✗ Low bandwidth utilization

- ✓ Log buffer partition，Parallelized writing
- ✓ Asynchronous multi-task threads, high bandwidth utilization
- ✓ Scattered I/O with high distributed writing performance

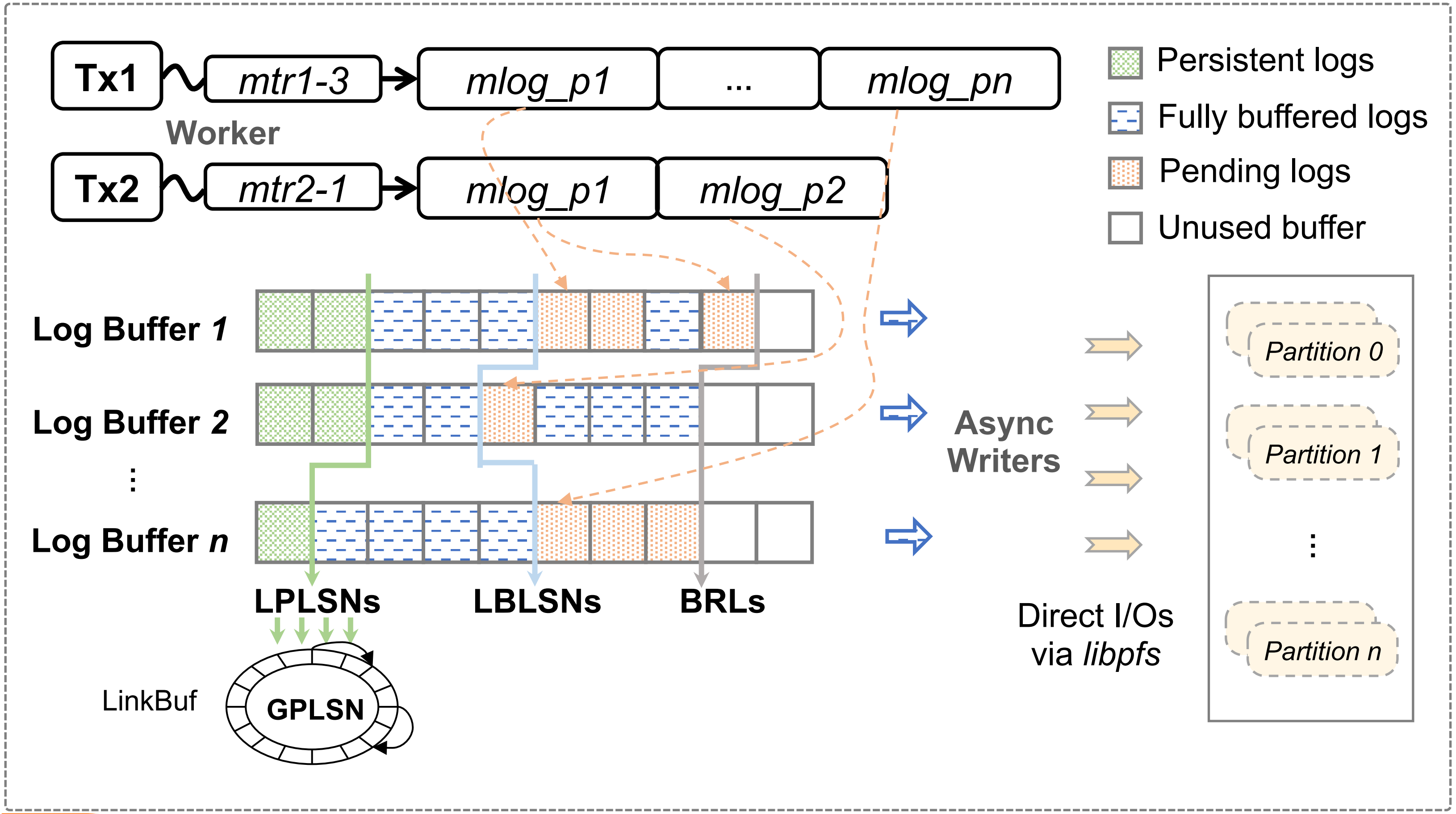


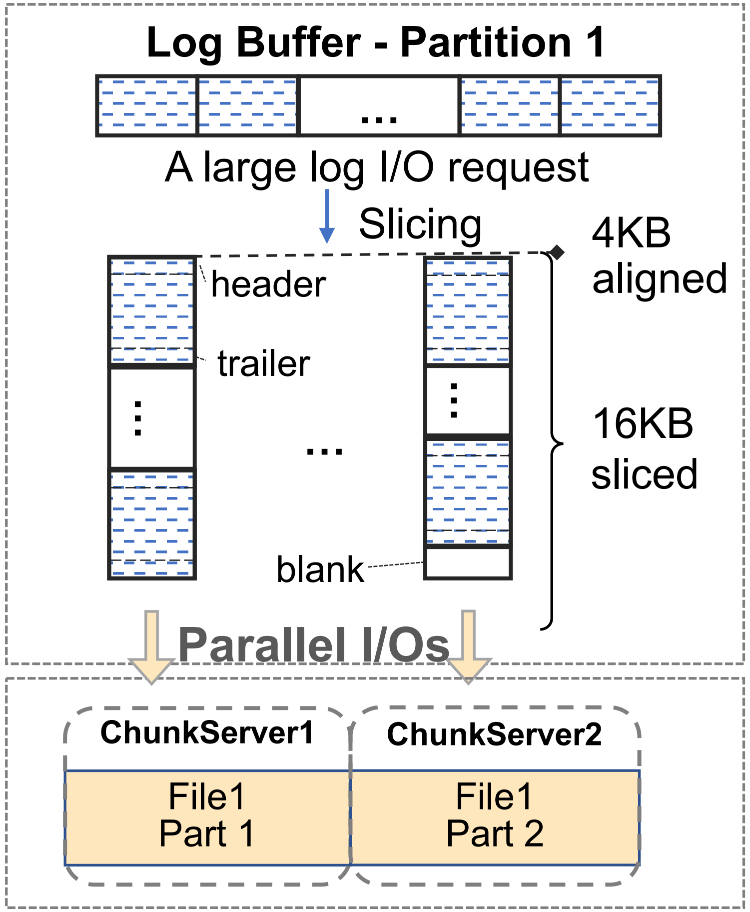


Case Study：PolarDB

VLDB 2022 Alibaba Group 阿里巴巴集团

Scattered & Partitioned Global Log

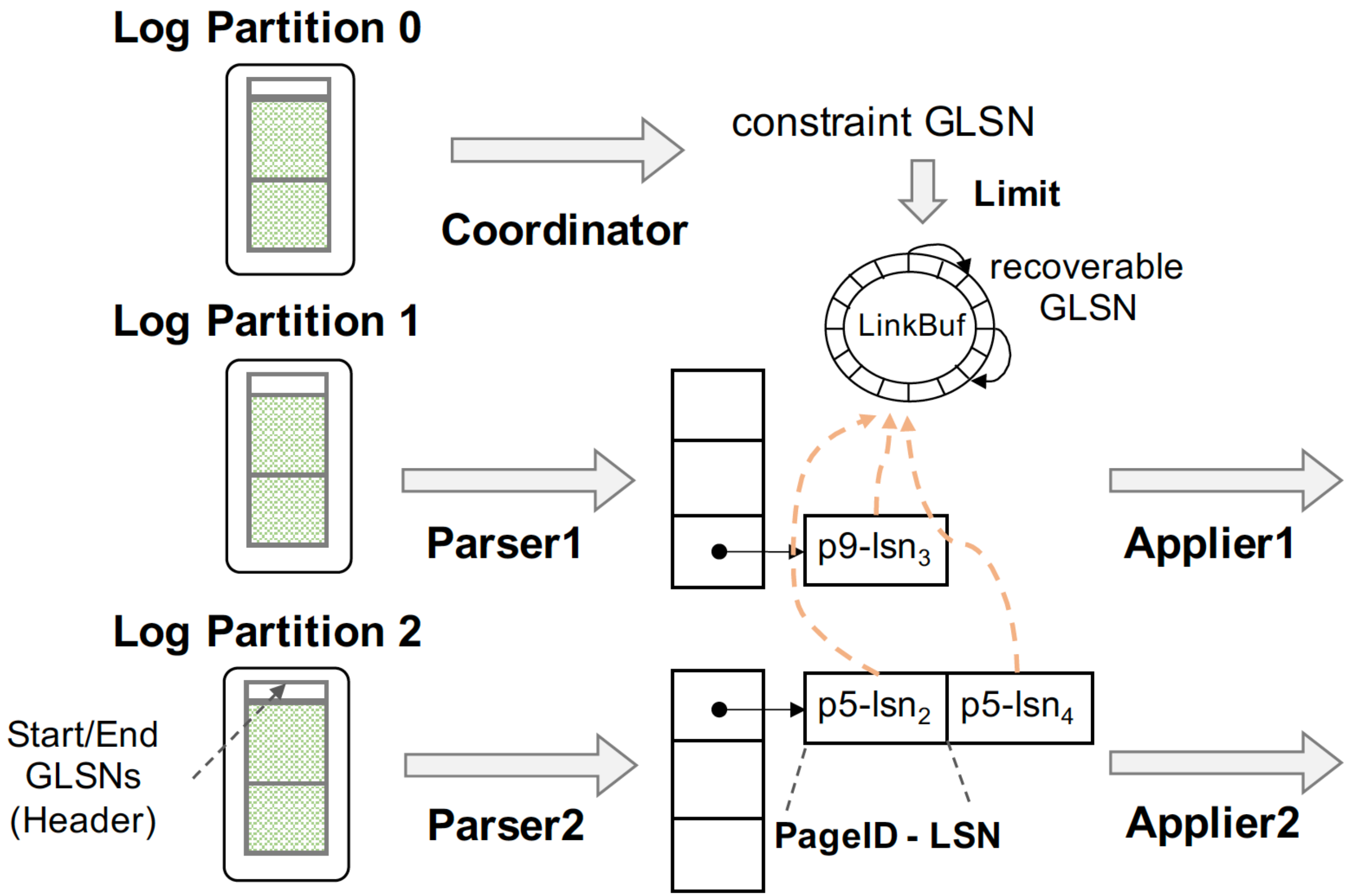




