MAP-i Doctoral Program in Computer Science

User-level Software-Defined Storage Data Planes

Ricardo Macedo

Under the supervision of
João Tiago Paulo
José Orlando Pereira

Data-centric systems

- Data-centric systems have become an integral part of modern I/O stacks
- Good performance for these systems requires storage optimizations
  - Scheduling, caching, tiering, …
- Optimizations are implemented sub-optimally

Data-centric systems

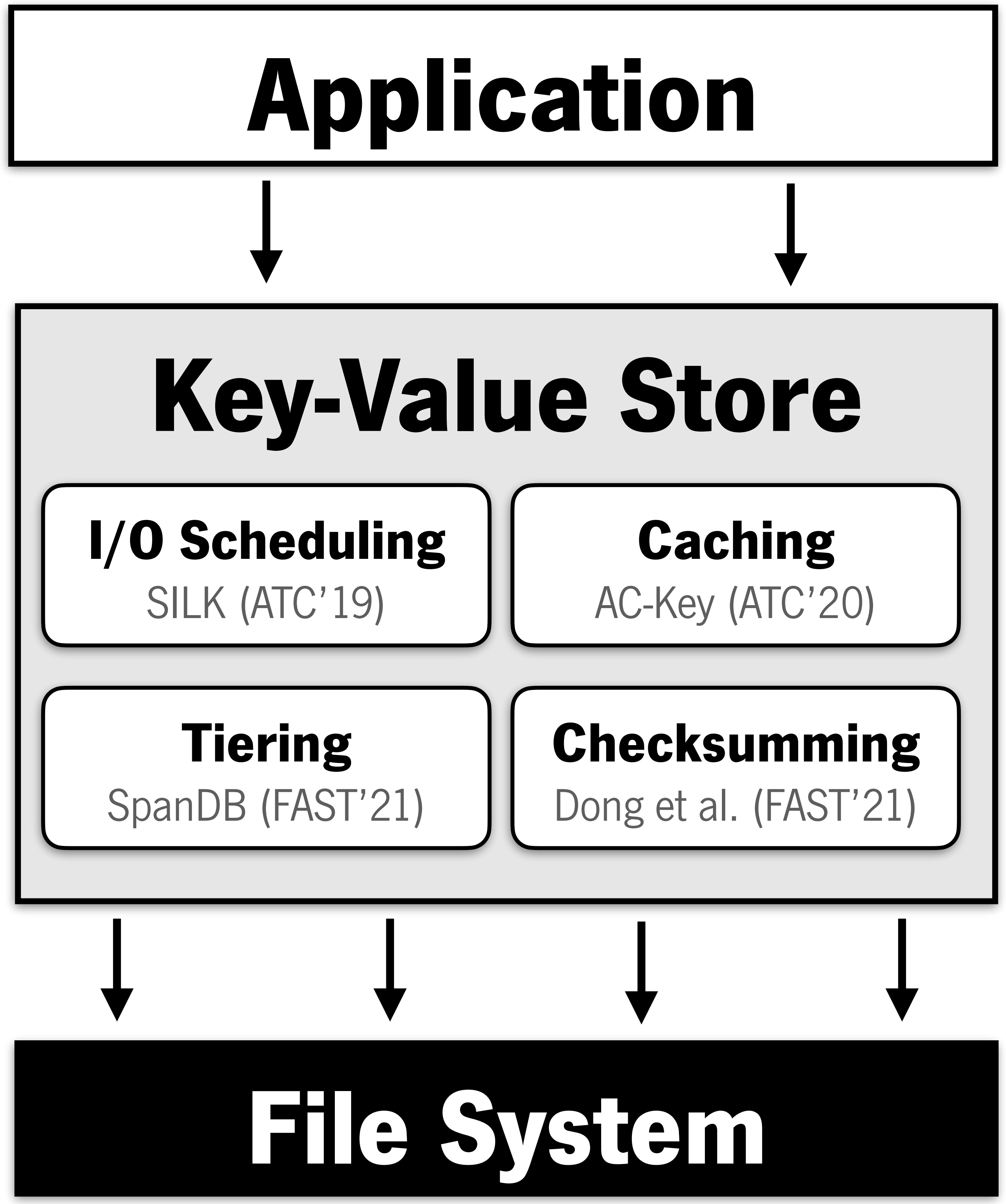
- Data-centric systems have become an integral part of modern I/O stacks
- Good pe
optimizat
  - Sche
- Optimizations are implemented sub-optimally

There is a better way to implement I/O optimizations

Challenge #1

Tightly coupled optimizations

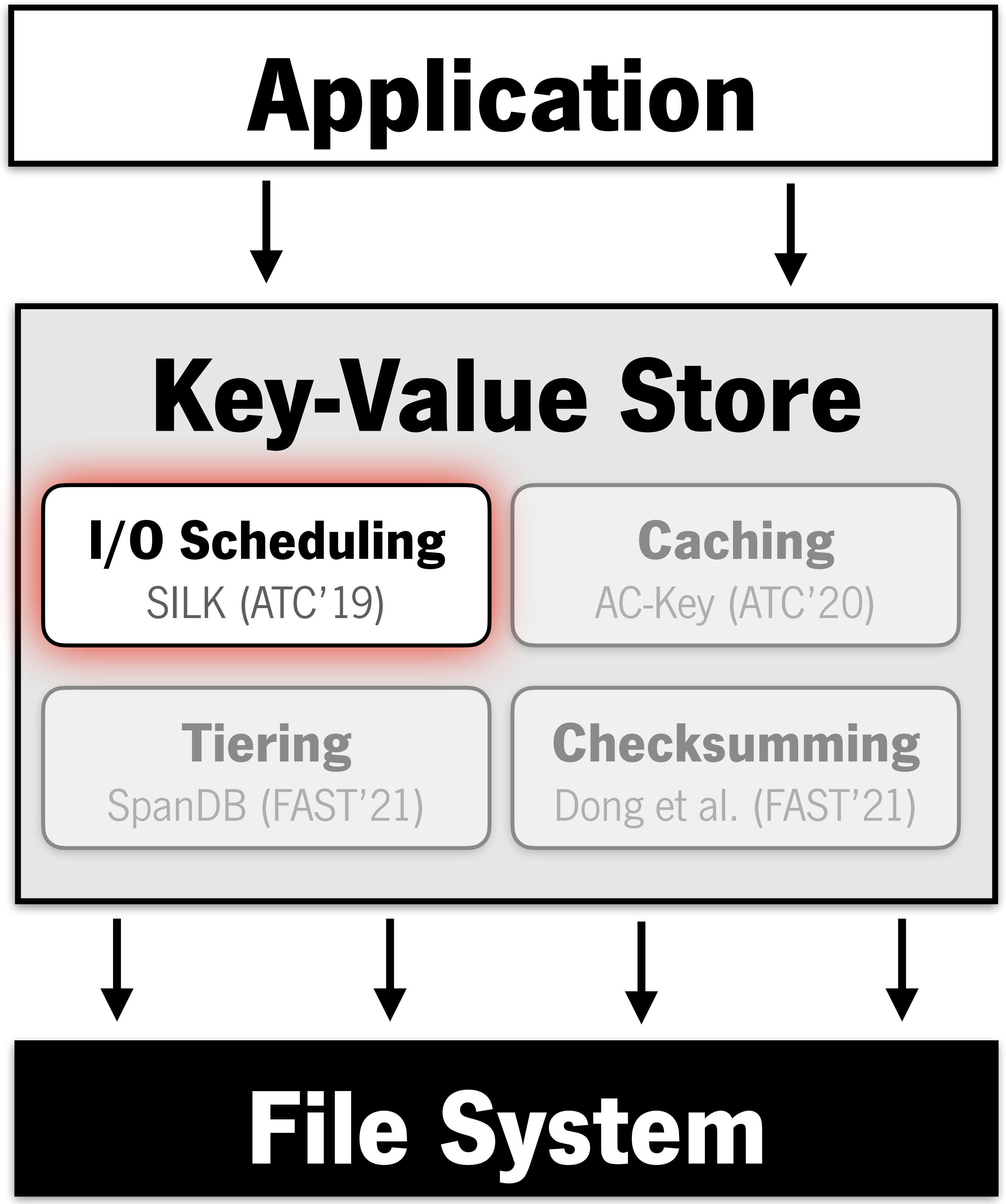
- I/O optimizations are single purposed
- Require deep understanding of the system's internal operation model
- Require profound system refactoring
- Have limited portability across systems



Challenge #1

❌ Tightly coupled optimizations

- I/O optimizations are single purposed
- Require deep understanding of the system's internal operation model
- Require profound system refactoring
- Have limited portability across systems



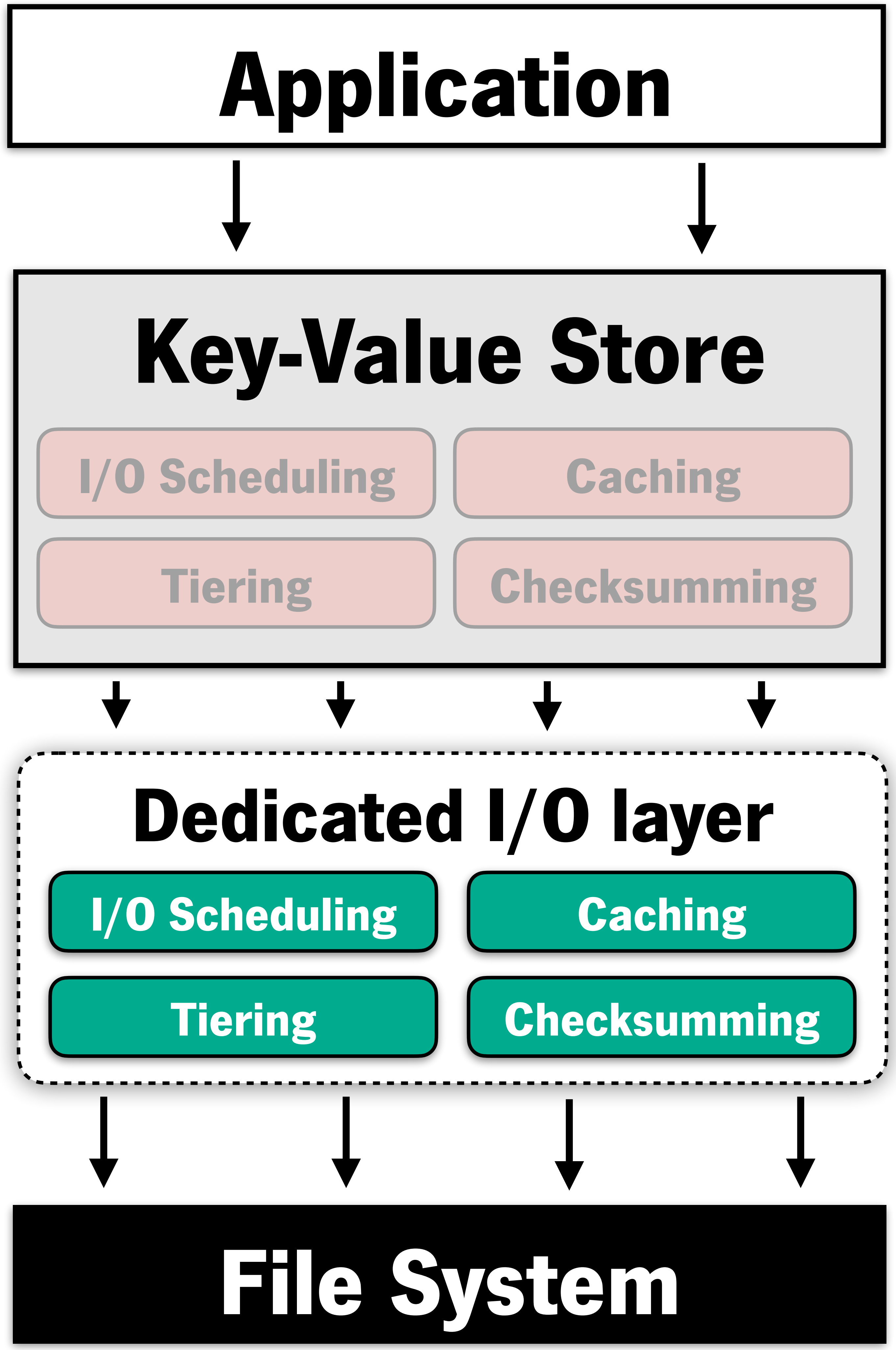
SILK's I/O Scheduler

- Reduces tail latency spikes in RocksDB
- Controls the interference between foreground and background tasks
- Requires changing several modules, such as background operation handlers, internal queuing logic, and thread pools

Challenge #1

Decoupled optimizations

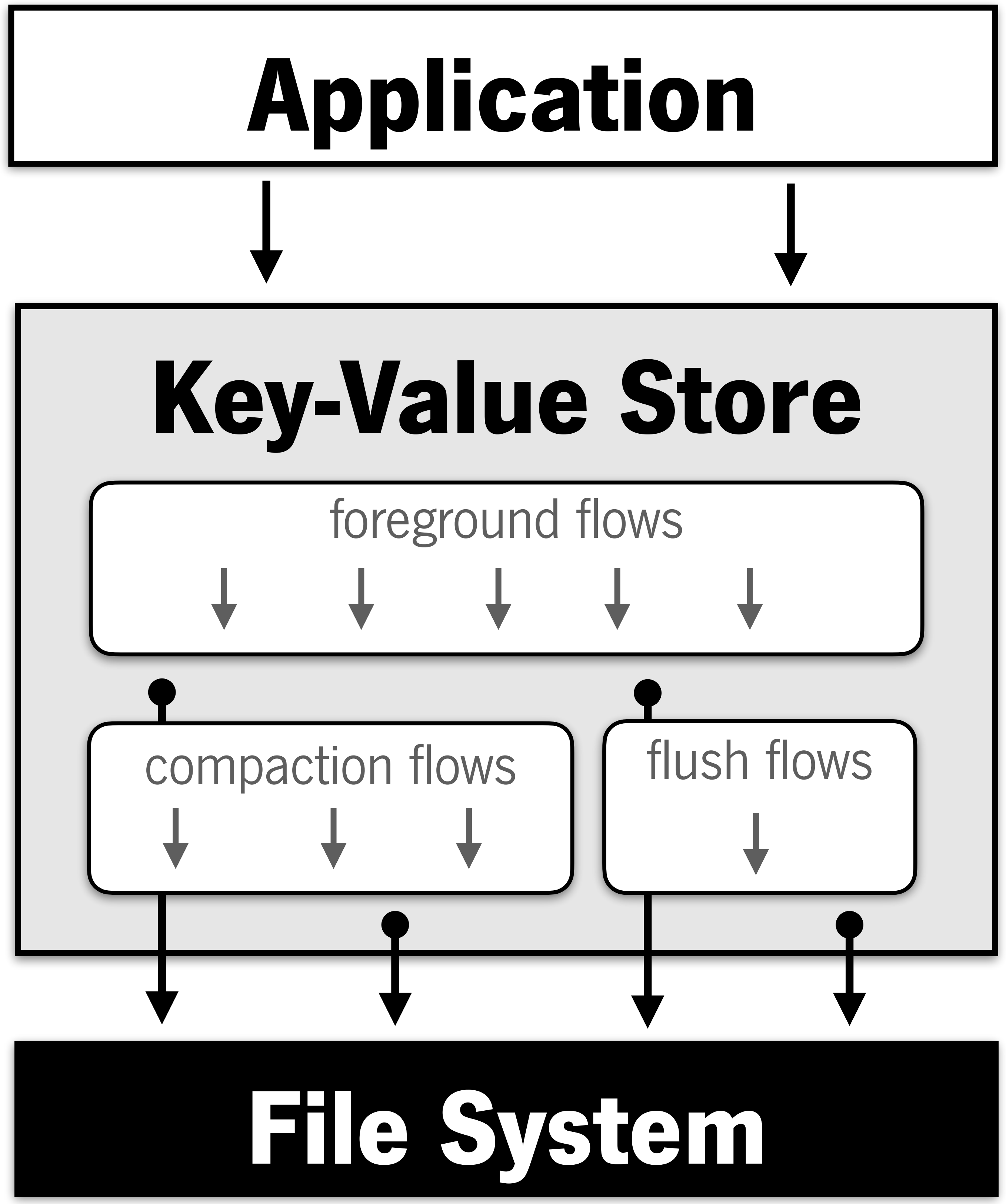
- I/O optimizations should be disaggregated from the internal logic of applications
- Moved to a dedicated I/O layer
- Generally applicable
- Portable across different scenarios



