2nd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO'22)

Protecting Metadata Servers From Harm Through Application-level I/O Control

*Ricardo Macedo, **Mariana Miranda**, João Paulo*
INESC TEC & University of Minho

Amit Ruhela, Stephen Lien Harrell
TACC & UTAustin

Richard Todd Evans
Intel

Yusuke Tanimura, Jason Haga
AIST

September 6, 2022

Universidade do Minho

Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

High-Performance Computing

- Modern supercomputers offer a large magnitude of **computing power.**
  - Enables large-scale parallel applications to run at massive scale.

- ~~HPC workloads are **compute-bound** and **write-dominated**~~.

Modern applications
are
data-intensive and **read-dominated**

High-Performance Computing

Modern applications
are
data-intensive and **read-dominated**

⚠️ High load and burstiness of metadata operations!

Metadata Study

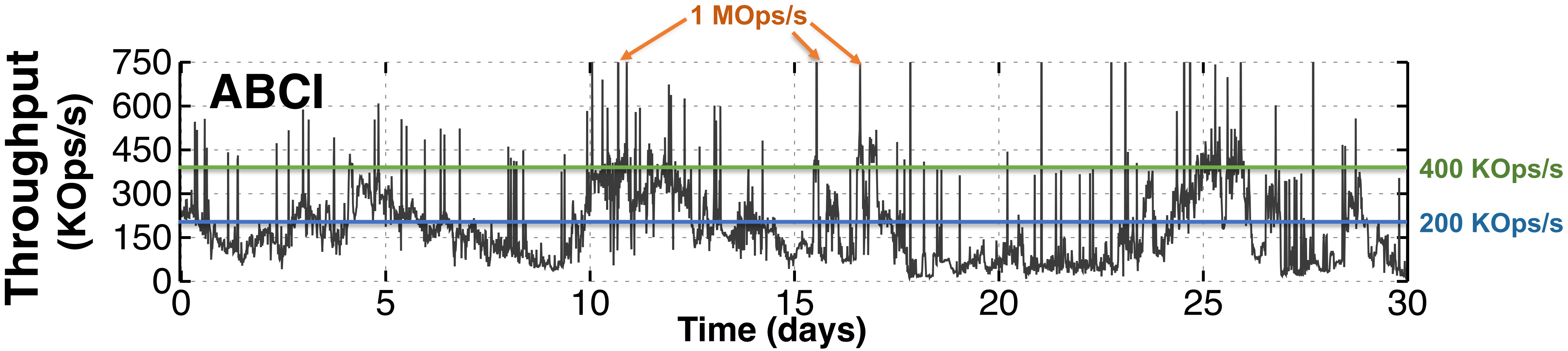
- **Analysis** of the **logs** of a **Lustre file system** from the *AI Bridging Cloud Infrastructure* (ABCI) at AIST.

- We monitored the performance of the **most frequent metadata operations** at **MDSs**/MDTs.
  - Namely, open, close, getattr, setattr, rename, mkdir, mknod, rmdir, statfs, sync and unlink.

- Collected over a **30-days** observation period.

Metadata Study

Overall metadata load

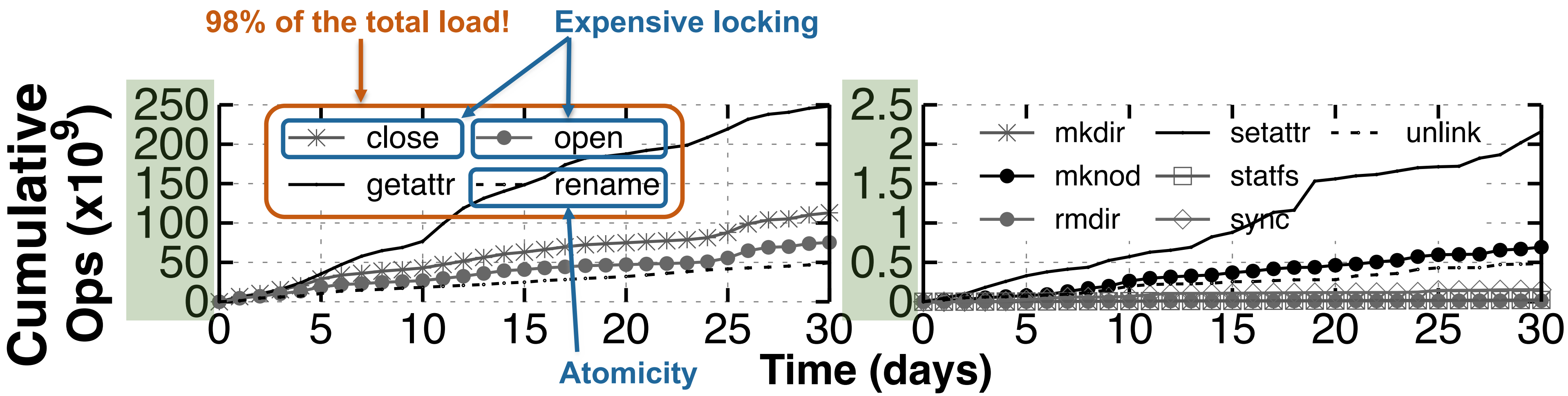


Observation #1:

Modern I/O workloads are generating **massive amounts of metadata operations**, with **high throughput rates**, and **bursts** that reach 1 MOps/s.

Metadata Study

Type and frequency of metadata operations



Operations have **different loads** and **costs**!

Observation #2:

Operations should be controlled with **fine-granularity**, ensuring that operations with different costs have **different QoS levels**.

Can HPC storage systems sustain these workloads?

The Metadata Challenge

Parallel File Systems

- Lustre-like parallel file systems provide a **centralized metadata management service.**
- Multiple **concurrent jobs** competing over shared I/O resources.

⚠️ Severe **I/O contention**

⚠️ Overall **performance degradation**

Existing solutions are suboptimal...

The Metadata Challenge

Existing Solutions

- **Manual Intervention**

  E.g., System administrators stop jobs with aggressive I/O behavior.

  ⚠️ Slow and reactive approach!

- **Intrusive to I/O layers**

  Solutions tightly coupled and intrusive to the system implementation (*e.g.*, GIFT, TBF)

  ⚠️ Low portability and maintainability!

- **Partial visibility and I/O control**

  Enabling QoS control from the application-side.

  ⚠️ Isolated and uncoordinated QoS!