◻ Case Study : PolarDB

VLDB 2022 Alibaba Group 阿里巴巴集团

Parallel Recovery

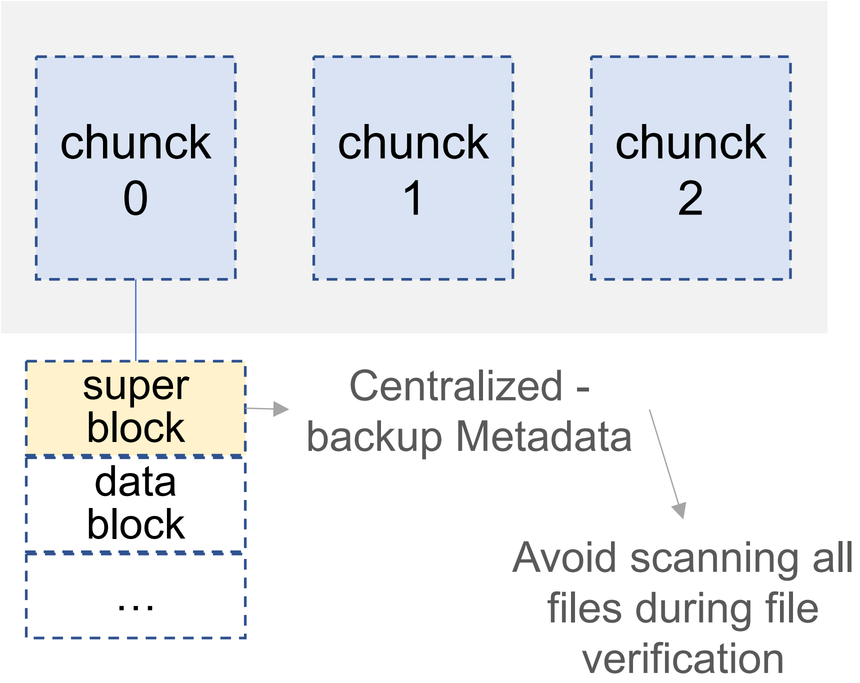


✓ Multi-task concurrent recovery / log application based on Log Partition