Challenge #2

Rigid interfaces

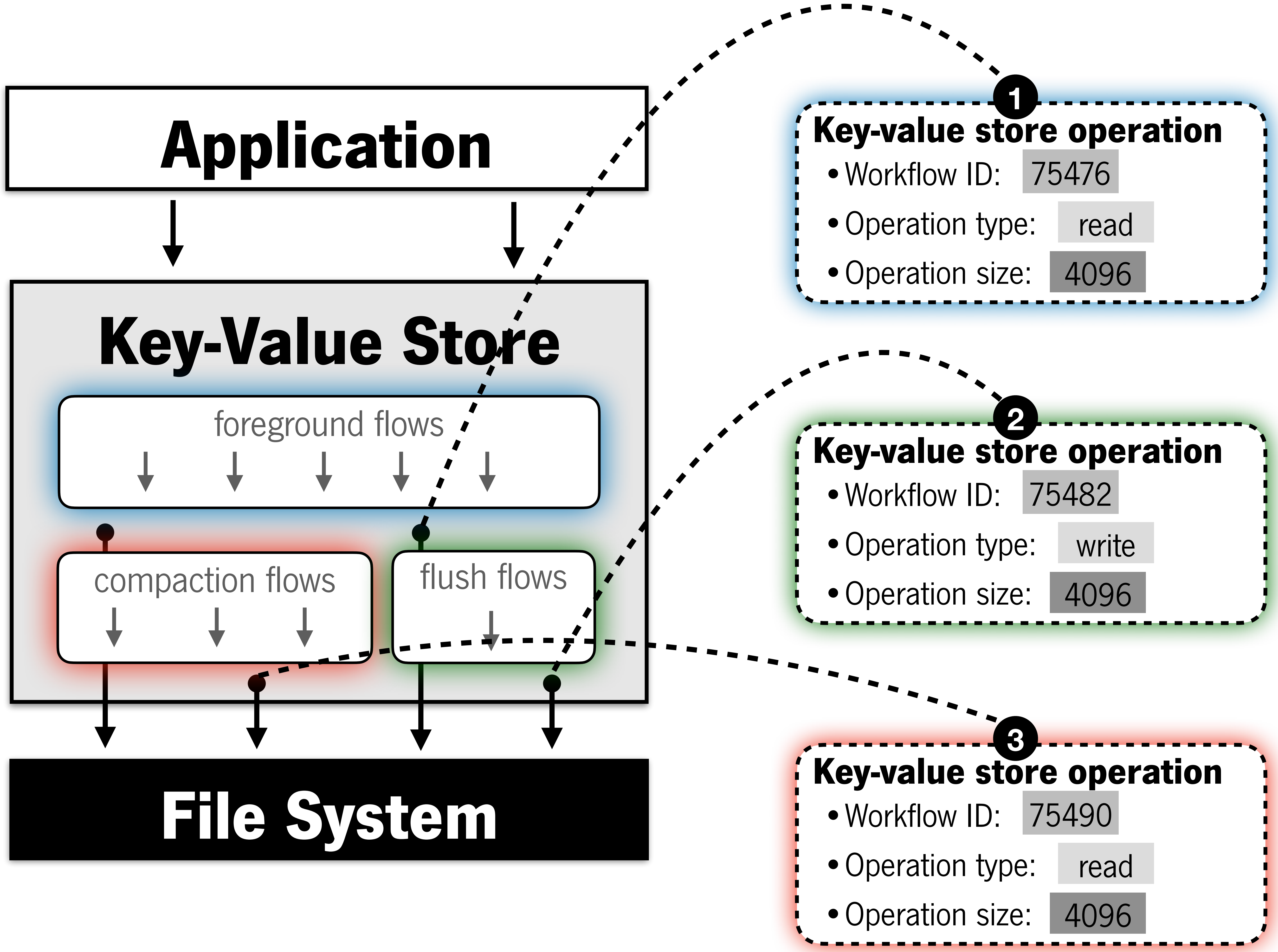
- Decoupled optimizations lose granularity and internal application knowledge
- I/O layers expose rigid interfaces
- Discard information that could be used to classify and differentiate requests



Challenge #2

Rigid interfaces

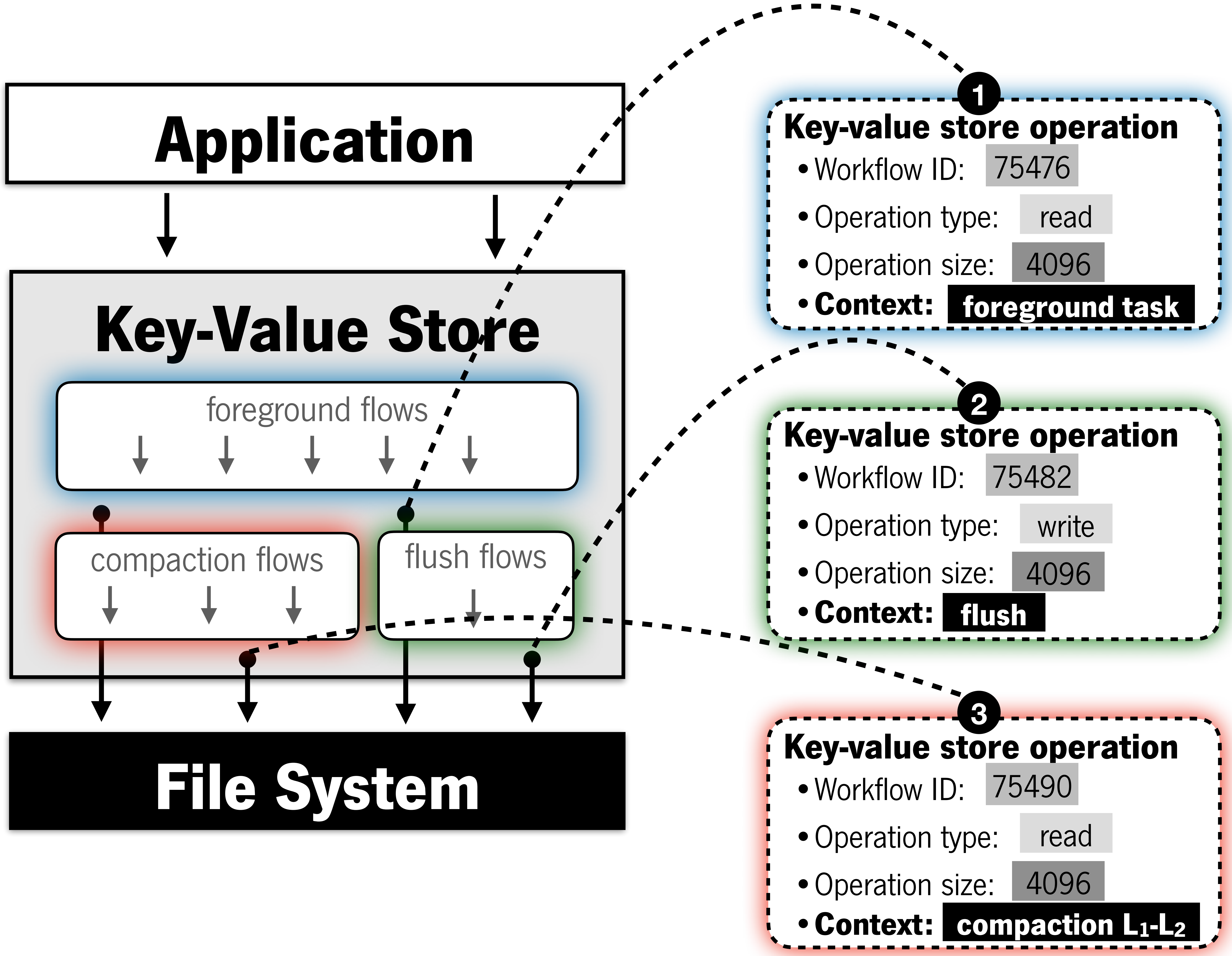
- Decoupled optimizations lose granularity and internal application knowledge
- I/O layers expose rigid interfaces
- Discard information that could be used to classify and differentiate requests



Challenge #2

✓ Information propagation

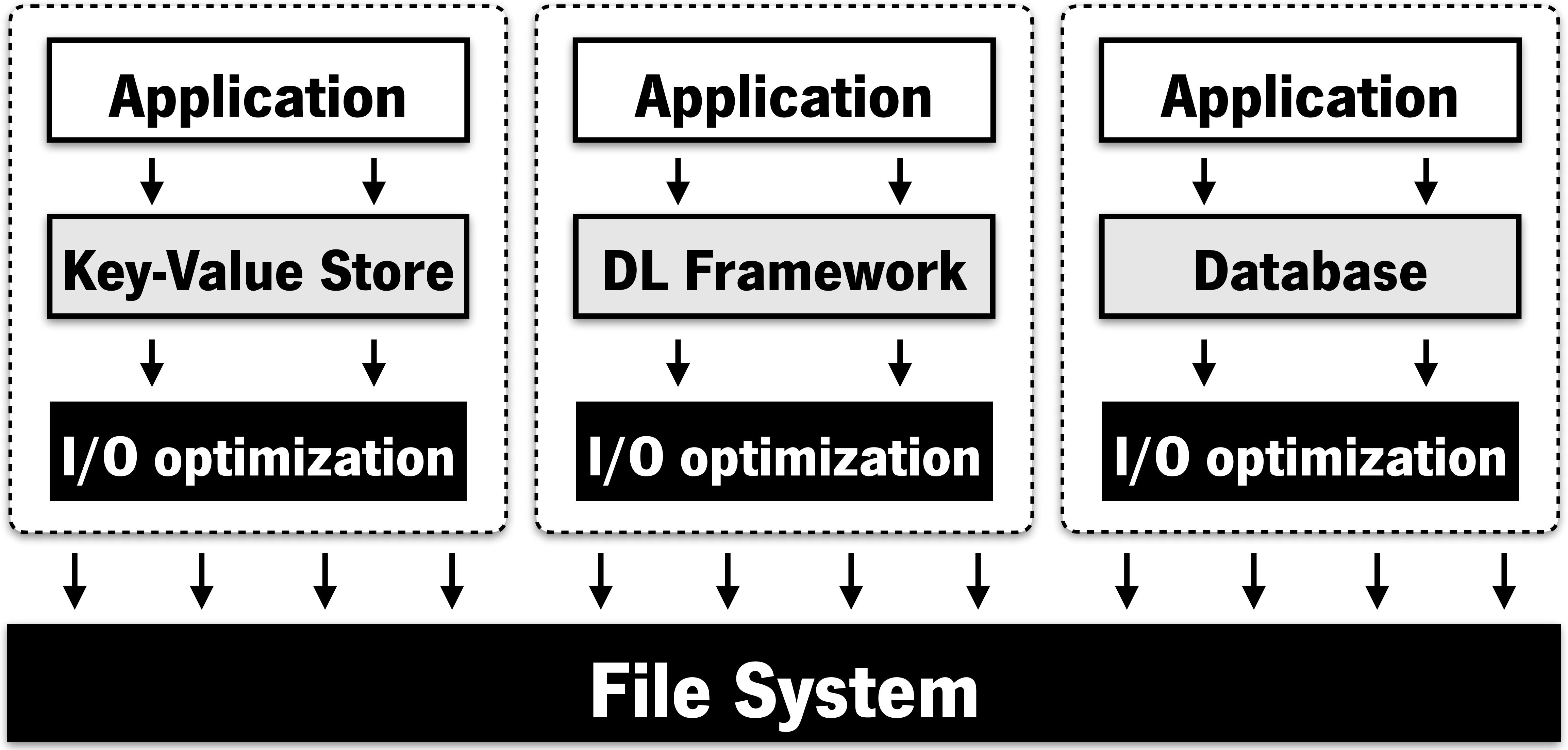
- Application-level information must be propagated throughout layers
- Decoupled optimizations can provide the same level of control and performance



Challenge #3

Partial visibility

- Optimizations are oblivious of other systems
- Lack of coordination
- Conflicting optimizations, I/O contention, and performance variation

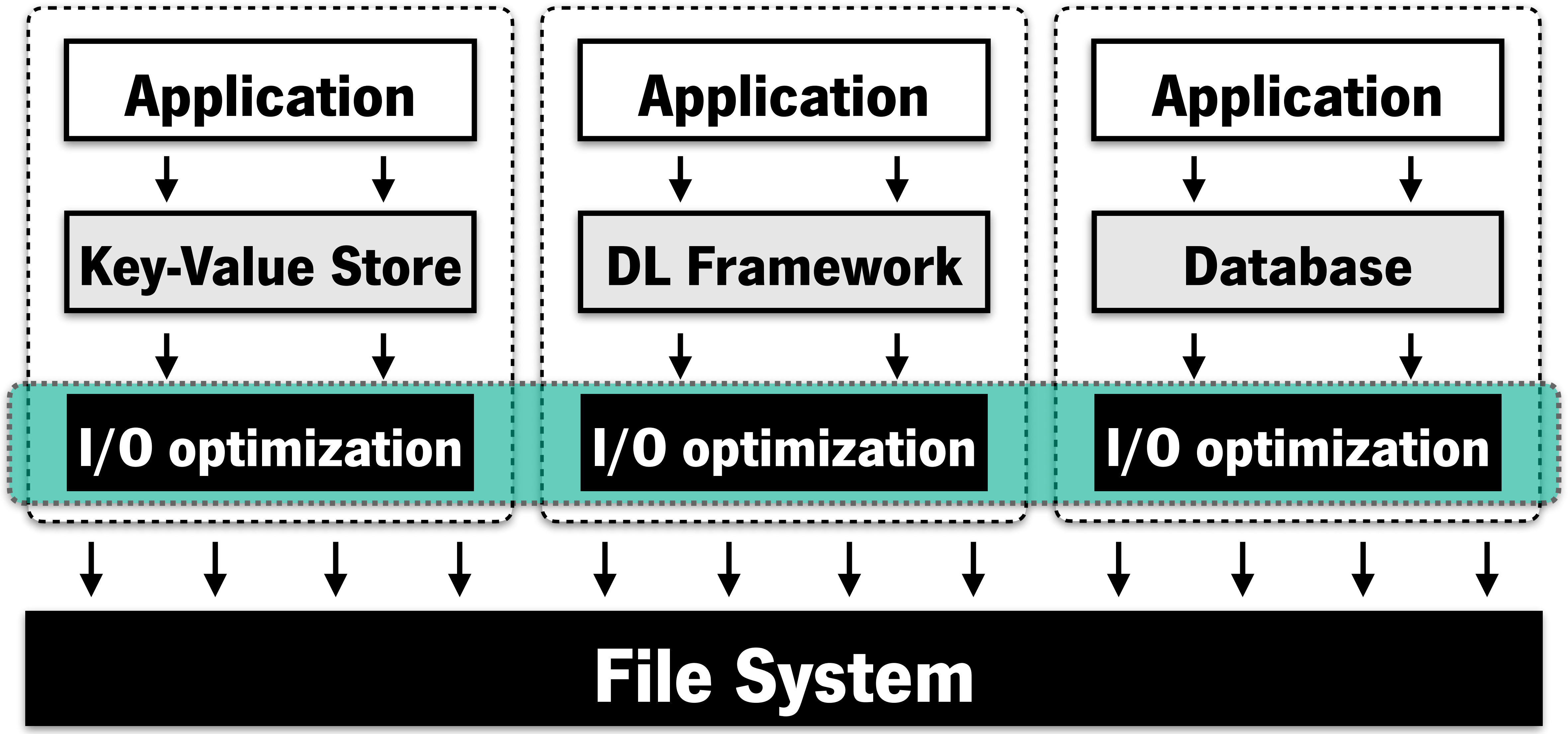


Note: the storage backend can either be local (e.g., ext4, xfs) or distributed (e.g., Lustre, GPFS), as well as the I/O layers on top

Challenge #3

Global I/O control

- Optimizations should be aware of the surrounding system stack
- Operate in coordination
- Holistic control of I/O workflows and shared resources



Note: the storage backend can either be local (e.g., ext4, xfs) or distributed (e.g., Lustre, GPFS), as well as the I/O layers on top