Discussion

- Metadata operations are **bursty** and have **high throughput rates**.
- Some operations are **more predominant than others**.
- Different operations entail **different costs**.
- **Existing solutions** have **limitations**.
- The solution for this problem must:
    - Prevent I/O burstiness
    - Ensure fine-grained QoS control
    - Be proactive
    - Be application and PFS-agnostic
    - Have global visibility

PADLL

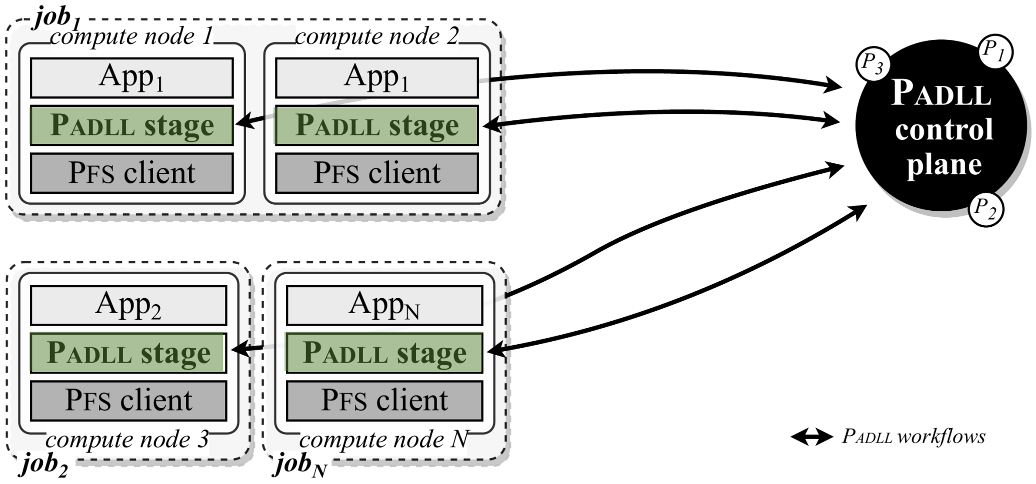
- Storage middleware that enables system administrators to proactively **ensure QoS** over **metadata workflows**.

- Adopts ideas from the **Software-Defined Storage** (SDS) paradigm.
  - **Decoupled design** that separates the I/O logic into two planes of functionality:
    - **Data plane:** application and file system agnostic middleware that applies storage policies over I/O requests (*e.g.*, rate limiting).
    - **Control plane:** holistically orchestrates and defines storage policies (*e.g.*, static rate, I/O fairness).

PADLL

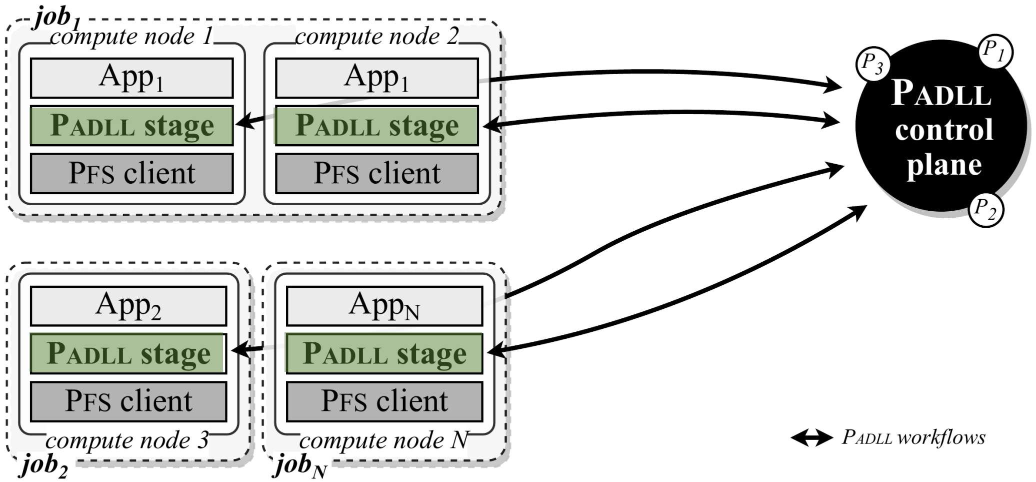
Data Plane

- **Multi-stage component** that actuates at the compute node level.
- Sits between the application and the file system client.
- Transparently **intercepts POSIX requests** before being submitted to the file system.



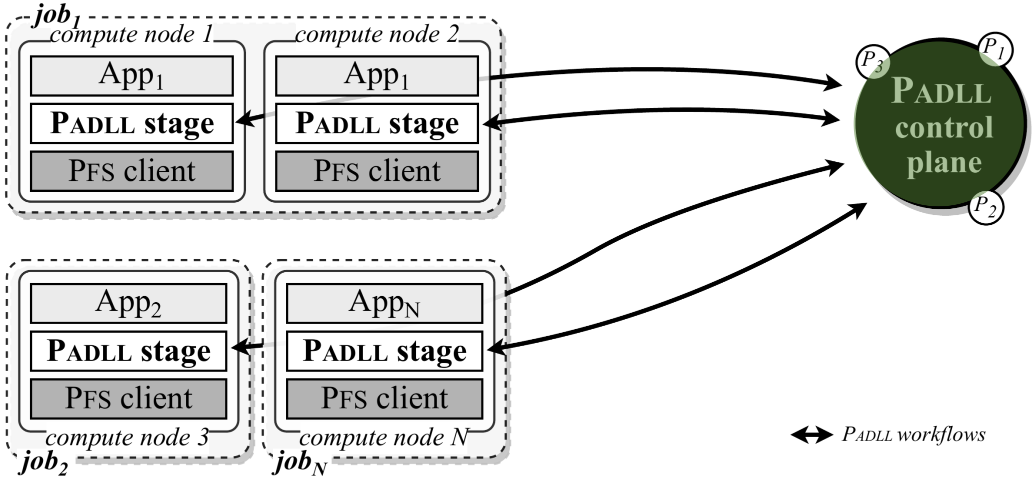
PADLL
Data Plane

- Handling requests requires two steps:
  1. **Request differentiation:** Filters which requests should be rate limited or be directly submitted to the PFS.
  2. **Rate limiting:** Stages are organized in multiple queues, each with a token-bucket that determines the rate of its requests.