Case Study : PolarDB

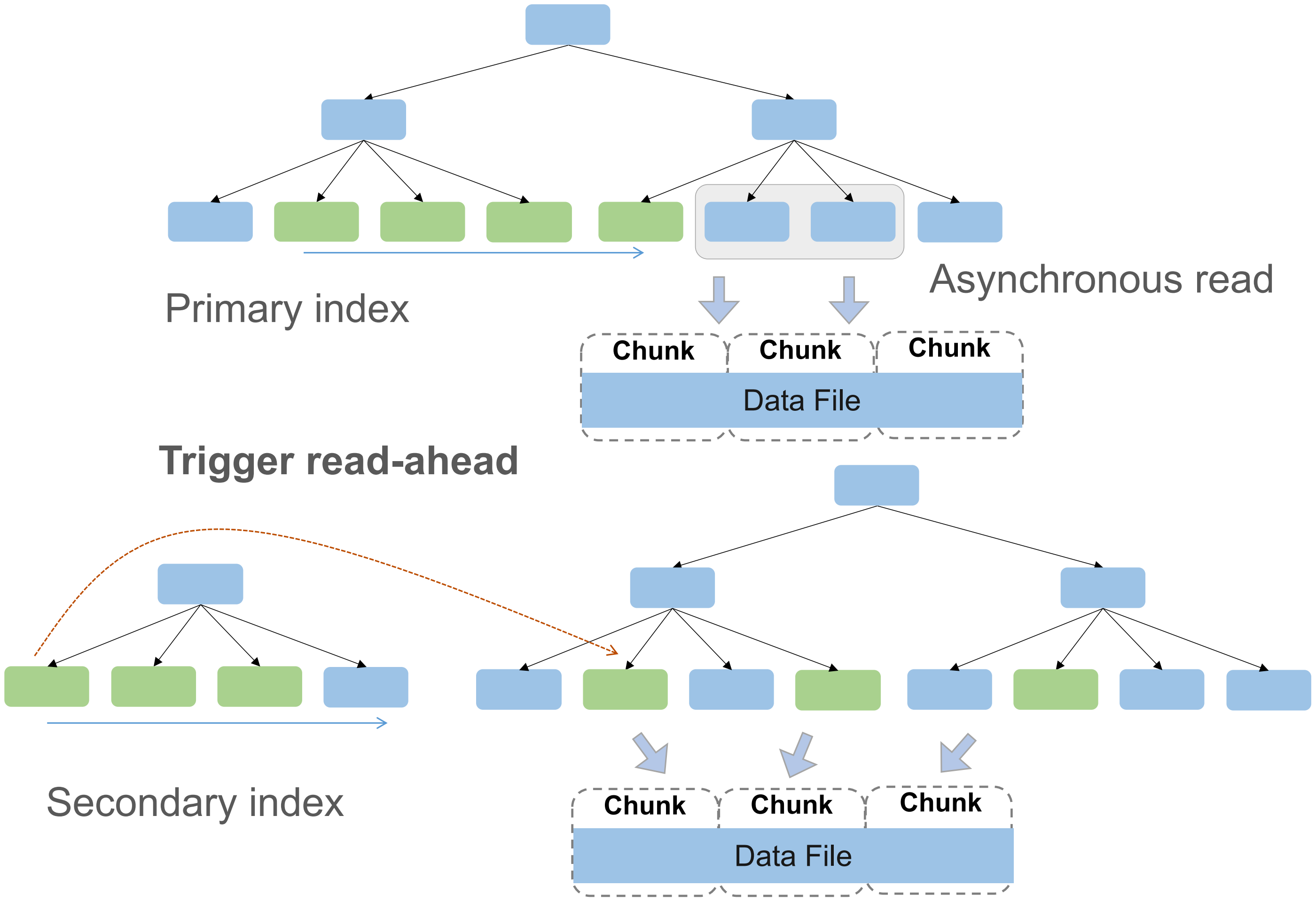
VLDB 2022 · Alibaba Group 阿里巴巴集团

Fast Validating



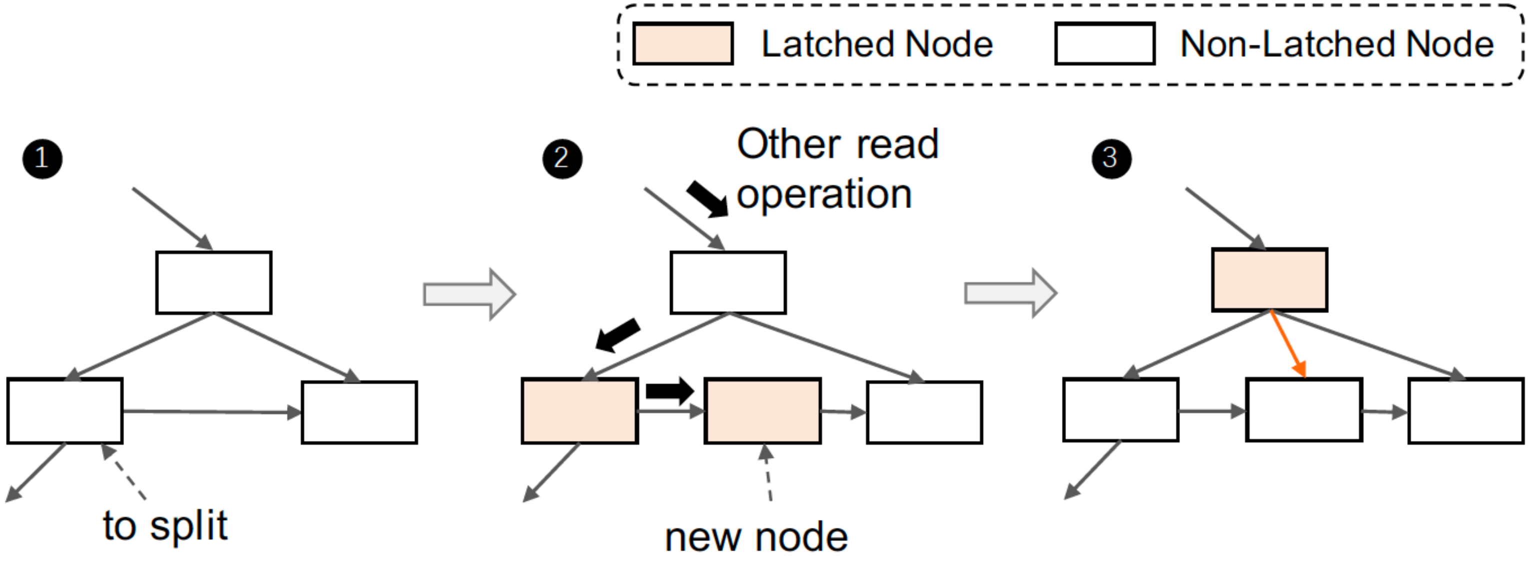
✓ **Reduce the access to remote storage during startup**