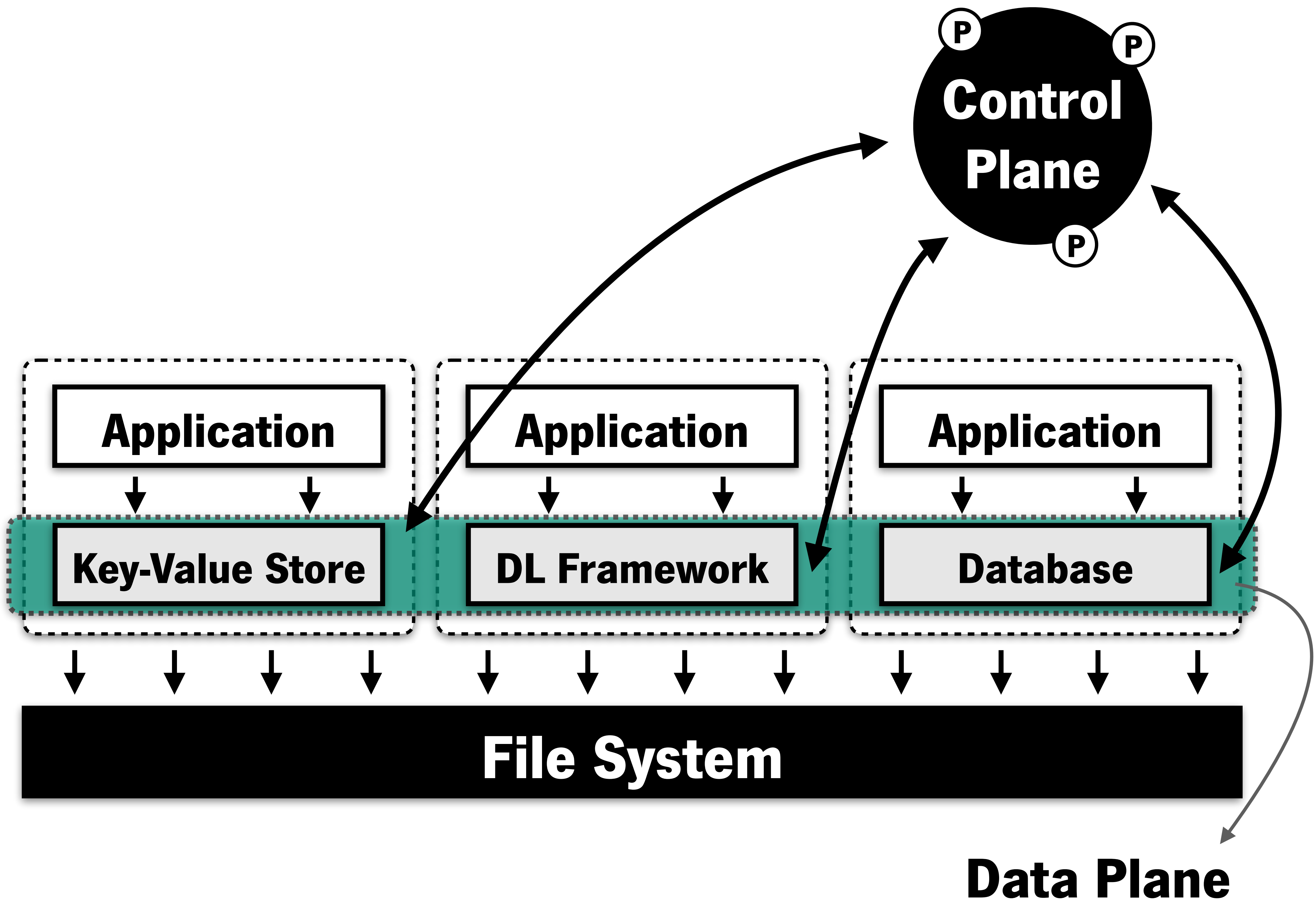
Objectives

Redefine how I/O optimizations are implemented

- Decoupled from the targeted system, minimizing intrusiveness
- Perform coordinated decisions over shared resources
- Impose minimal performance overhead
- Programmable and adaptable to different requirements and storage objectives

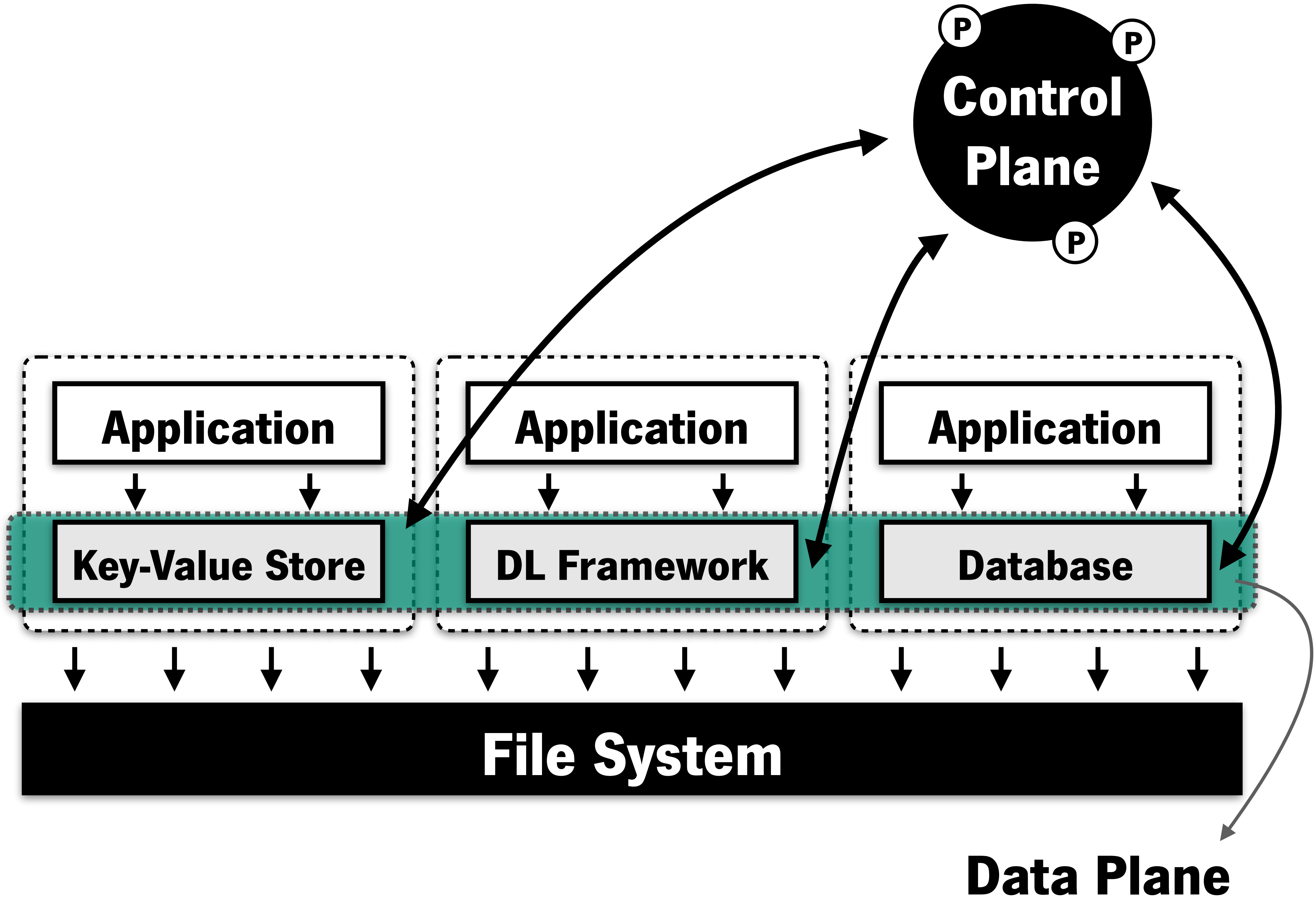
Software-Defined Storage

- **Software-Defined Storage** (SDS) decouples I/O mechanisms from the policies that govern them
- **Control plane** acts as a global coordinator that enforces policies holistically
  - QoS provisioning, performance control, resource fairness
- **Data plane** is a multi-stage component that implements custom I/O logic over requests
  - I/O schedulers, encryption, compression, and caching



Software-Defined Storage

- **Software-Defined Storage** (SDS) decouples I/O mechanisms from the policies that govern them
- **Control plane** acts as a global coordinator that enforces policies holistically
  - QoS provisioning, performance control, resource fairness
- **Data plane** is a multi-stage component that implements custom I/O logic over requests
  - I/O schedulers, encryption, compression, and caching



Survey and classification of SDS systems

- Targeted for **specific I/O layers** or **storage objectives** (e.g., virtualization, file system, resource management)
- Tightly coupled design, driven by the architecture and specificities of the context they are applied
- Existing SDS systems follow a **similar path** as traditionally implemented I/O optimizations

Macedo *et al. "A Survey and Classification of Software-Defined Storage Systems"*. ACM Computing Surveys, 2020.

Contributions

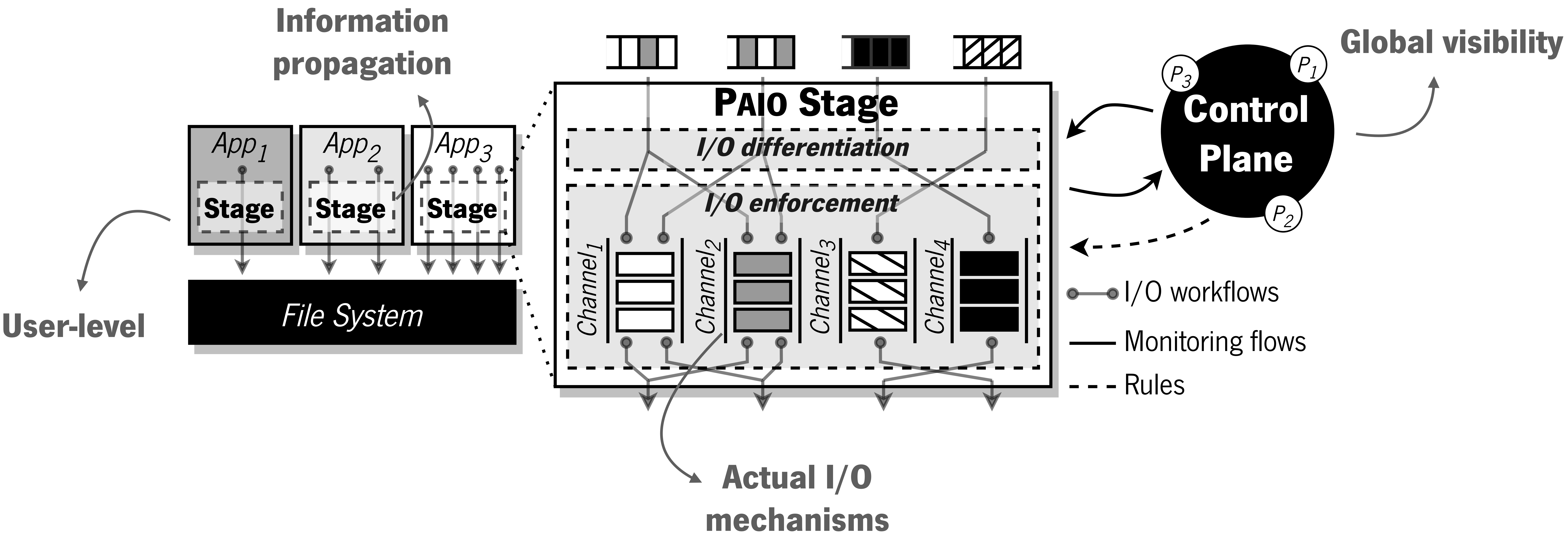
- **Software-Defined Storage survey**
  - Systematization of knowledge, taxonomy, and classification of existing SDS work
- **PAIO** data plane framework
  - Enables building user-level, portable, and generally applicable I/O optimizations
- **Data plane stages** built with PAIO
  - **Tail latency control** in LSM-based key-value stores
  - **Per-application bandwidth control** under shared storage environments
  - **Metadata control** in parallel file systems

PAIO: Programmable and Adaptable I/O Workflows

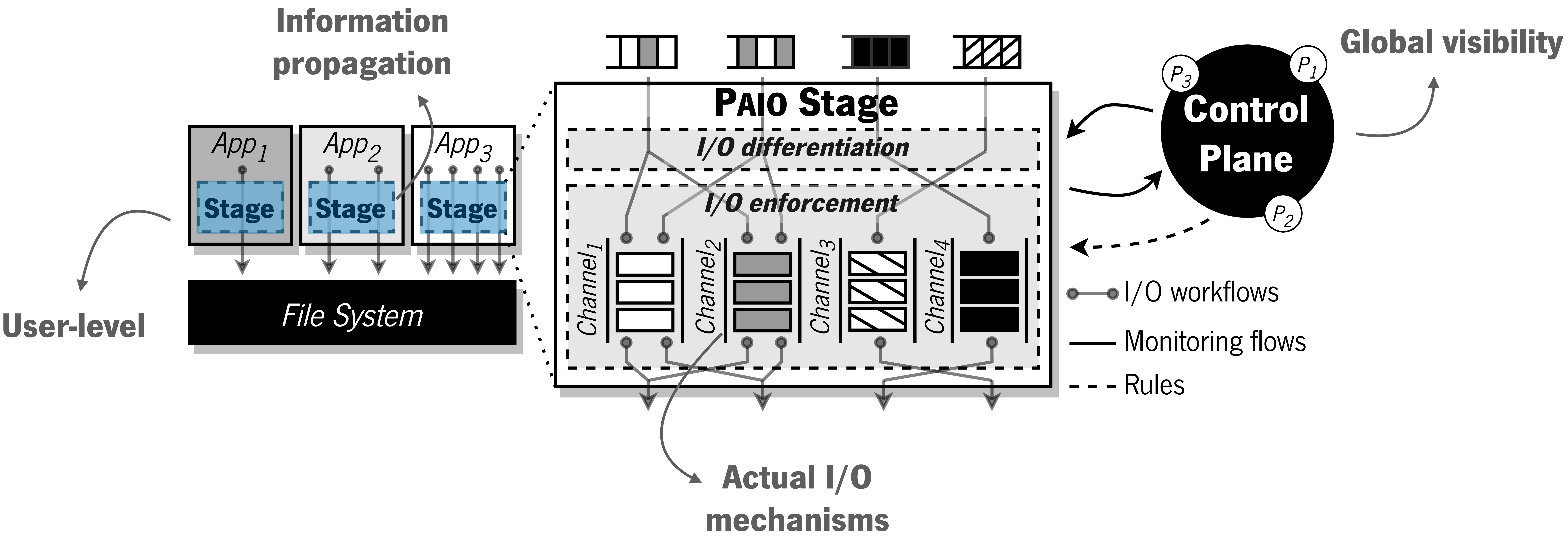
- **User-level** framework for building **portable** and **generally applicable** I/O optimizations
- Follows a Software-Defined Storage design
  - I/O optimizations are implemented **outside** applications as **data plane stages**
  - **Stages** are controlled through a **control plane** for coordinated access to resources
- Enables the propagation of application-level information through **context propagation**
- Porting I/O layers to use PAIO requires **none to minor** code changes

Macedo *et al. "PAIO: General, Portable I/O Optimizations With Minor Application Modifications"*. USENIX FAST, 2022.