PADLL
Control Plane

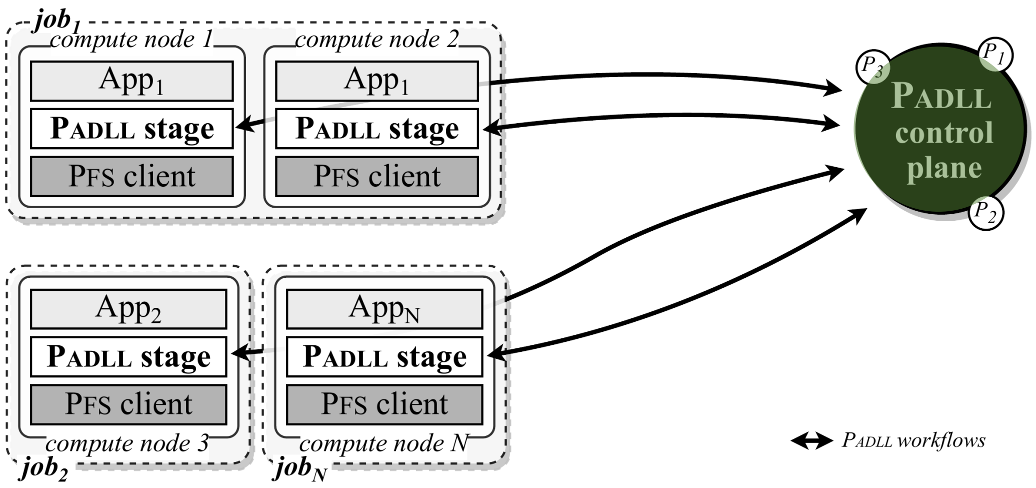
- **Logically centralized** component with **system-wide visibility** that orchestrates how the I/O workflows of all running jobs should be handled.
- **Enables system administrators** to define how the system should act.
  - **Static policies** – *e.g.*, limit open operations at X Ops/s.
  - **Dynamic policies** – *e.g.*, dynamically reserves shares of metadata operations.



PADLL
Control Plane

The control plane continuously:

1. **Collects I/O metrics** from the data plane stages (*e.g.*, workflows' rate).
2. **Computes** if all policies are being met.
3. **Enforces** new rates to respond to workload or system variations.



Implementation

- Implemented in C++
  - Data plane:
    - 16K lines of code.
    - Uses LD_PRELOAD to transparently intercept I/O requests.
    - Built using PAIO[1], a framework for building custom-made user-level storage data planes.
  - Control plane:
    - 6K lines of code.
    - Communicates with the data plane stages through RPC calls, using the gRPC framework.

[1] R. Macedo, Y. Tanimura, J. Haga, V. Chidambaram, J. Pereira, and J. Paulo, “PAIO: General, Portable I/O Optimizations With Minor Application Modifcations,” in 20th USENIX Conference on File and Storage Technologies. USENIX Association, 2022, pp. 413–428.

Evaluation

- Can PADLL enforce policies at different granularities?
- Can PADLL control I/O burstiness?
- Can PADLL enforce control algorithms over multiple concurrent jobs?
- What is the overhead of using PADLL?

Evaluation

- **Experimental testbed**
  - Compute nodes of the Frontera supercomputer.
  - Two 28-core Intel Xeon processors.
  - 192 GiB of RAM, and a single 240 GiB SSD.
  - CentOS 7.9 with the Linux kernel v3.10 and the XFS file system.
  - Lustre file system as production PFS.

- **Benchmarks and workloads**
  - Data workloads: *IOR*
  - Metadata workloads: Real traces collected from the ABCI.
    - Produced a *trace replayer* that replicates the original traces at a smaller scale.

Evaluation

- **Experiments**
  - **Per-operation type rate limiting**
    Specific operation (*e.g.*, open, close).
  - **Per-operation class rate limiting**
    E.g., metadata, data.
  - **Per-job rate limiting and QoS control**
    Multi-job setup with global orchestration.