Logical Prefetch



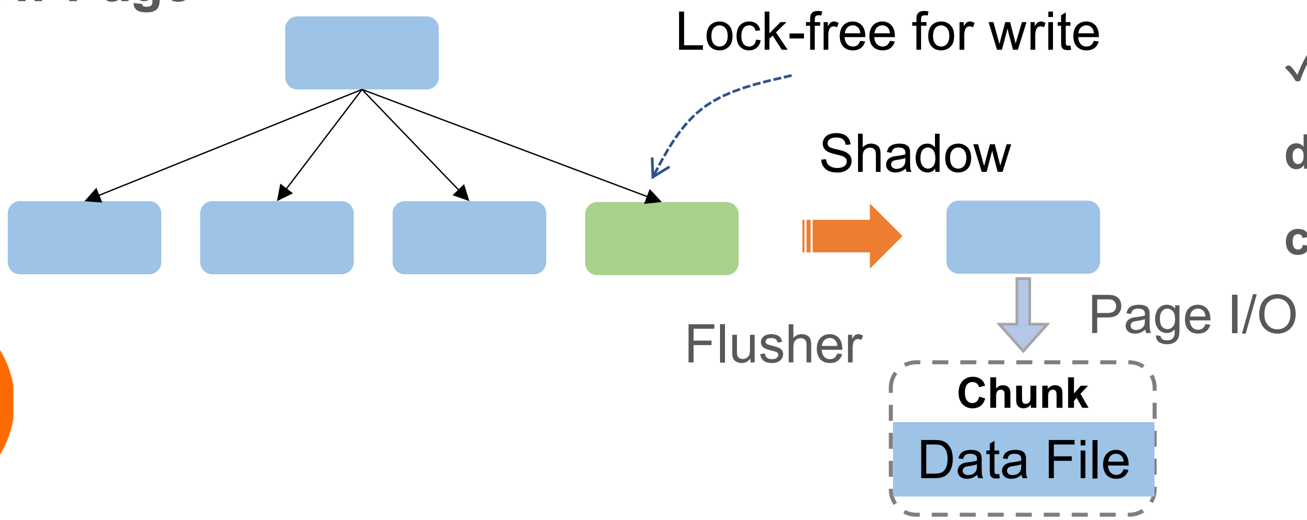
✓ **Utilize aggregate bandwidth to reduce read delay**

Lock-optimized B-tree Index



✓ Remove redundant locks for operations (eg., SMO) to improve the concurrency of memory and I/O operations

Shadow Page



✓ Optimize the long locking time during Page I/Os, to improve operation concurrency

I/O Alignment & Scheduler

- For the direct I/O as bypassing the Cache of distributed file system
  - ✓ Align the optimal I/O **offset & length** to accelerate the direct I/O
  - ✓ Remove **invalid I/O merge** and perform random write
  - ✓ Adopt multi-asynchronous **I/O task queue**, fully utilize the advantage of high bandwidth
- For the long remote access and low I/O isolation
  - ✓ Adopt I/O **priority scheduling**: prioritizes critical I/Os to eliminate low isolation effects