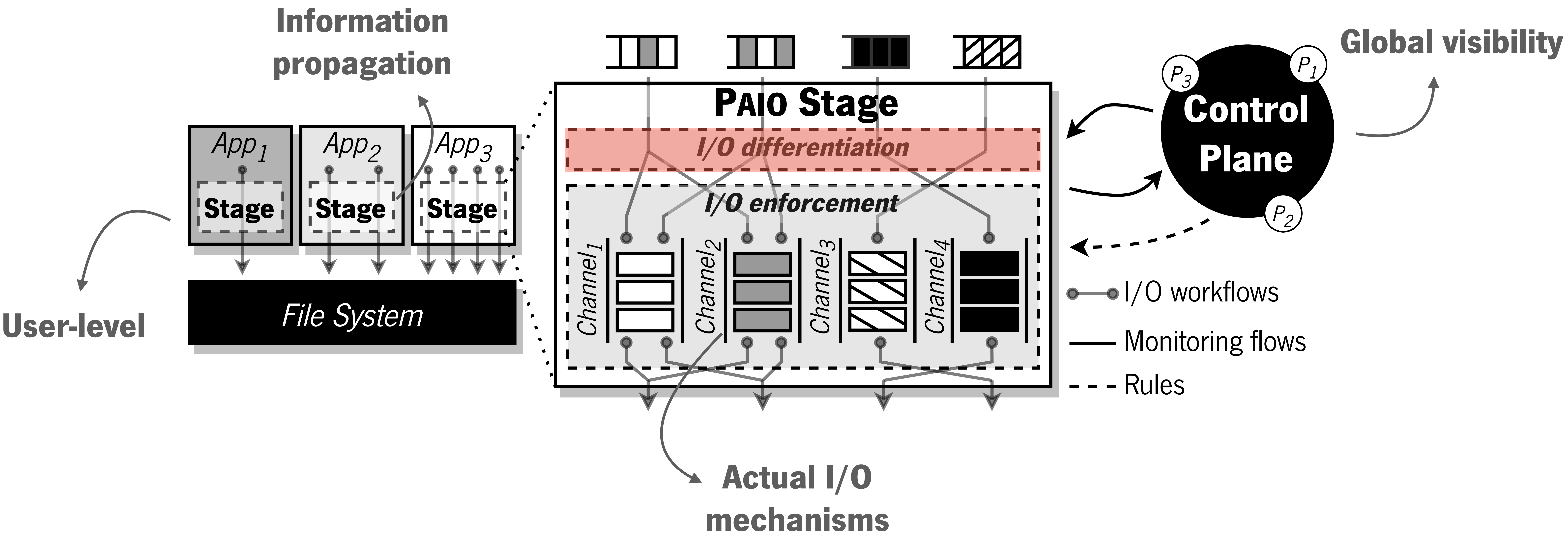
PAIO design


Information propagation
App1
App2
App3
Stage
Stage
Stage
File System
User-level
PAIO Stage
I/O differentiation
I/O enforcement
Channel1
Channel2
Channel3
Channel4
Control Plane
P1
P2
P3
Global visibility
I/O workflows
Monitoring flows
Rules
Actual I/O mechanisms

PAIO design

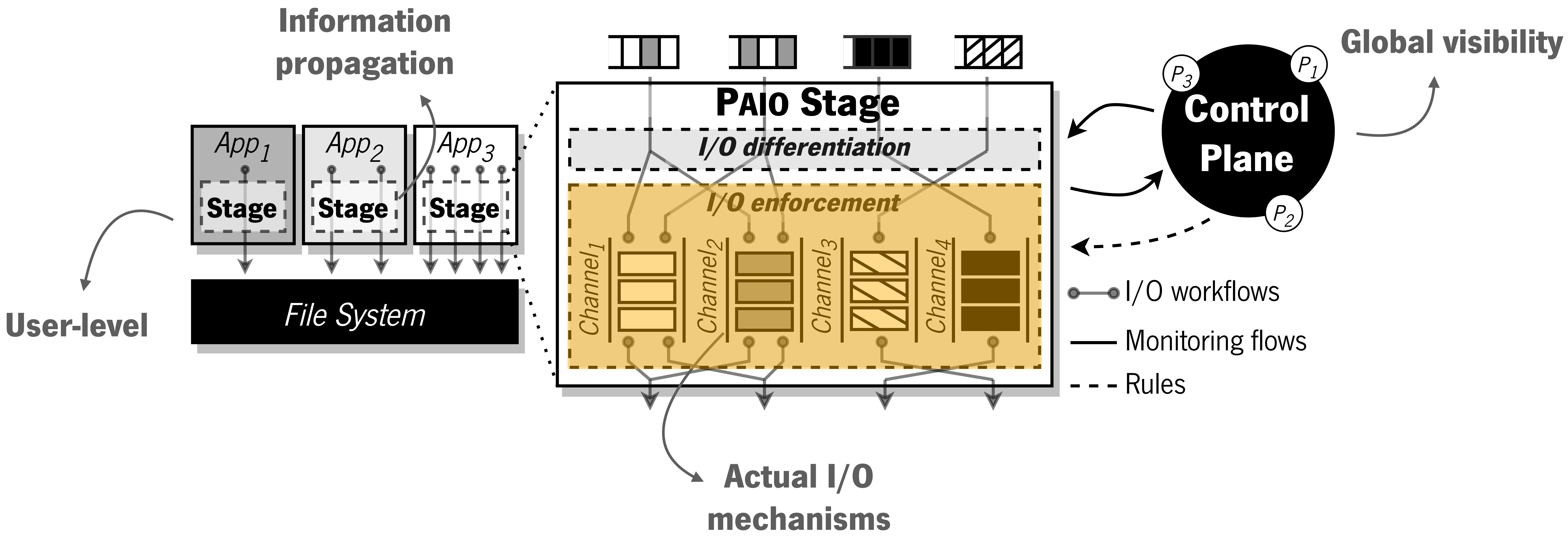

Information propagation
Global visibility
App1
App2
App3
Stage
Stage
Stage
File System
User-level
PAIO Stage
I/O differentiation
I/O enforcement
Channel1
Channel2
Channel3
Channel4
Control Plane
P1
P2
P3
I/O workflows
Monitoring flows
Rules
Actual I/O mechanisms

PAIO design



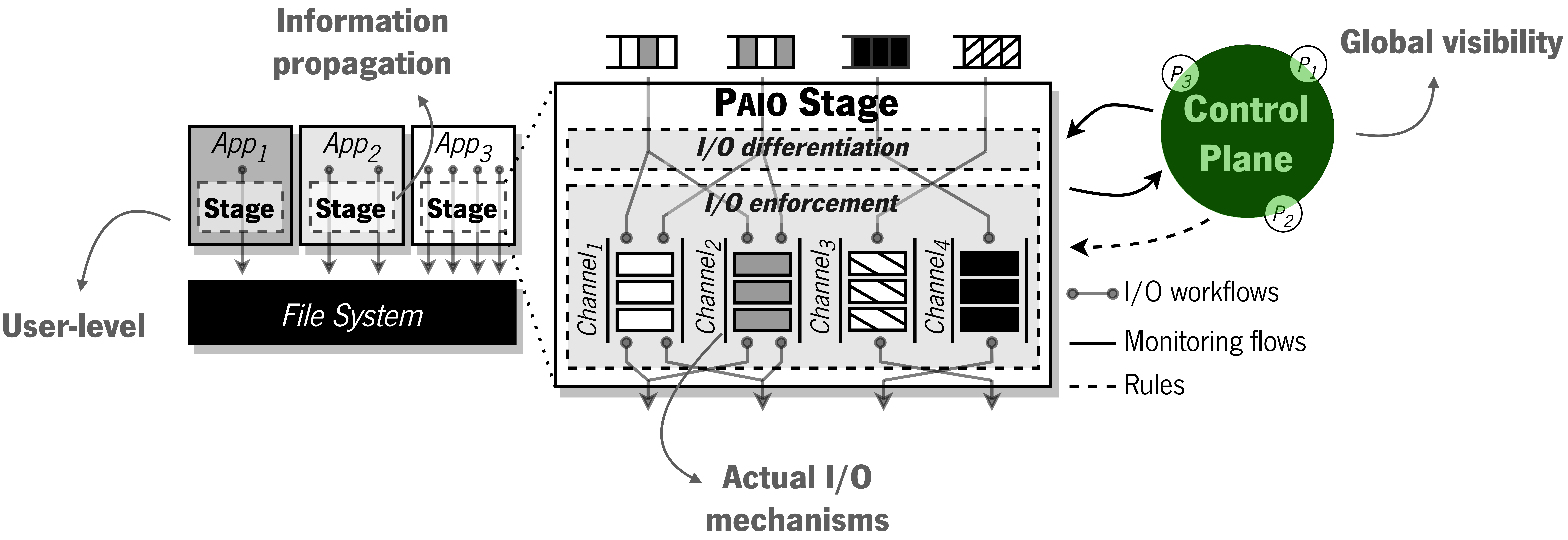
Information propagation
App1
App2
App3
Stage
Stage
Stage
File System
User-level
PAIO Stage
I/O differentiation
I/O enforcement
Channel1
Channel2
Channel3
Channel4
Control Plane
P1
P2
P3
Global visibility
I/O workflows
Monitoring flows
Rules
Actual I/O mechanisms

PAIO design



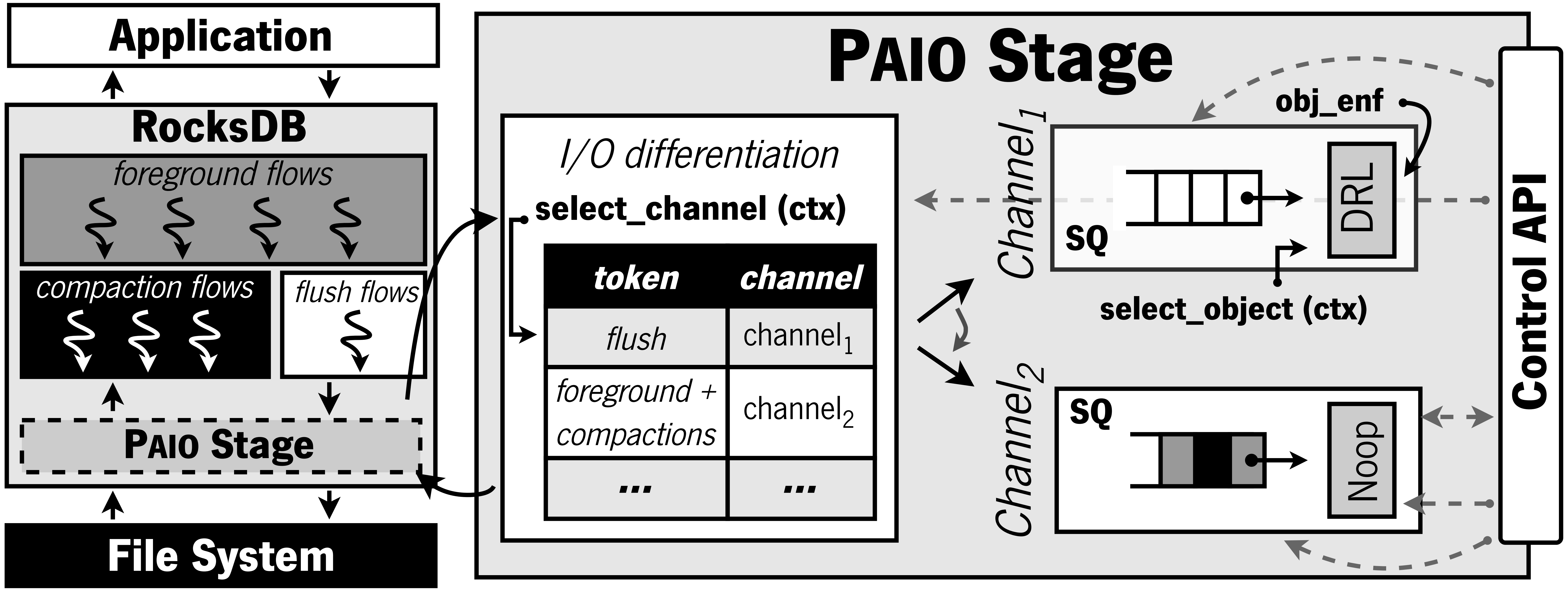
Information propagation
Global visibility
User-level
Actual I/O mechanisms
App1
App2
App3
Stage
Stage
Stage
File System
PAIO Stage
I/O differentiation
I/O enforcement
Channel1
Channel2
Channel3
Channel4
Control Plane
P1
P2
P3
I/O workflows
Monitoring flows
Rules

PAIO design


Information propagation
App1
App2
App3
Stage
Stage
Stage
File System
User-level
PAIO Stage
I/O differentiation
I/O enforcement
Channel1
Channel2
Channel3
Channel4
Control Plane
P1
P2
P3
Global visibility
I/O workflows
Monitoring flows
Rules
Actual I/O mechanisms