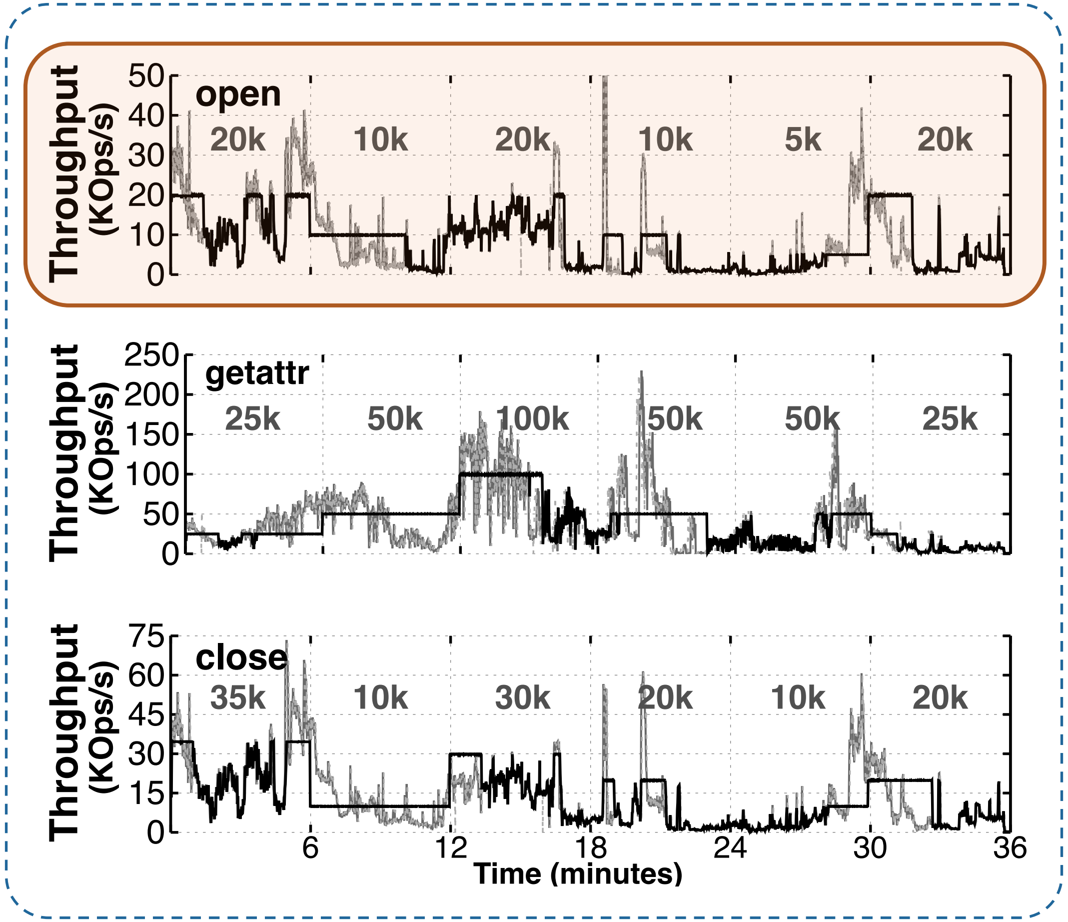
Evaluation

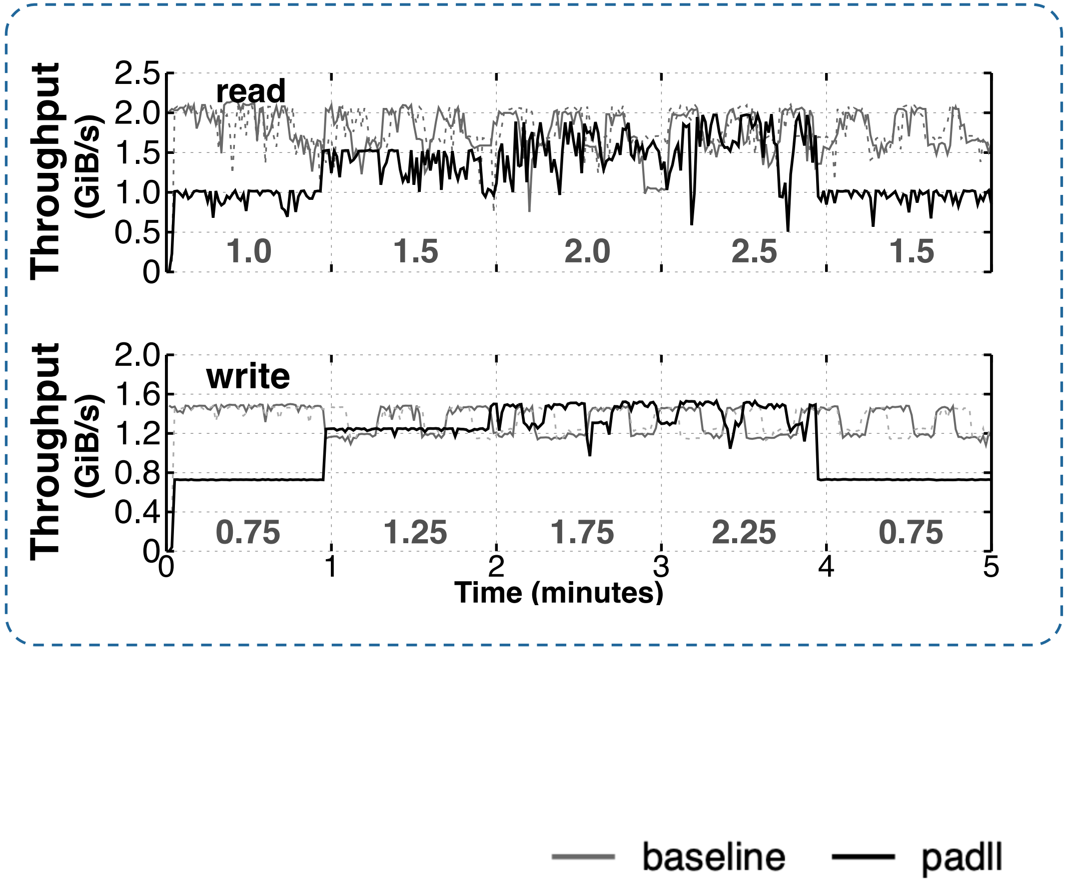
Per-operation type and class rate limiting

- Setups:
  - **Baseline**
    Benchmark (*IOR* or trace replayer) without using PADLL.
  - **PADLL**
    Operations intercepted by PADLL and throttled at a given rate.