Case Study：PolarDB

VLDB 2022 · Alibaba Group 阿里巴巴集团

Experimental Results

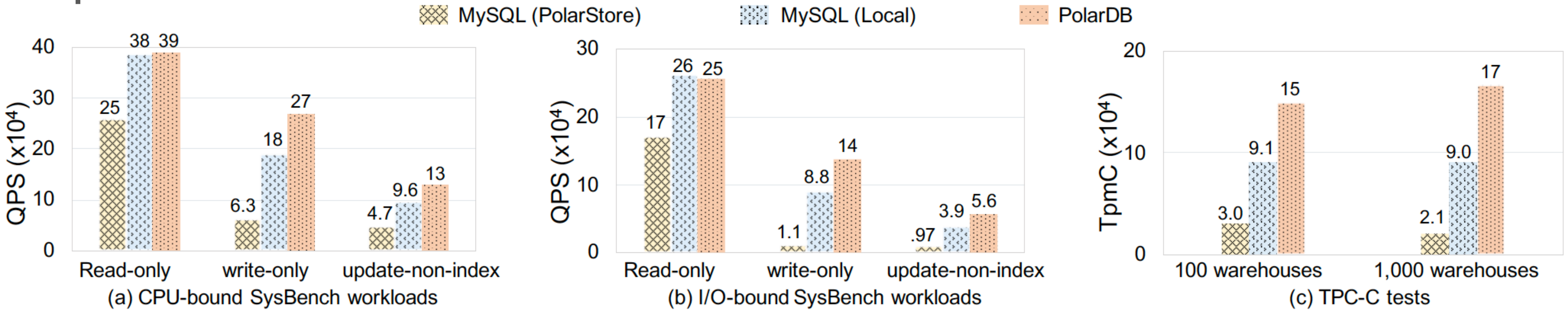


Figure 10: Total performance evaluation of PolarDB.

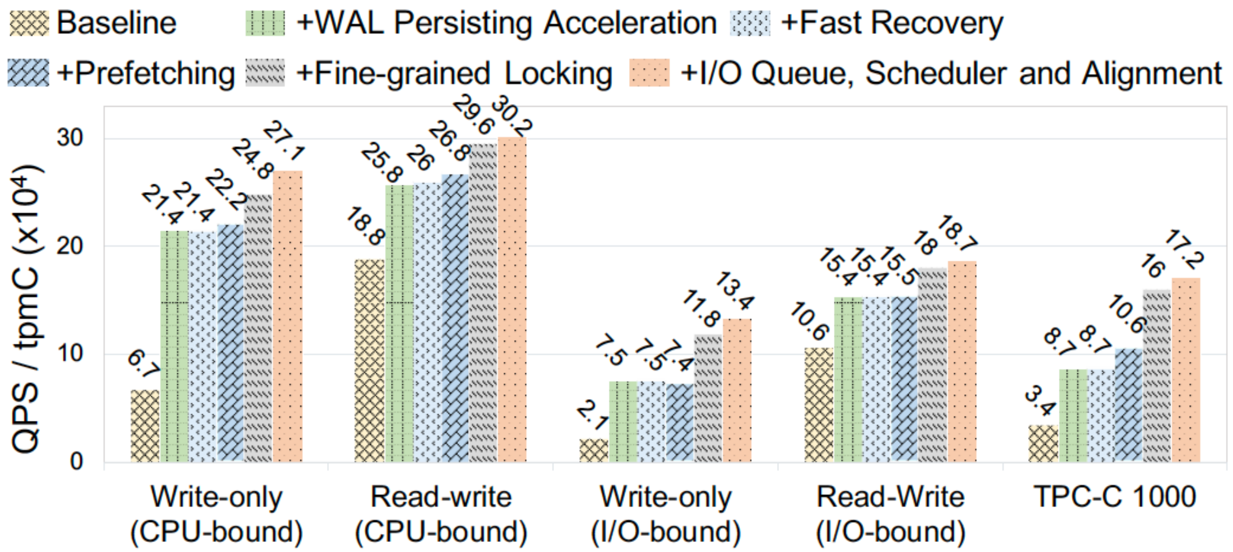


Figure 11: Performance Breakdown.