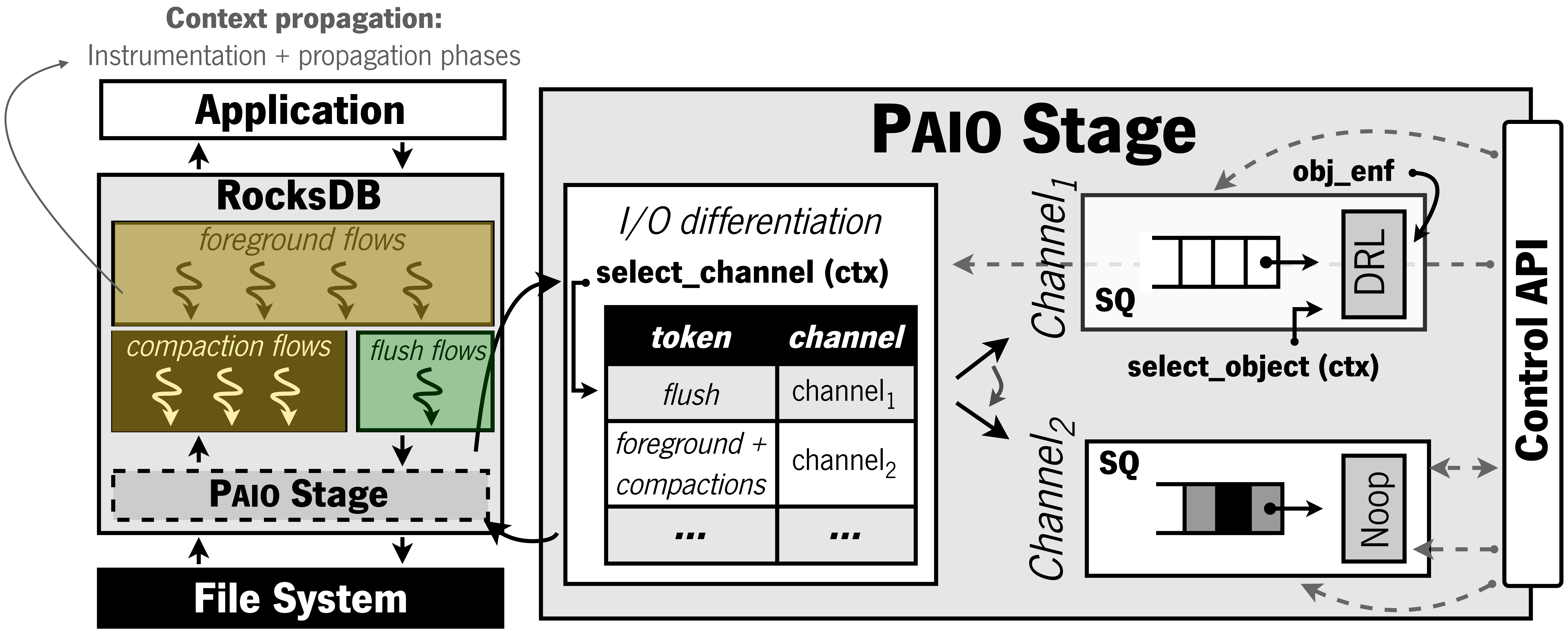
PAIO design

- Context propagation
- I/O differentiation
- I/O enforcement



Policy: limit the rate of RocksDB's flush operations to X MiB/s

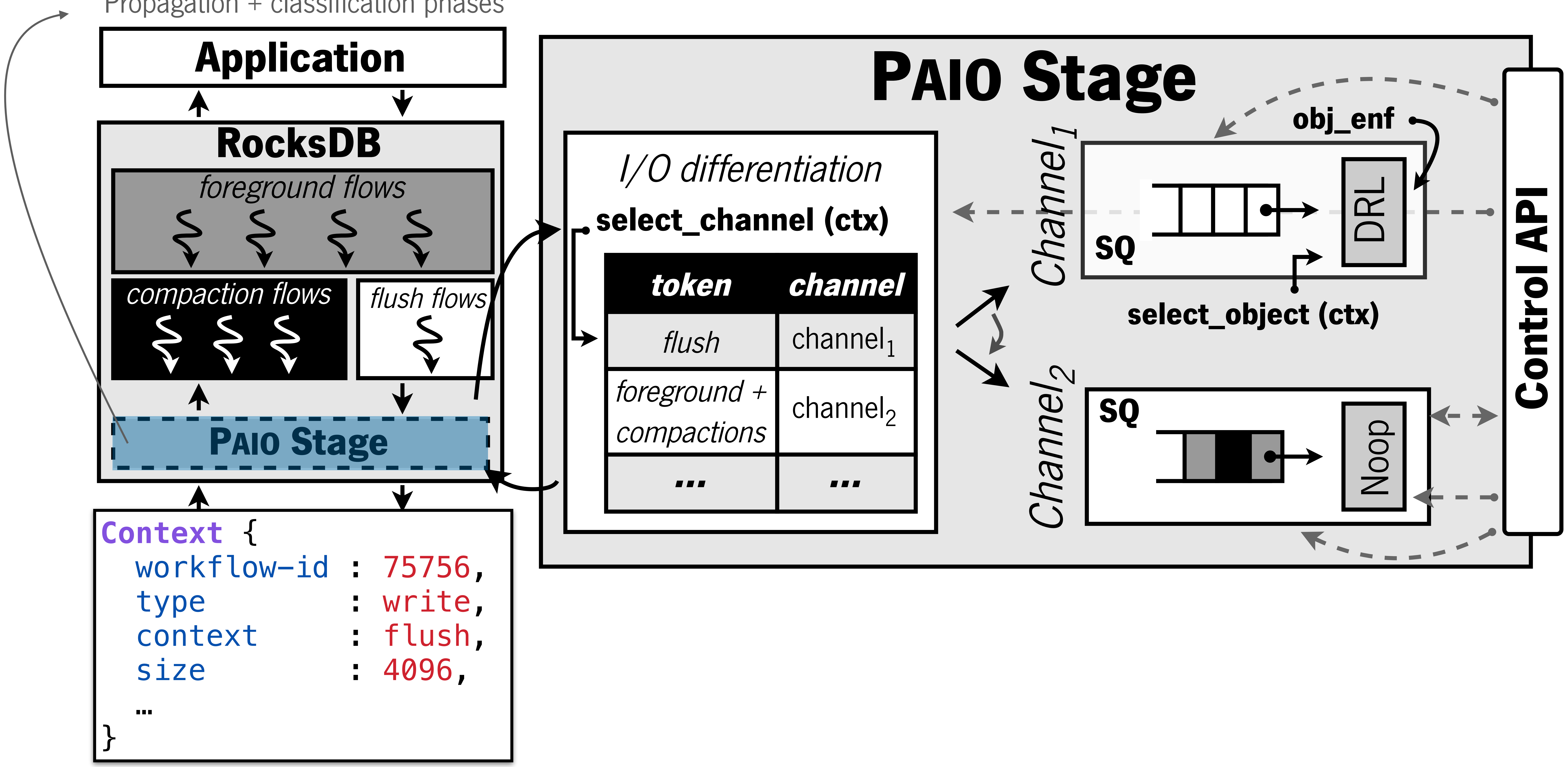
I/O differentiation



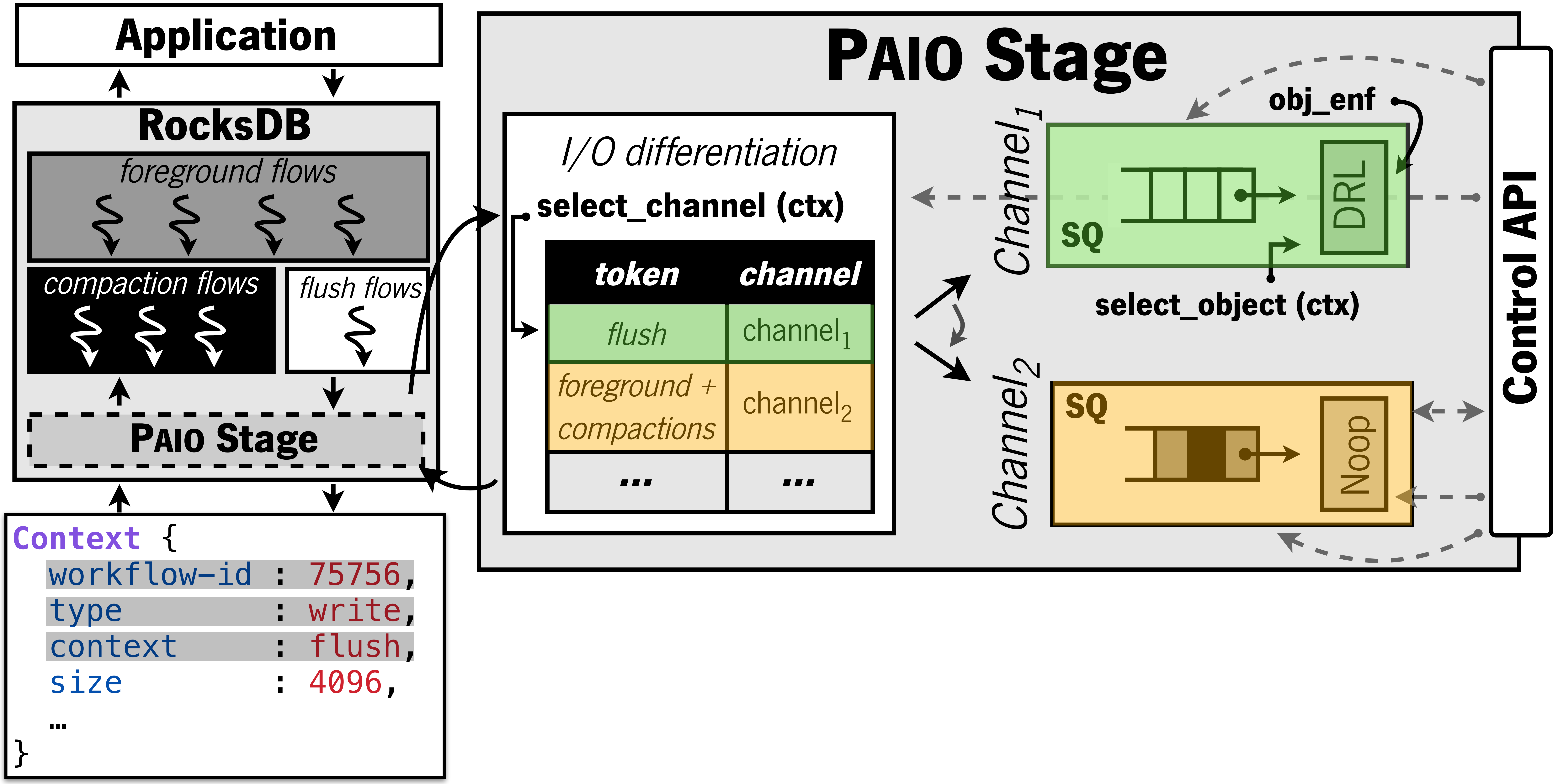
Identify the origin of POSIX operations (i.e., **foreground**, **compaction**, or **flush** operations)

I/O differentiation

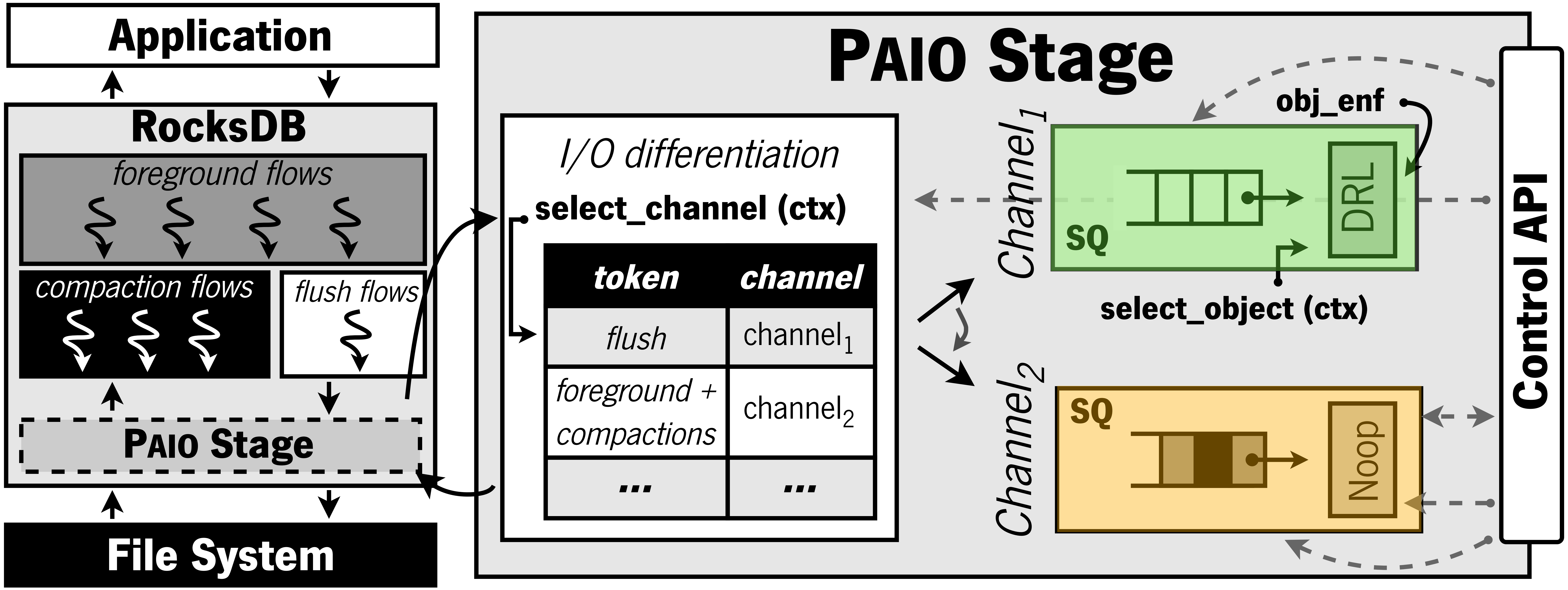
Context propagation:
Propagation + classification phases



I/O differentiation

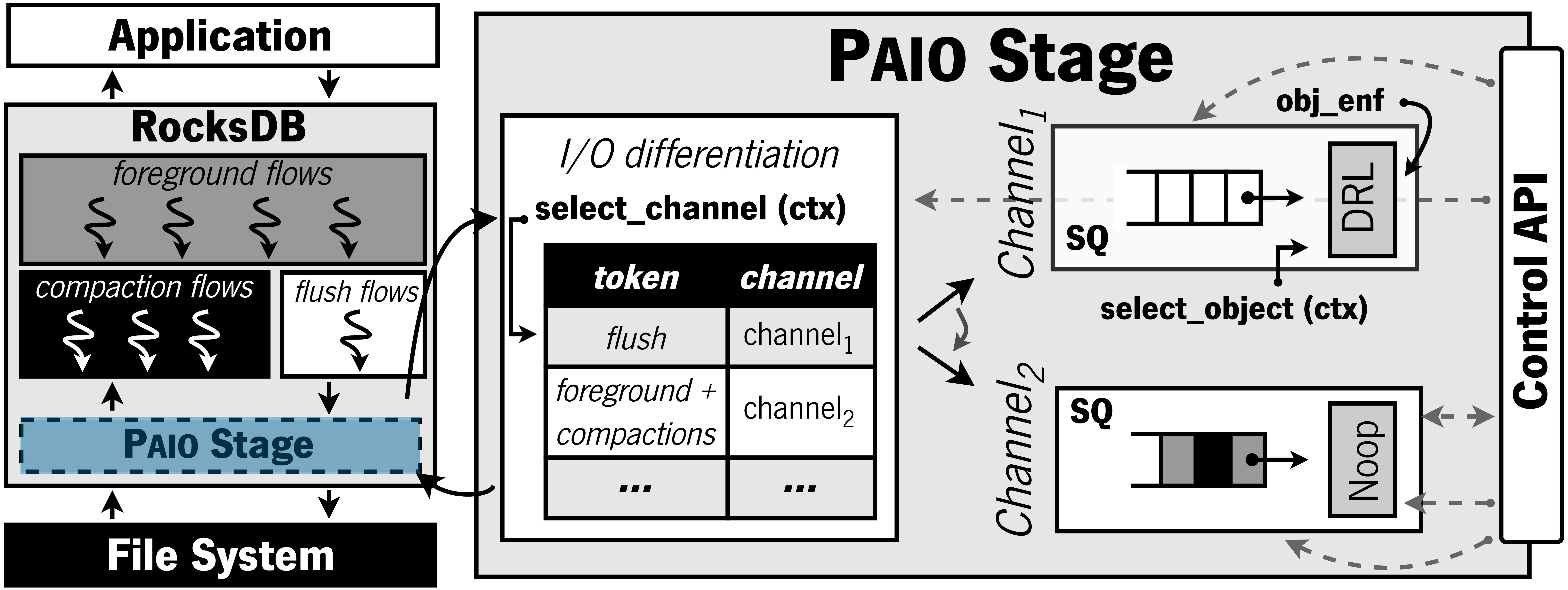


I/O enforcement



PAIO currently supports **Noop** (passthrough) and **DRL** (token-bucket) enforcement objects

I/O enforcement



Requests return to their original I/O path

Data plane stages built with PAIO

- **Per-application bandwidth control** under shared storage environments
  - Applied over multiple **TensorFlow** instances in the ABCI (AIST) supercomputer
- **Tail latency control** in Log-Structured Merge-tree key-value stores*
  - Applied over **RocksDB**, a production-ready key-value store from Meta
- **Metadata control** in Parallel File Systems*
  - Applied over **metadata-aggressive jobs** in Frontera (TACC) and ABCI supercomputers

*** Discussed in this presentation.**

Tail latency control in LSM-based key-value stores

RocksDB

- Interference between foreground and background tasks generates high latency spikes
- Latency spikes occur due to L_0-L_1 compactions and flushes being slow or on hold

SILK

- I/O scheduler
  - Allocates bandwidth for internal operations when client load is low
  - Prioritizes flushes and low level compactions
  - Preempts high level compactions with low level ones
- Requires changing several core modules made of thousands of LoC (≈335K LoC)

Tail latency control in LSM-based key-value stores

RocksDB

- Interference between foreground and background tasks generates high latency spikes
- Latency spikes occur due to L_0-L_1 compactions and flushes being slow or on hold

SILK

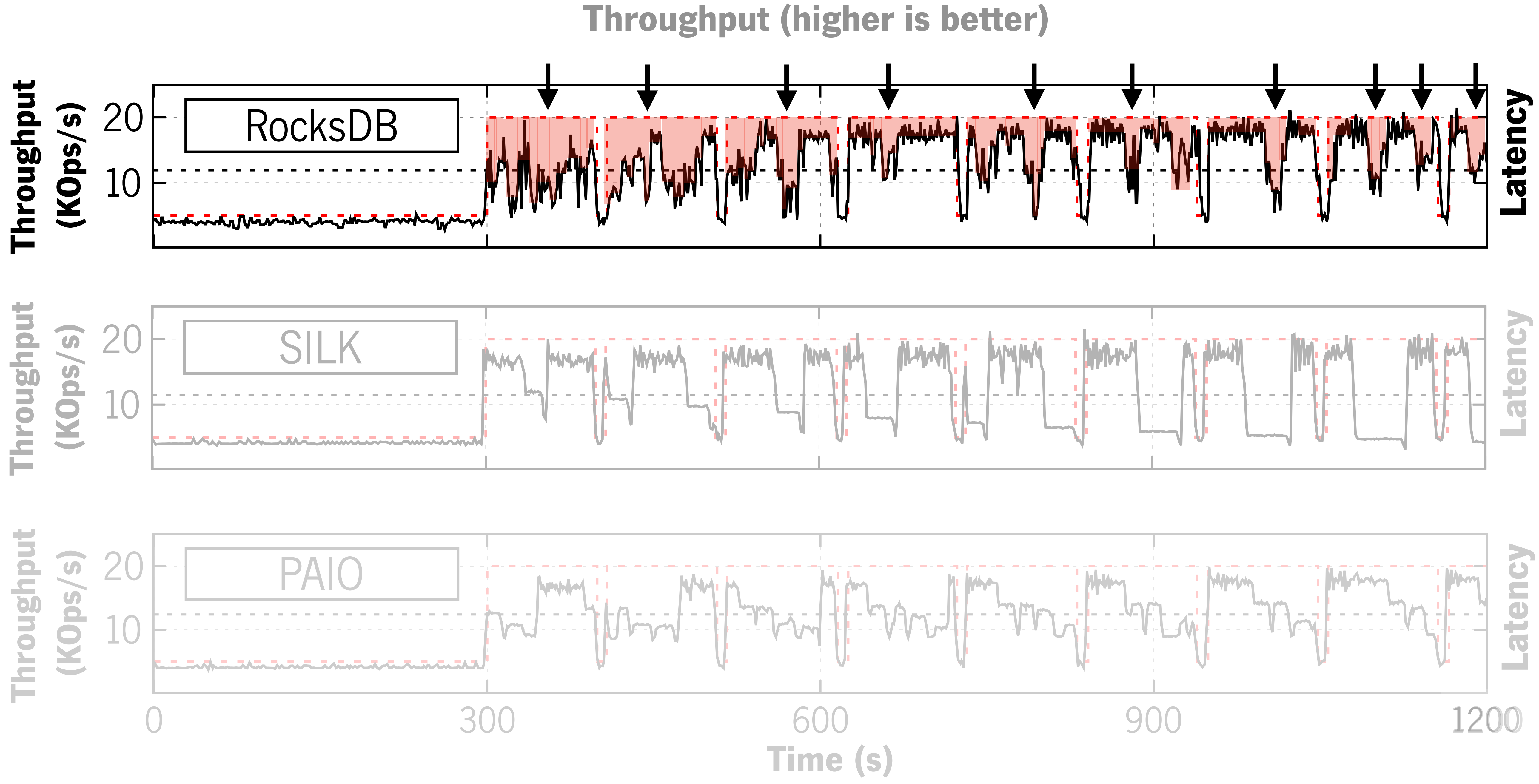
- I/O scheduler
  - **Allocates bandwidth for internal operations when client load is low**
  - **Prioritizes flushes and low level compactions**
  - ~~Preempts high level compactions with low level ones~~
- Requires changing several core modules made of thousands of LoC (≈335K LoC)