Evaluation

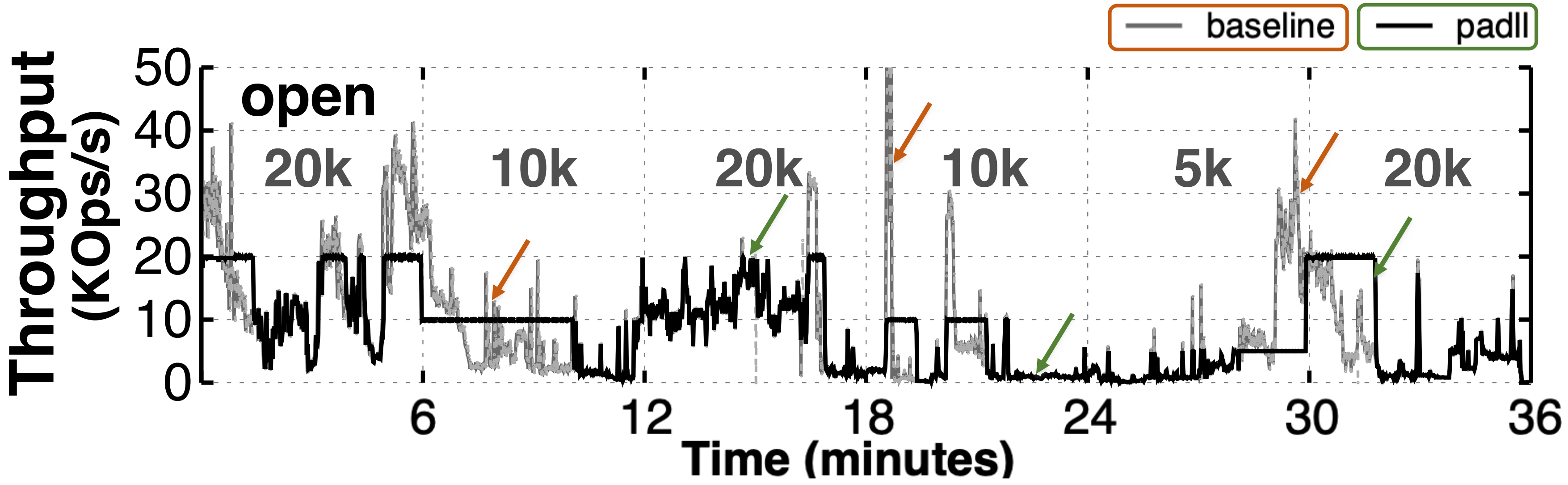
Per-operation type rate limiting





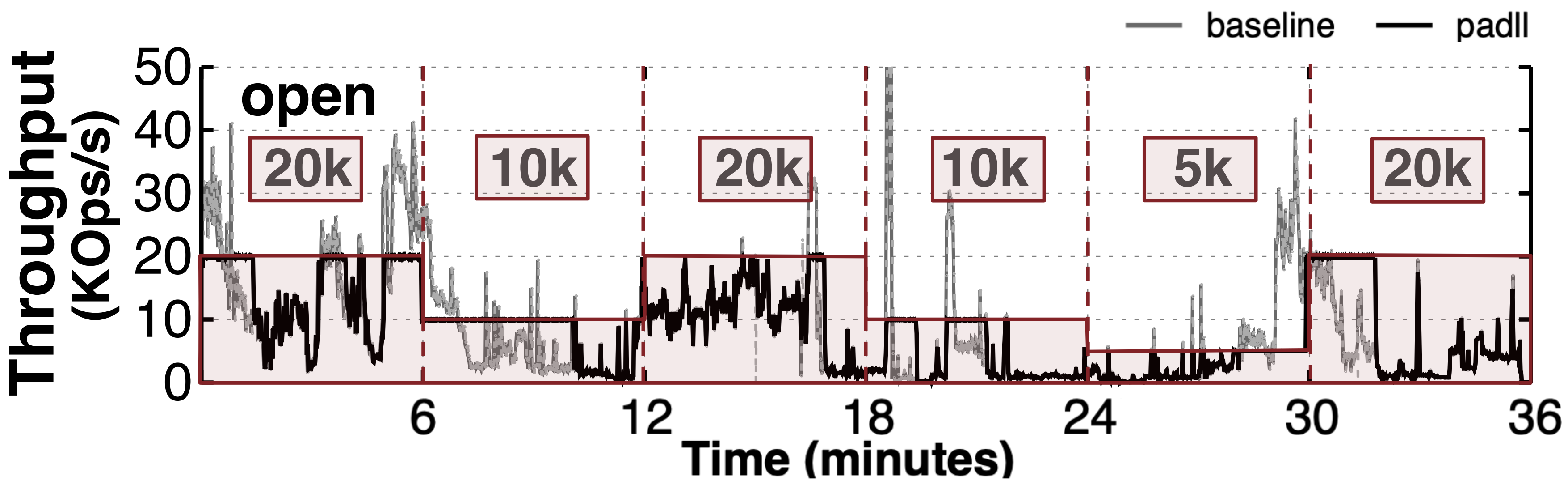
Evaluation

Per-operation type rate limiting



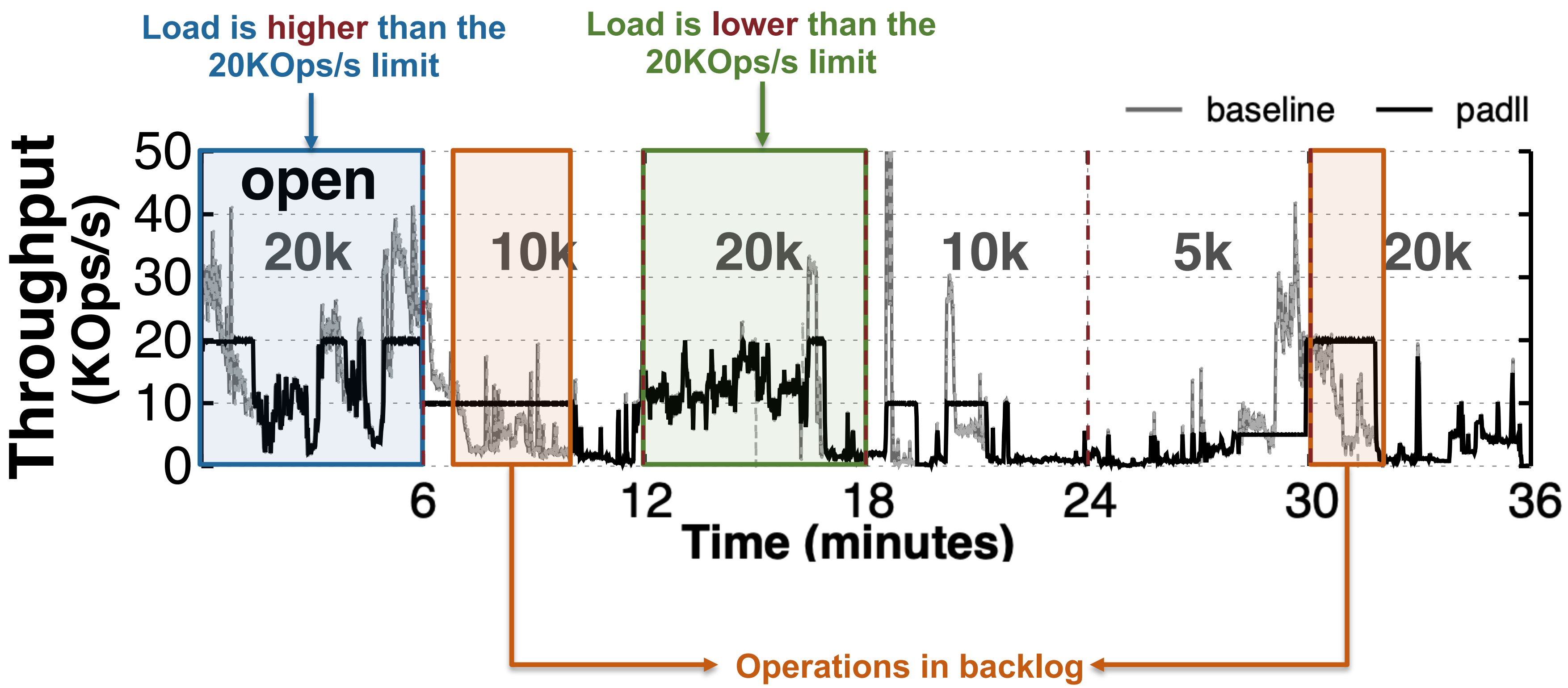
Evaluation

Per-operation type rate limiting



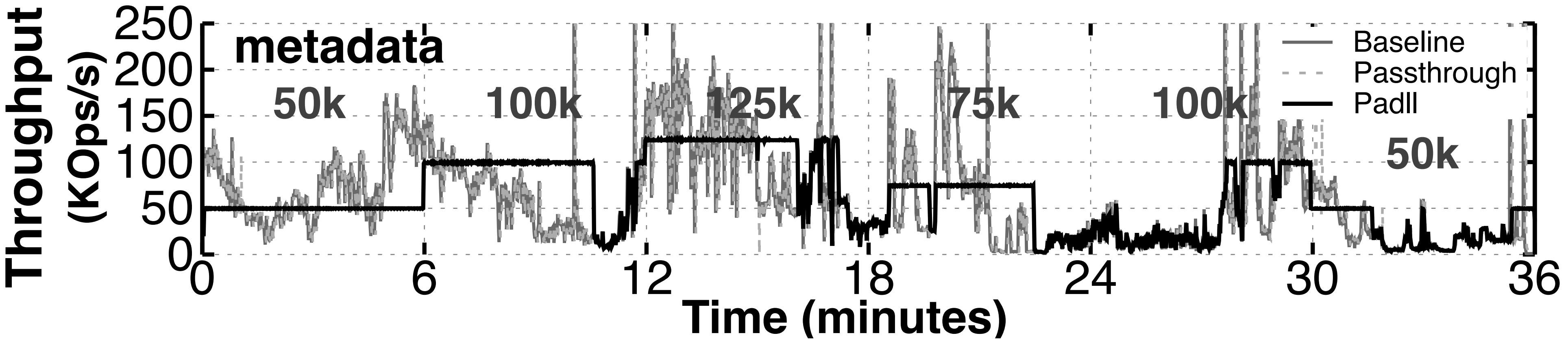
Evaluation