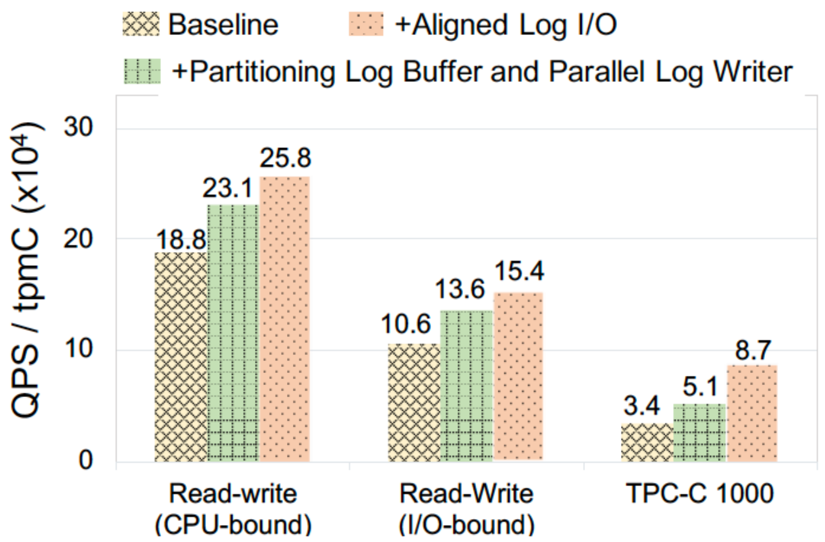
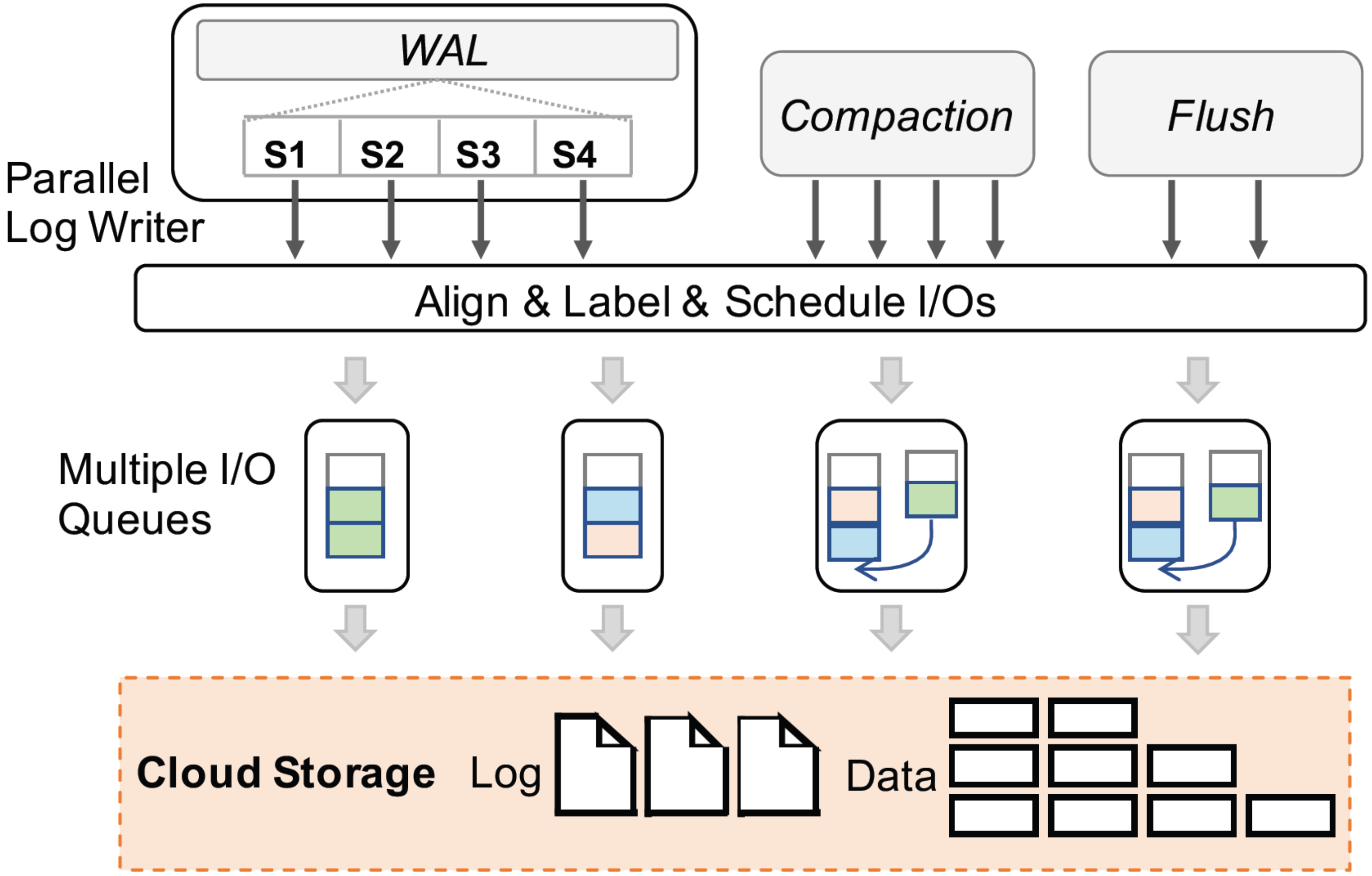


Figure 12: WAL acceleration breakdown.