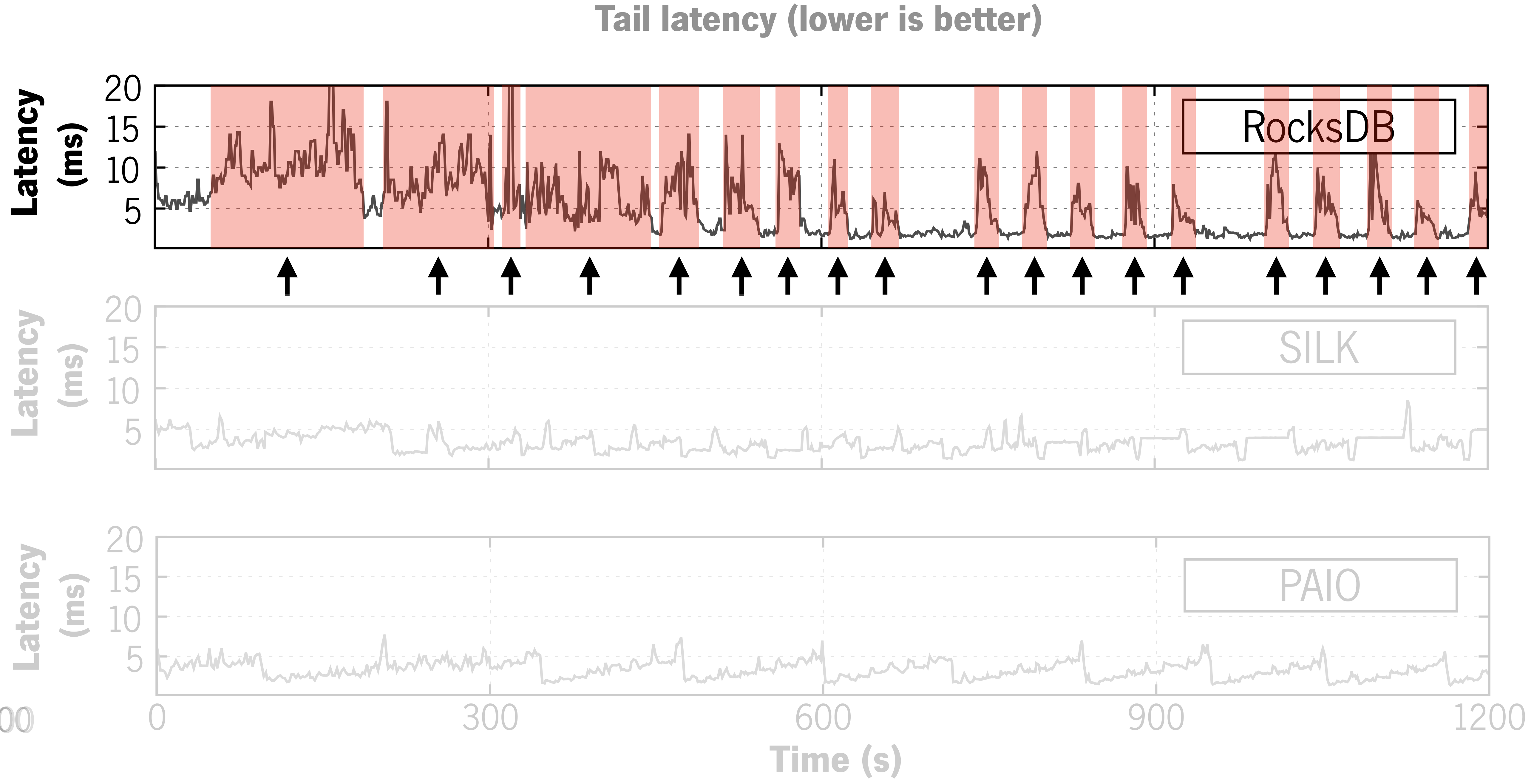
PAIO

- Stage provides the I/O mechanisms for prioritizing and rate limiting background flows
  - Integrating PAIO in RocksDB only required adding 85 LoC
- Control plane provides a SILK-based I/O scheduling algorithm

Evaluation
Mixture workload

System configuration and workload
- 8 client threads and 8 background threads
- Memory limited to 1GB and I/O BW to 200MB/s
- Bursty workload with peaks and valleys
- db_bench with YCSB A (50% read 50% write)





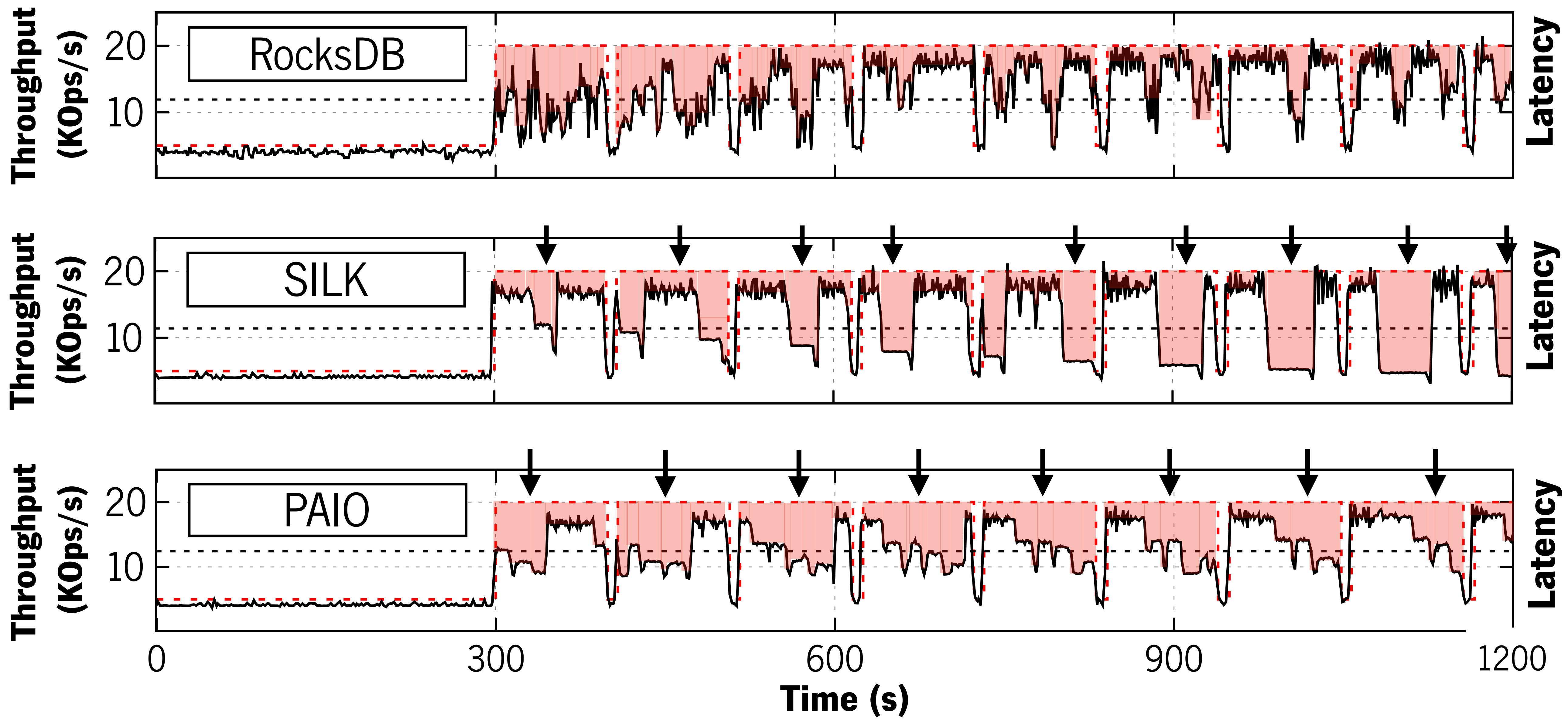
Throughput: high variability due to constant flushes and compactions

99th latency: high tail latency with peaks with an average range between 3 and 15 ms

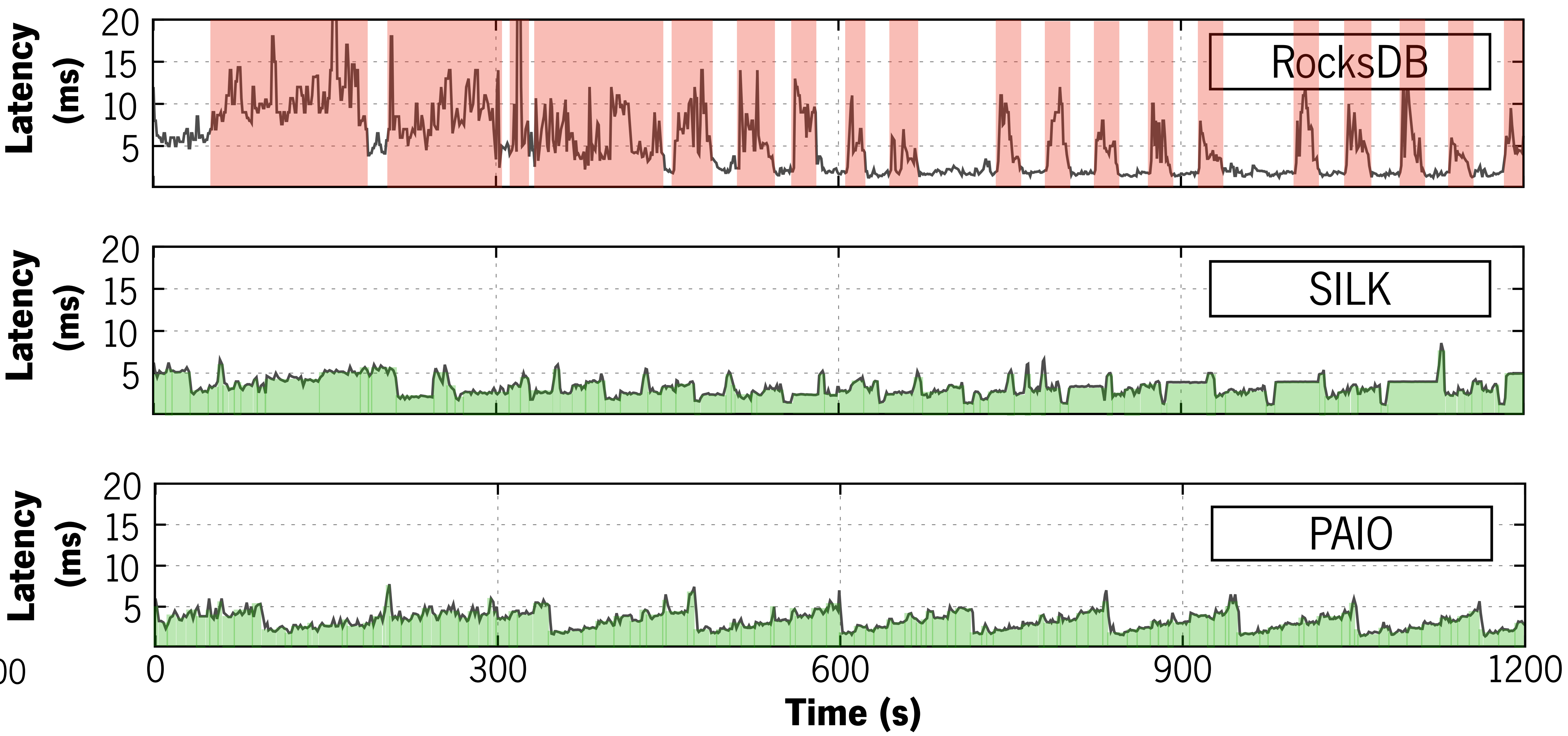
Evaluation
Mixture workload

System configuration and workload
- 8 client threads and 8 background threads
- Memory limited to 1GB and I/O BW to 200MB/s
- Bursty workload with peaks and valleys
- db_bench with YCSB A (50% read 50% write)

Throughput (higher is better)



Tail latency (lower is better)



Throughput: suffers periodic throughput drops due to accumulated backlog

99th latency: low and sustained tail latency

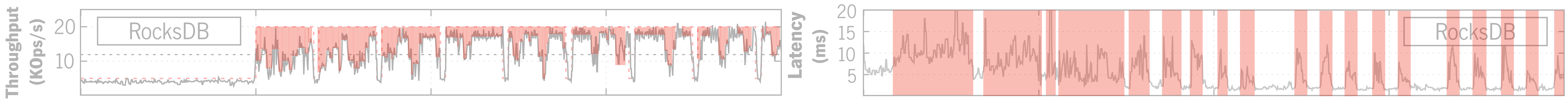
PAIO and SILK observe a 4x decrease in absolute tail latency

Evaluation
Mixture workload

System configuration and workload
- 8 client threads and 8 background threads
- Memory limited to 1GB and I/O BW to 200MB/s
- Bursty workload with peaks and valleys
- db_bench with YCSB A (50% read 50% write)

Throughput (higher is better)

Tail latency (lower is better)



By propagating application-level information to the stage, PAIO can enable similar control and performance as system-specific optimizations

Throughput: suffers periodic throughput drops due to accumulated backlog

99th latency: low and sustained tail latency

PAIO and SILK observe a 4x decrease in absolute tail latency

Metadata control in parallel file systems

- HPC workloads are no longer compute-bound and write-dominated
  - Modern workloads are **read-dominated** and with massive **bursts of metadata** operations
- Lustre-like parallel file systems (PFS) provide a **centralized metadata management** service
- Multiple jobs competing over **shared metadata resources**
  - Severe I/O contention
  - Overall performance degradation

Metadata control in parallel file systems

- **PADLL**, an application and file system agnostic storage middleware that enables QoS of metadata workflows in HPC storage