Per-operation type rate limiting



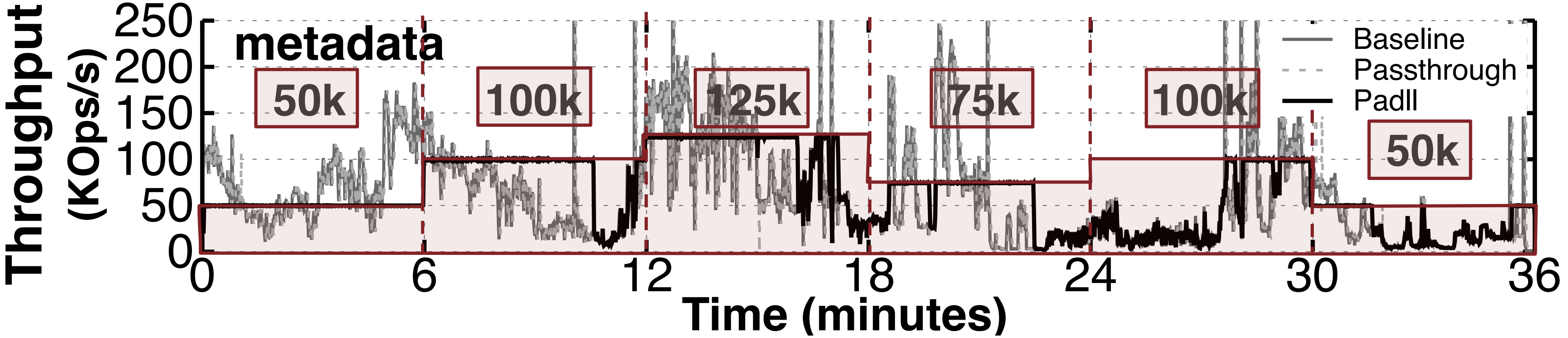
Evaluation

Per-operation class rate limiting



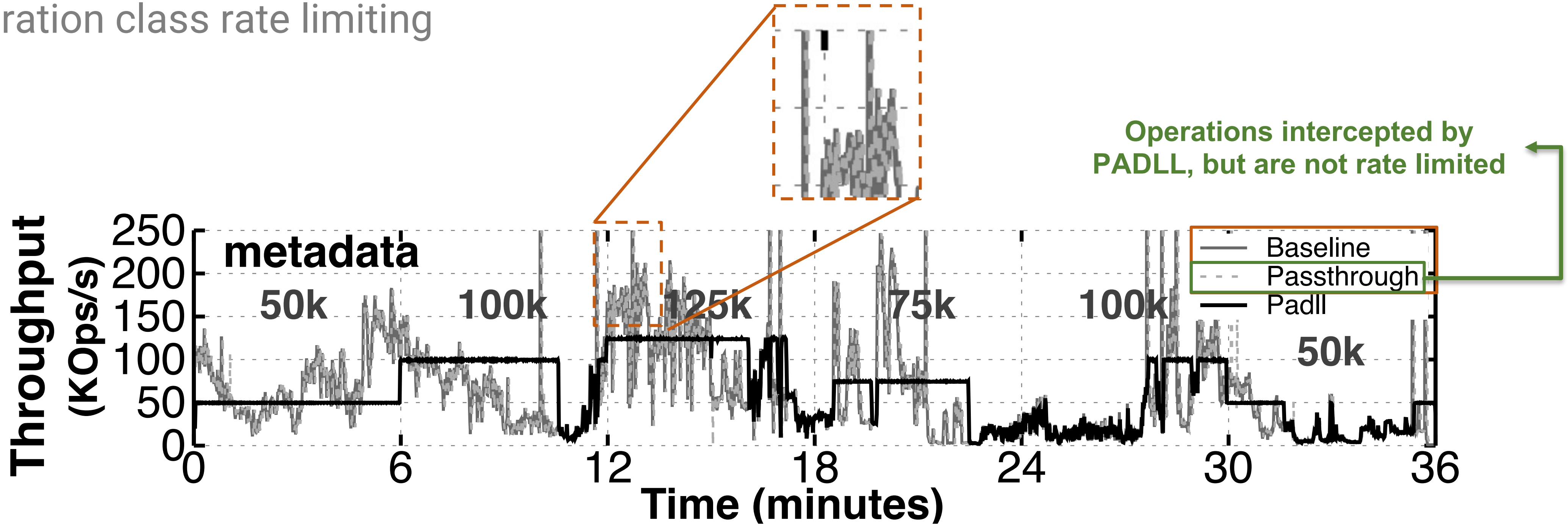
Evaluation

Per-operation class rate limiting



Evaluation

Per-operation class rate limiting



Overhead is negligible!

Less than 0.9% across all experiments.