Case Study：RocksDB



Case Study：RocksDB

Port corresponding optimizations to *RocksDB*

- ✓ Scattered & Partitioned Global Log
- ✓ Scheduled Multi-queue Scatter I/O
- ✓ Direct I/O Alignment

Achieve expected performance gains

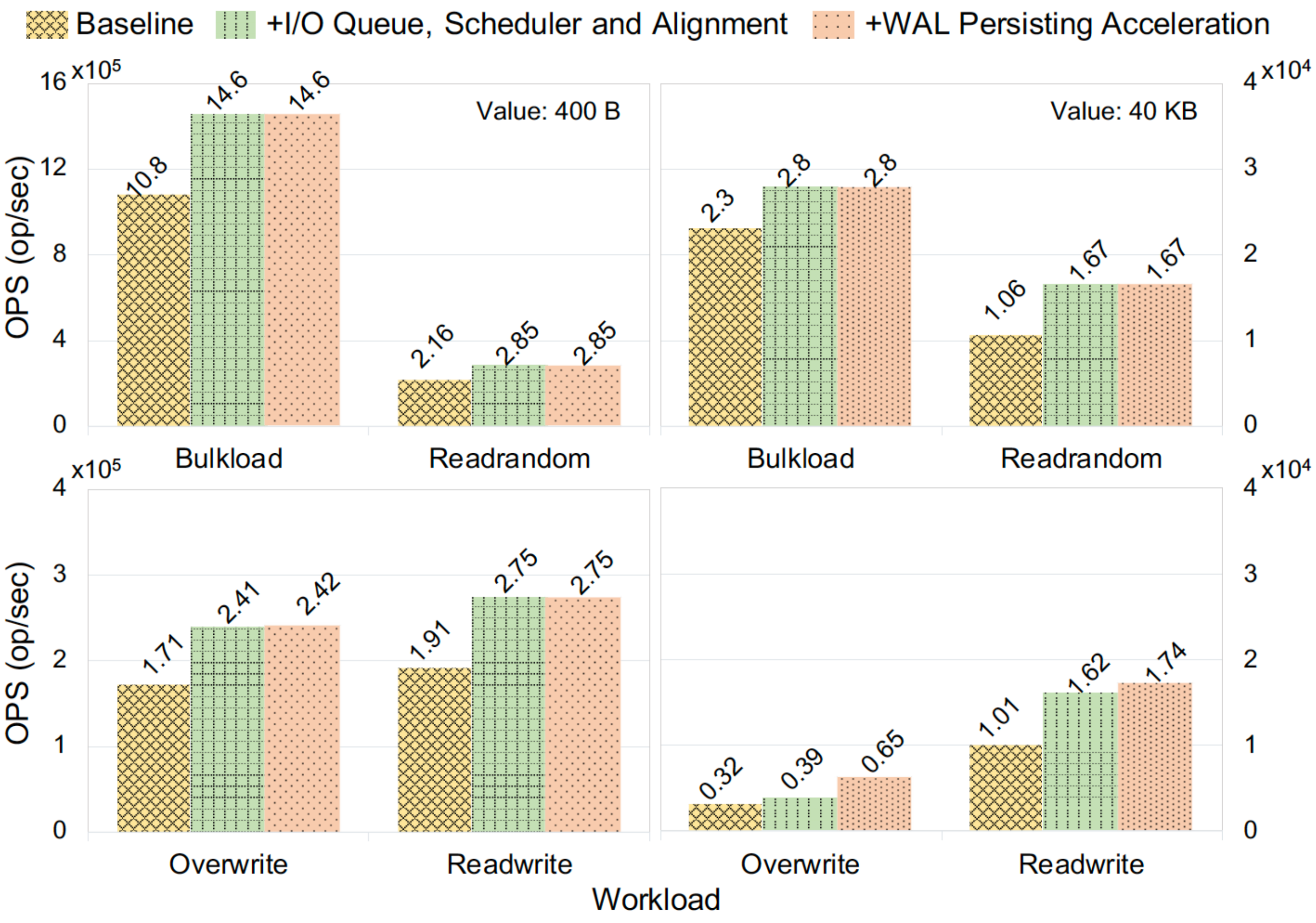


Figure 15: RocksDB Performance.

End

Thanks！