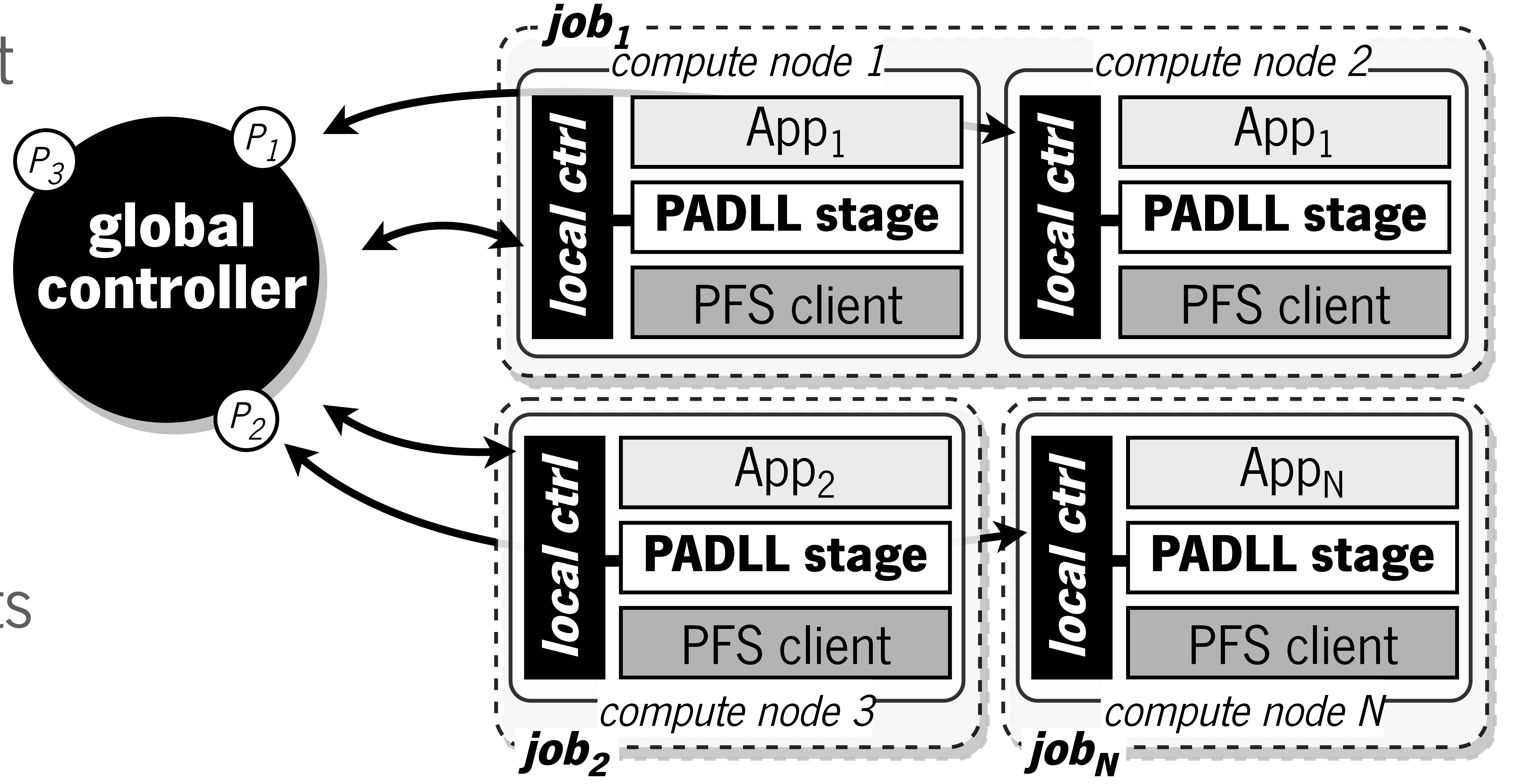
- **Proactively** and **holistically controls** the rate at which POSIX requests are submitted to the PFS

- Data plane actuates at the **compute node level**

- Control plane follows a **hierarchical** organization

- New **max-min fair share algorithm** that prevents resource over-provisioning under volatile workloads

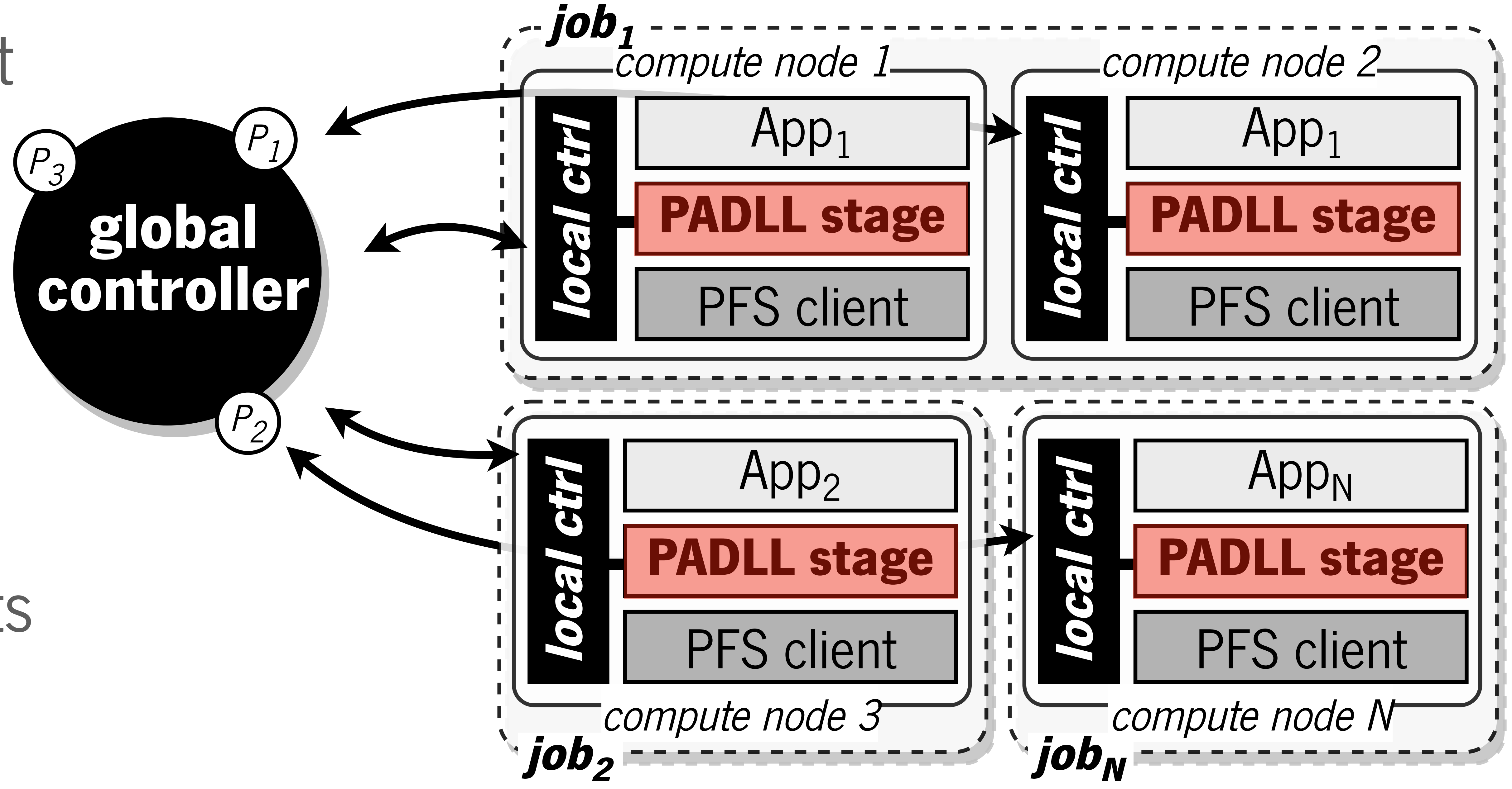
- PADLL does not require any code changes



Macedo *et al. "Protecting Metadata Servers From Harm Through Application-level I/O Control"*. IEEE Cluster @ REX-IO, 2022.
Macedo *et al. Taming Metadata-intensive HPC Jobs Through Dynamic, Application-agnostic QoS Control"*. 23rd IEEE/ACM CCGrid, 2023. In ***submission***.

Metadata control in parallel file systems

- **PADLL**, an application and file system agnostic storage middleware that enables QoS of metadata workflows in HPC storage
- **Proactively** and **holistically controls** the rate at which POSIX requests are submitted to the PFS
- Data plane actuates at the **compute node level**
- Control plane follows a **hierarchical** organization
- New **max-min fair share algorithm** that prevents resource over-provisioning under volatile workloads
- PADLL does not require any code changes



Macedo *et al. "Protecting Metadata Servers From Harm Through Application-level I/O Control"*. IEEE Cluster @ REX-IO, 2022.
Macedo *et al. Taming Metadata-intensive HPC Jobs Through Dynamic, Application-agnostic QoS Control"*. 23rd IEEE/ACM CCGrid, 2023. In ***submission***.

Metadata control in parallel file systems

- **PADLL**, an application and file system agnostic storage middleware that enables QoS of metadata workflows in HPC storage

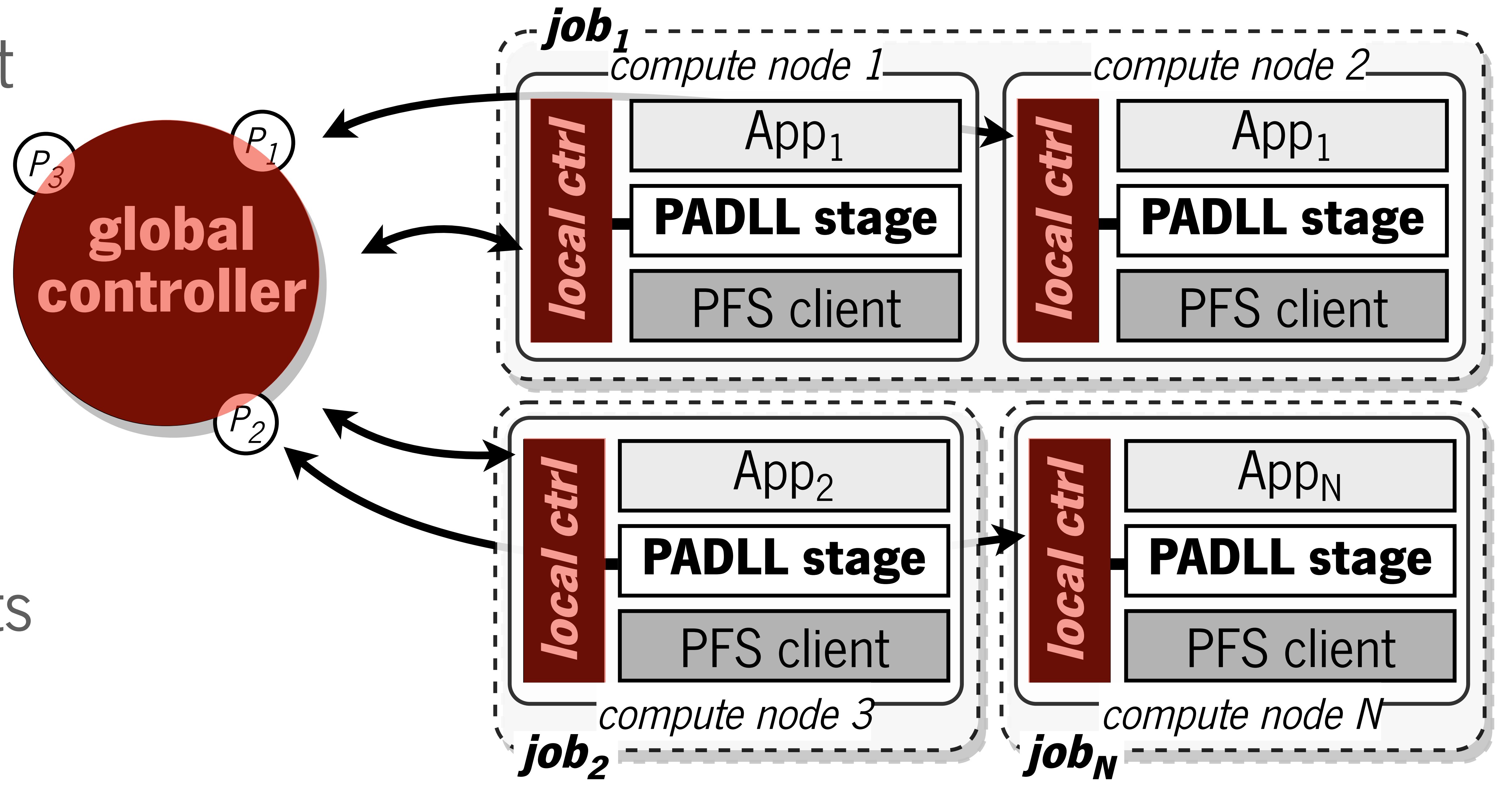
- **Proactively** and **holistically controls** the rate at which POSIX requests are submitted to the PFS

- Data plane actuates at the **compute node level**

- Control plane follows a **hierarchical** organization

- New **max-min fair share algorithm** that prevents resource over-provisioning under volatile workloads

- PADLL does not require any code changes



Macedo *et al. "Protecting Metadata Servers From Harm Through Application-level I/O Control"*. IEEE Cluster @ REX-IO, 2022.
Macedo *et al. Taming Metadata-intensive HPC Jobs Through Dynamic, Application-agnostic QoS Control"*. 23rd IEEE/ACM CCGrid, 2023. In ***submission***.

Evaluation
Metadata-aggressive jobs

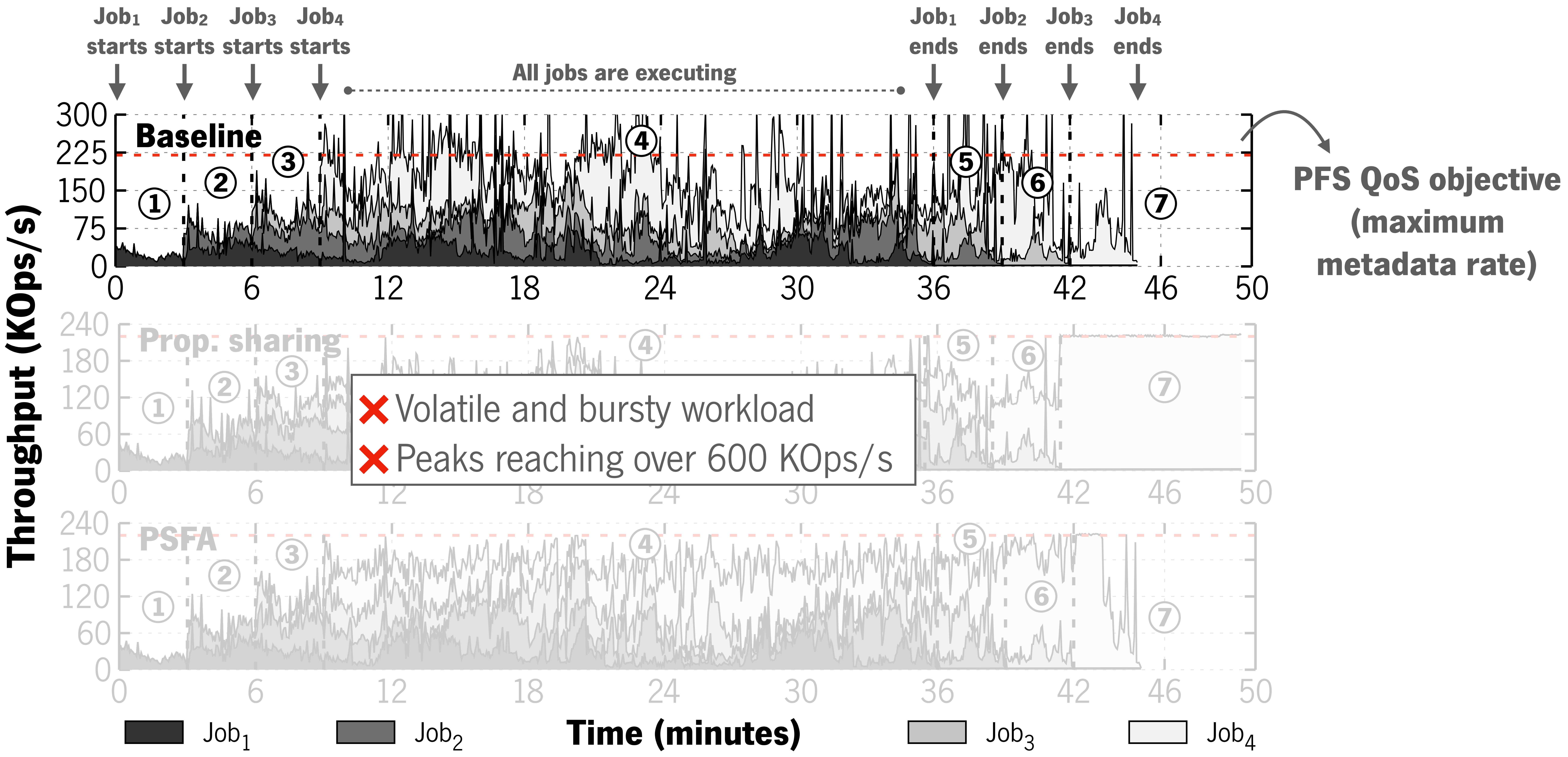
- **Objective**
  - **Limit overall metadata load in the PFS, while assigning different I/O priorities to jobs**
- **Experimental environment**
  - Multi-job QoS control in the Frontera supercomputer
  - Trace replayer with metadata traces from the ABCI production cluster
- **Setups**
  - Baseline
  - Proportional Sharing (state-of-the-art QoS algorithm)
  - Proportional Sharing Without False Resource Allocation (new QoS algorithm)