Evaluation

Per-job rate limiting and QoS control

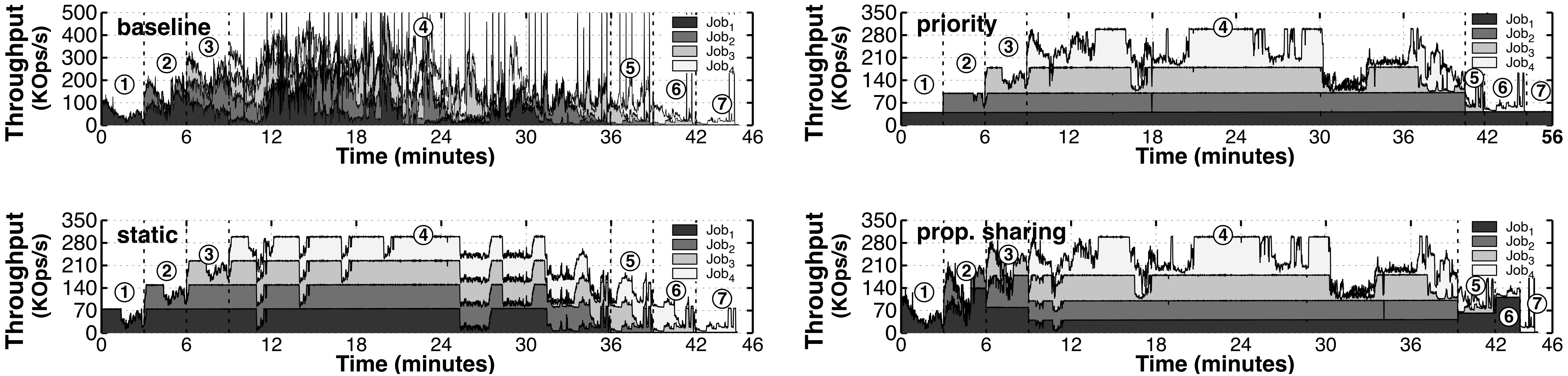
- Multi-job, global orchestration experiment.
- Setup:
  - There are at most **four jobs** in the system, each running the same workload.
  - Jobs are incrementally added to the system every 3 minutes.
  - When limiting, the PFS's **maximum** metadata rate is set to **300 KOps/s**.

Evaluation

Per-job rate limiting and QoS control

- Setup:
    - **Baseline**
      All jobs execute without being rate limited.
    - **Static**
      Each job is rate limited with the same priority (75 KOps/s).
    - **Priority**
      Each job is rate limited with a different priority (40, 60, 80 and 120 KOps/s).
    - **Proportional sharing**
      Control algorithm that enforces *per-job metadata rate reservations*.

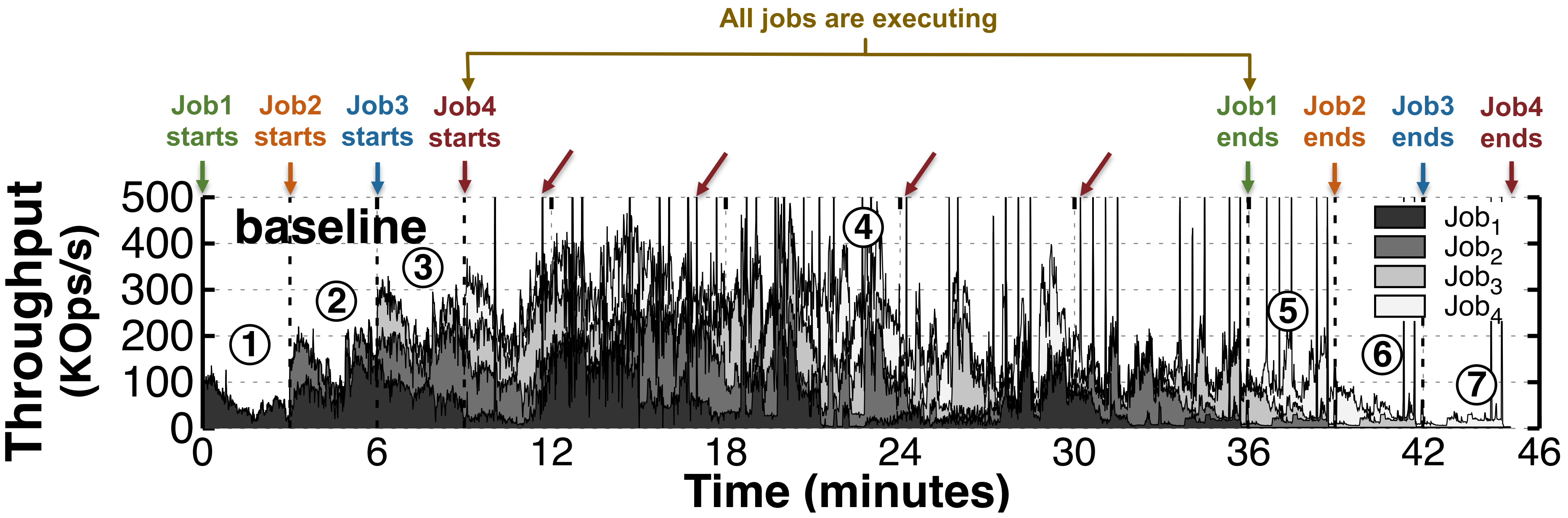
Evaluation
Per-job rate limiting and QoS control



Evaluation

Per-job rate limiting and QoS control

Baseline: All jobs execute without being rate limited.

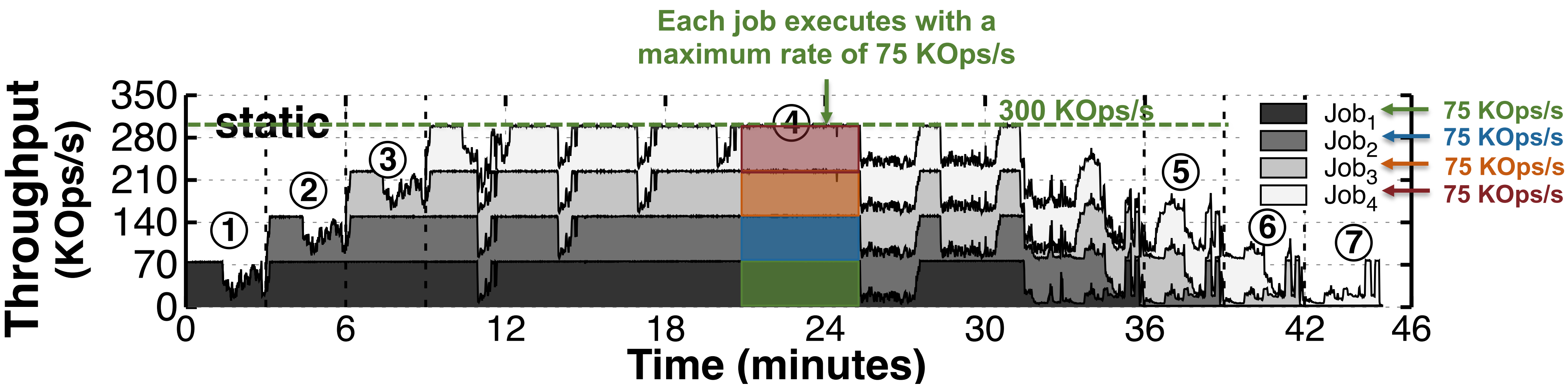


⚠️ **Volatile and bursty workload!**

Evaluation

Per-job rate limiting and QoS control

Static: Each job is rate limited with the same priority (**75 KOps/s**).