Evaluation

Metadata-aggressive jobs

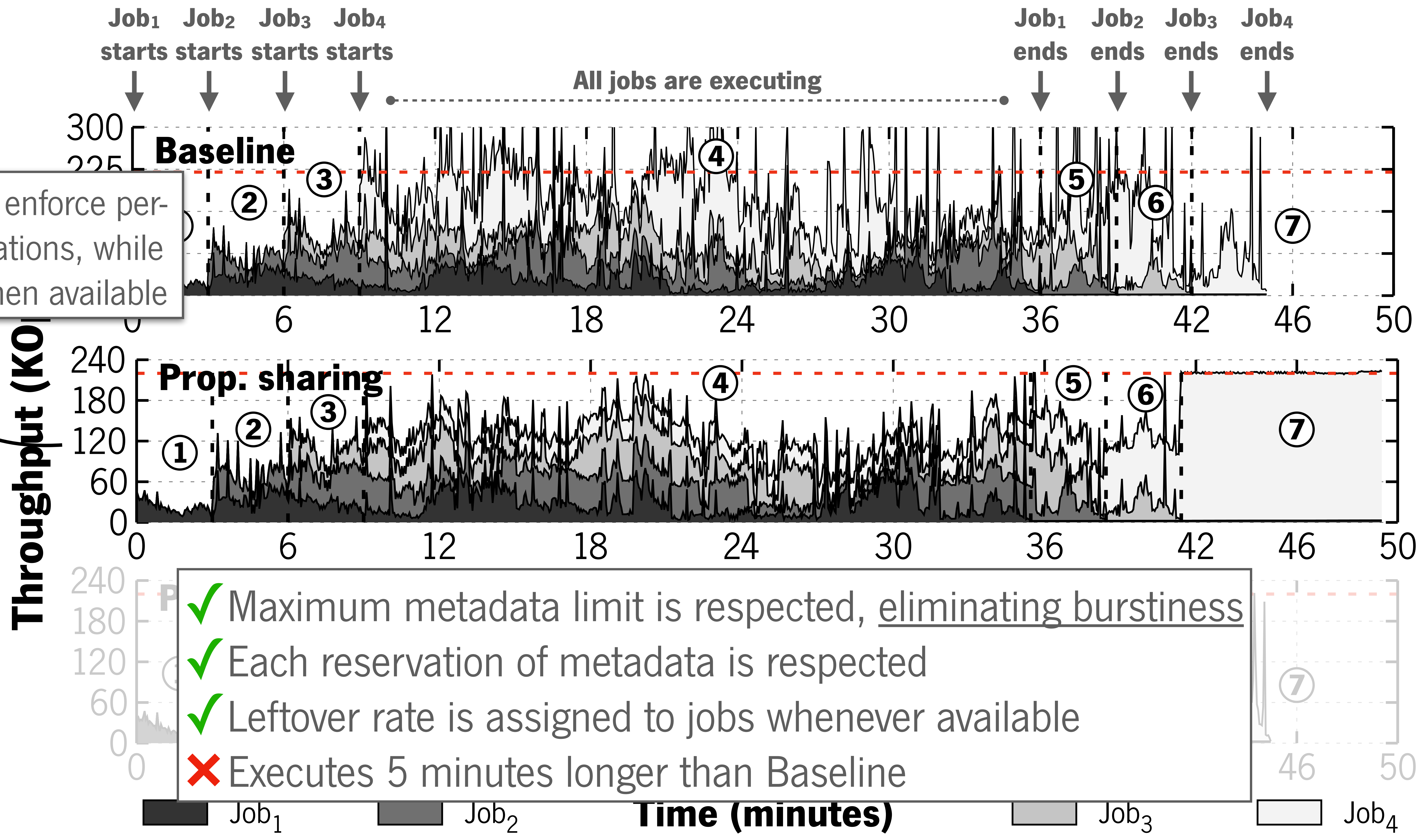
System configuration and workload

- Maximum metadata rate is set to **220 KOps/s**
- New job is added every 3 minutes
- Baseline execution time is 36 minutes (per job)
- Jobs execute with different loads {15%,20%,20%,45%}



Evaluation
Metadata-aggressive jobs

System configuration and workload
- Maximum metadata rate is set to **220 KOps/s**
- New job is added every 3 minutes
- Baseline execution time is 36 minutes (per job)
- Jobs execute with different loads {15%,20%,20%,45%}

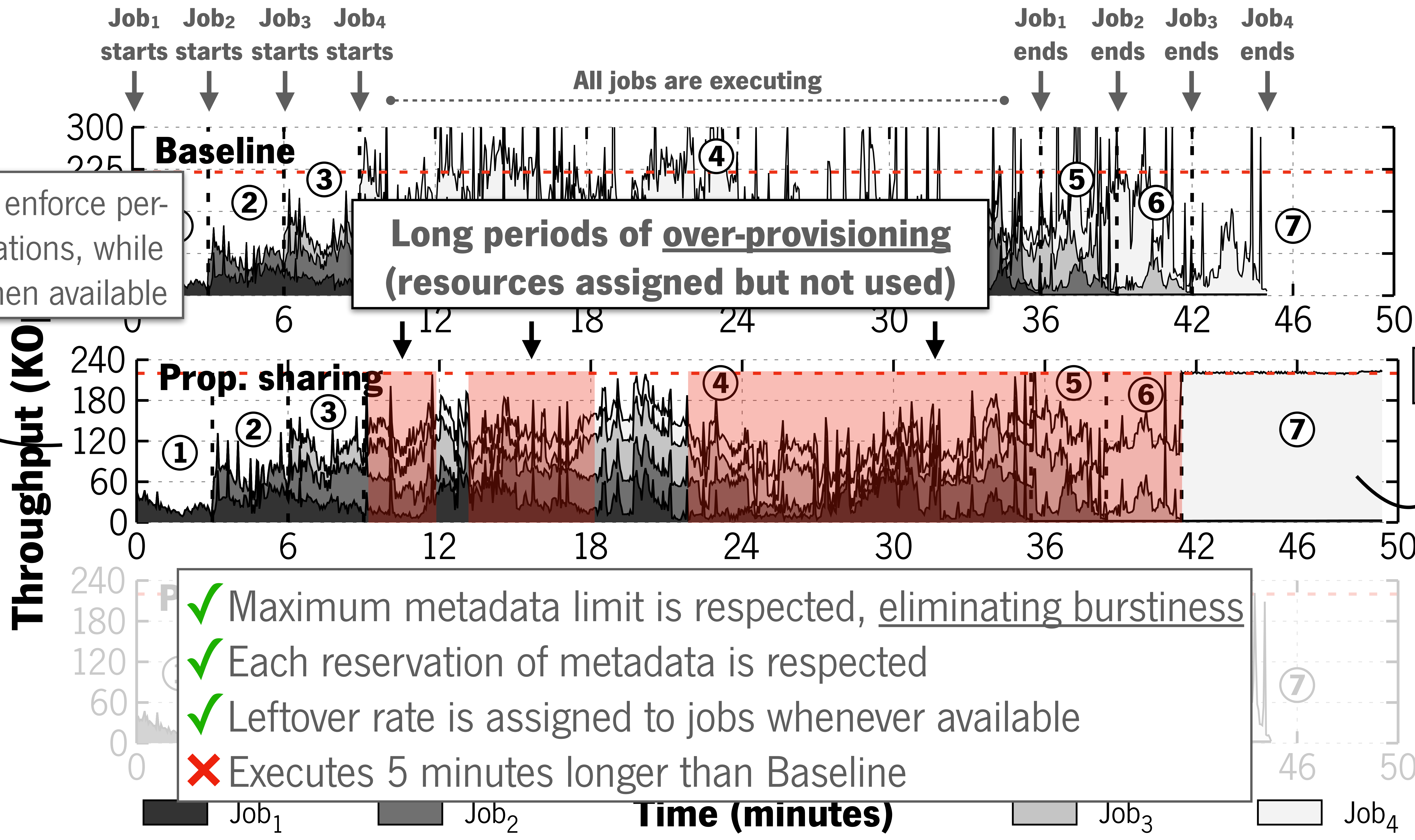


Proportional sharing: enforce per-job metadata rate reservations, while assigning leftover rate when available

- ✓ Maximum metadata limit is respected, eliminating burstiness
- ✓ Each reservation of metadata is respected
- ✓ Leftover rate is assigned to jobs whenever available
- ✗ Executes 5 minutes longer than Baseline

Evaluation
Metadata-aggressive jobs

System configuration and workload
- Maximum metadata rate is set to **220 KOps/s**
- New job is added every 3 minutes
- Baseline execution time is 36 minutes (per job)
- Jobs execute with different loads {15%,20%,20%,45%}



Proportional sharing: enforce per-job metadata rate reservations, while assigning leftover rate when available

Long periods of over-provisioning (resources assigned but not used)

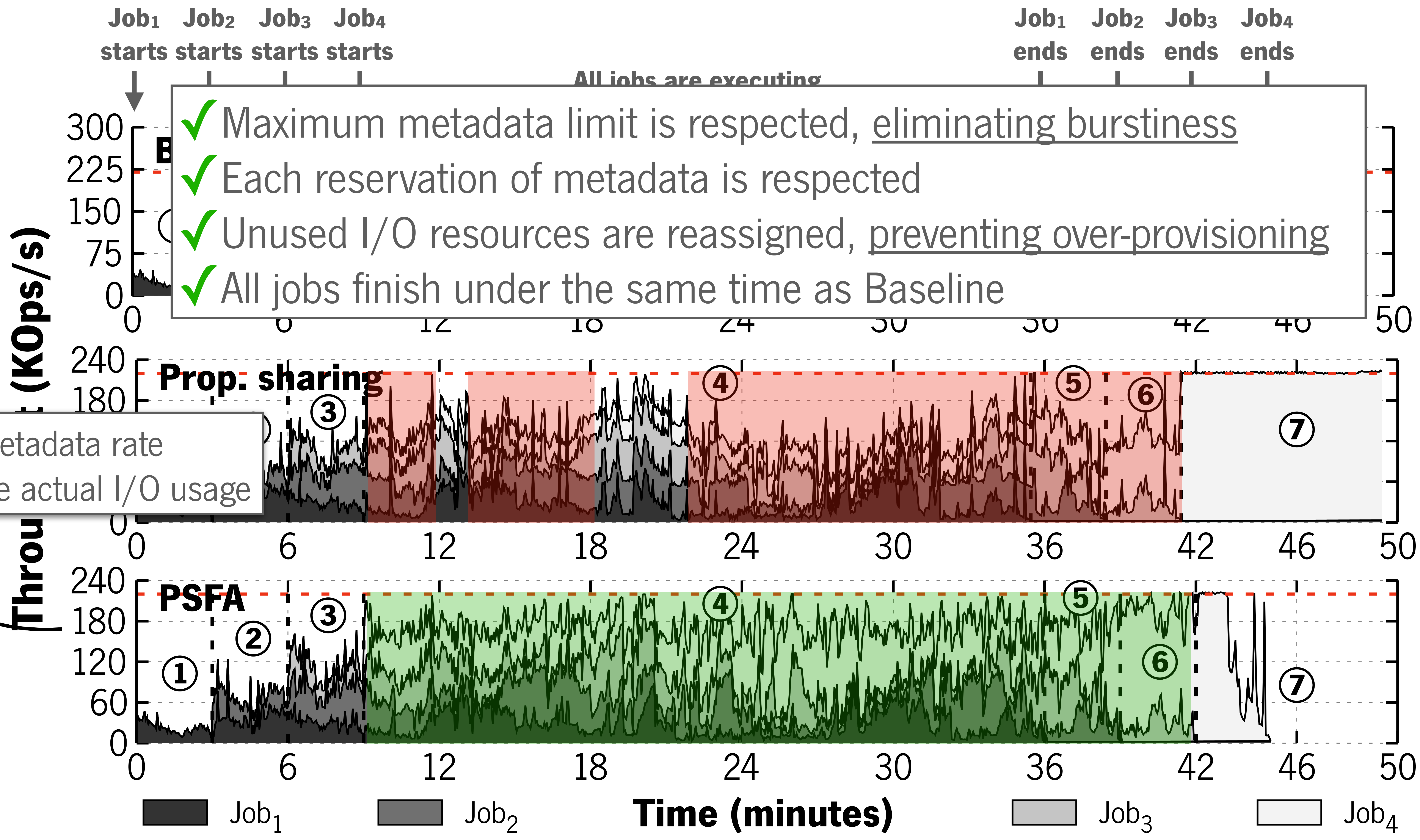
Accumulated backlog!

- ✓ Maximum metadata limit is respected, eliminating burstiness
- ✓ Each reservation of metadata is respected
- ✓ Leftover rate is assigned to jobs whenever available
- ✗ Executes 5 minutes longer than Baseline

Evaluation
Metadata-aggressive jobs

System configuration and workload

- Maximum metadata rate is set to **220 KOps/s**
- New job is added every 3 minutes
- Baseline execution time is 36 minutes (per job)
- Jobs execute with different loads {15%,20%,20%,45%}



✓ Maximum metadata limit is respected, eliminating burstiness
✓ Each reservation of metadata is respected
✓ Unused I/O resources are reassigned, preventing over-provisioning
✓ All jobs finish under the same time as Baseline

PSFA: enforce per-job metadata rate reservations based on the actual I/O usage

Summary

- **Survey** and **classification** of **SDS** systems
  - Systematization of knowledge and taxonomy of existing SDS solutions
  - Uncovers open research challenges in the field
- **PAIO**, a novel **SDS system** that enables building **complex I/O optimizations**
  - Decoupled from the targeted system
  - Perform coordinated control decisions over shared resources
  - Programmable and adaptable
- Data plane **stages** built with **PAIO**
  - Reimplement complex I/O optimizations that achieve similar performance as system-specific ones
  - New optimizations that address unsolved challenges present in modern I/O infrastructures
  - Currently working with leading HPC centers in the integration of PAIO and PADLL in production

Publications

Core publications

- **R. Macedo**, Y. Tanimura, J. Haga, V. Chidambaram, J. Pereira, J. Paulo. **“PAIO: General, Portable I/O Optimizations With Minor Application Modifications”**. *20th USENIX Conference on File and Storage Technologies*, 2022.
- **R. Macedo**, J. Paulo, J. Pereira, A. Bessani. **“A Survey and Classification of Software-Defined Storage Systems”**. *ACM Computing Surveys 53, 3 (48)*, 2020.
- **R. Macedo**, A. Faria, J. Paulo, J. Pereira. **“A Case for Dynamically Programmable Storage Background Tasks”**. *38th International Symposium on Reliable Distributed Systems Workshops*, 2019.
- **R. Macedo**, C. Correia, M. Dantas, C. Brito, W. Xu, Y. Tanimura, J. Haga, J. Paulo. **“The Case for Storage Optimization Decoupling in Deep Learning Frameworks”**. *IEEE Cluster @ REX-IO Workshop*, 2021.
- **R. Macedo**, M. Miranda, Y. Tanimura, J. Haga, A. Ruhela, S. Harrell, R. Evans, J. Paulo. **“Protecting Metadata Servers From Harm Through Application-level I/O Control”**. *IEEE Cluster @ REX-IO Workshop*, 2022.
- **R. Macedo**, M. Miranda, Y. Tanimura, J. Haga, A. Ruhela, S. Harrell, R. Evans, J. Pereira, J. Paulo. **“Taming Metadata-intensive HPC Jobs Through Dynamic, Application-agnostic QoS Control”**. *23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2023. In ***submission***.

Complementary publications

- M. Dantas, D. Leitão, P. Cui, **R. Macedo**, X. Liu, W. Xu, J. Paulo. **“Accelerating Deep Learning Training Through Transparent Storage Tiering”**. *22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2022.
- A. Faria, **R. Macedo**, J. Pereira, J. Paulo. **“BDUS: Implementing Block Devices in User Space”**. *14th ACM International System and Storage Conference*, 2021. ***Best paper runner-up.***
- M. Dantas, D. Leitão, C. Correia, **R. Macedo**, W. Xu, J. Paulo. **“Monarch: Hierarchical Storage Management for Deep Learning Frameworks”**. *IEEE Cluster @ REX-IO Workshop*, 2021.
- A. Faria, **R. Macedo**, J. Paulo. **“Pods-as-Volumes: Effortlessly Integrating Storage Systems and Middleware into Kubernetes”**. *ACM/IFIP Middleware @ WoC*, 2021.
- T. Esteves, **R. Macedo**, A. Faria, B. Portela, J. Paulo, J. Pereira, D. Harnik. **“TrustFS: An SGX-enabled Stackable File System Framework”**. *38th International Symposium on Reliable Distributed Systems Workshops*, 2019.