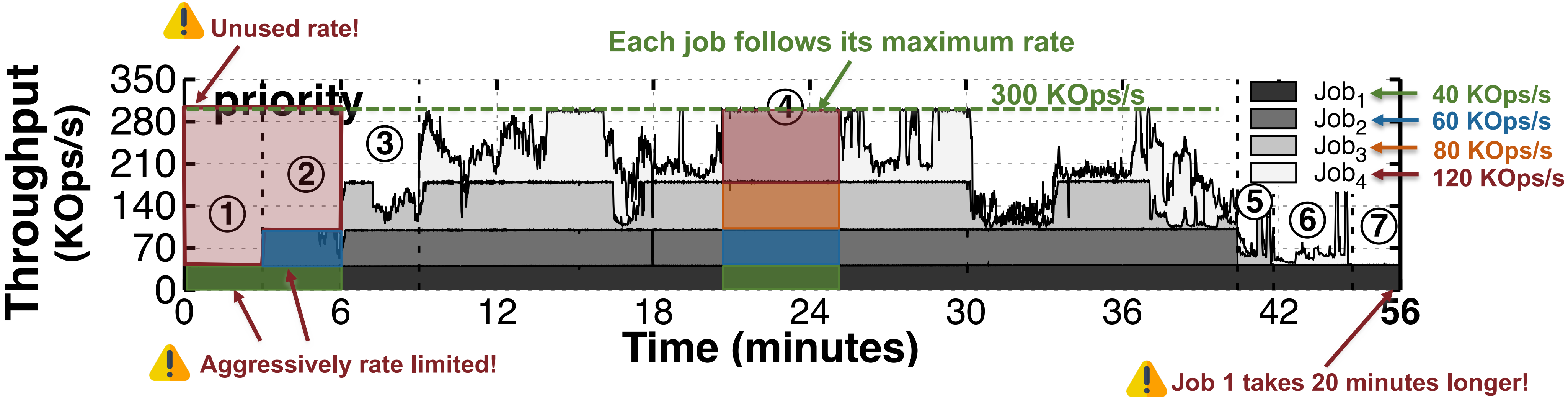
✔ **Sustained throughput and prevents bursty workloads!**

But, what if we want to enforce different priorities?

Evaluation

Per-job rate limiting and QoS control

Priority: Each job is rate limited with a different priority (**40**, **60**, **80** and **120 KOps/s**).



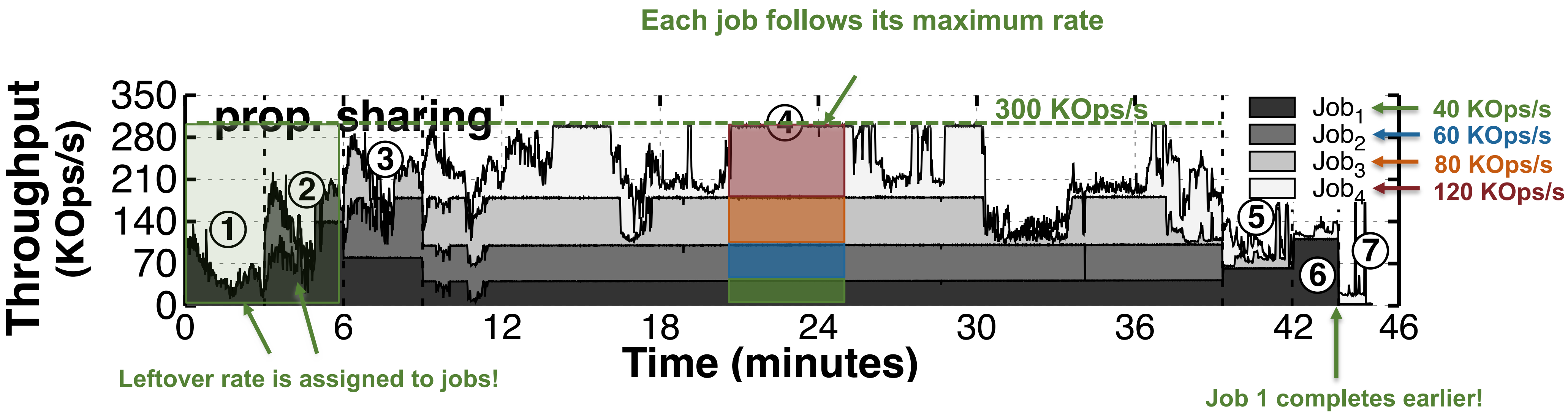
✔ **Jobs have different priorities!**

⚠ **Jobs can be unnecessarily rate limited!**

Evaluation

Per-job rate limiting and QoS control

Proportional sharing: Control algorithm that enforces *per-job metadata rate reservations.*



✓ Leftover rate is assigned to jobs!

✓ Each reservation of metadata is respected!

Discussion

- PADLL is able to:
    - **Control** the rate of I/O workflows - *data, metadata* - at **different granularities** – *type, class, per-job*.
    - **Prevent I/O burstiness**.
    - Ensure I/O fairness and **prioritization**.
    - Orchestrate the storage system **holistically**.

Note: To prevent overloading the production PFS, all metadata workloads were submitted to the local file system, however, *IOR* experiments were conducted using the PFS.

Conclusion

- **PADLL** is an **application and PFS-agnostic** storage middleware that enables **enforcing QoS policies** over workflows in HPC clusters.

- Enables system administrators to **proactively** and **holistically control the I/O rate** of all running jobs.

- **Prevent metadata-aggressive** jobs **from harming the PFS**, as well as other jobs in execution.

Future Work

- **Control algorithms**
  - *Explore other algorithms and analyze their impact in PFSs in production.*
- **Control plane scalability**
  - *The control plane is a centralized component, thus investigating its scalability and dependability is fundamental.*
- **Additional experiments**
  - *Evaluate with large-scale I/O applications (e.g., Tensorflow) with different I/O workloads and access patterns.*
  - *Evaluate the performance impact of PADLL for saturated PFSs.*

2nd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO'22)

Protecting Metadata Servers From Harm Through Application-level I/O Control

*Ricardo Macedo, **Mariana Miranda**, João Paulo*
INESC TEC & University of Minho

Amit Ruhela, Stephen Lien Harrell
TACC & UTAustin

Richard Todd Evans
Intel

Yusuke Tanimura, Jason Haga
AIST

September 6, 2022