

Parallel Tools for Large-Scale File Migration and Verification on HPC Storage Systems

Sergey Noskov
30.09.2026
LAD 26

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ



OUTLOOK FOR THE NEXT 25 MINUTES

- Motivation – Why the Tool Was Created
- Problem Statement
- Architecture
 - Scheduler and worker model
 - Worker operations: rsync, copy, diff, lustre OST migration
 - Lustre file migration helper
- Benchmarks
- Conclusion

MOTIVATION

- Copy very large directory trees
 - 10+ millions files
 - 100+ TiB
- Preserve file metadata
 - permissions, ownership and timestamps
 - extended attributes
- Exploit filesystem parallelism
- Verify data during or after the copy
- Continue processing despite individual file errors

MOTIVATION

Move data efficiently,
while keeping the load
on the storage system under control.

MOTIVATION

EXISTING APPROACHES

- Simple parallel approaches:
 - Fpsync
 - partitions the directory tree
 - runs multiple copy/sync processes in parallel
 - gnu parallel + rsync/cp/...
 - parsync - a parallel rsync wrapper in Perl
 - xcp – a rust program
- Large-scale parallel copy tools:
 - mpiFileUtils
 - MPI-based
 - designed for large HPC datasets
 - pftool (Parallel File Tool)
 - multi-process / multi-node architecture

PROBLEM STATEMENT

We needed a tool that could:

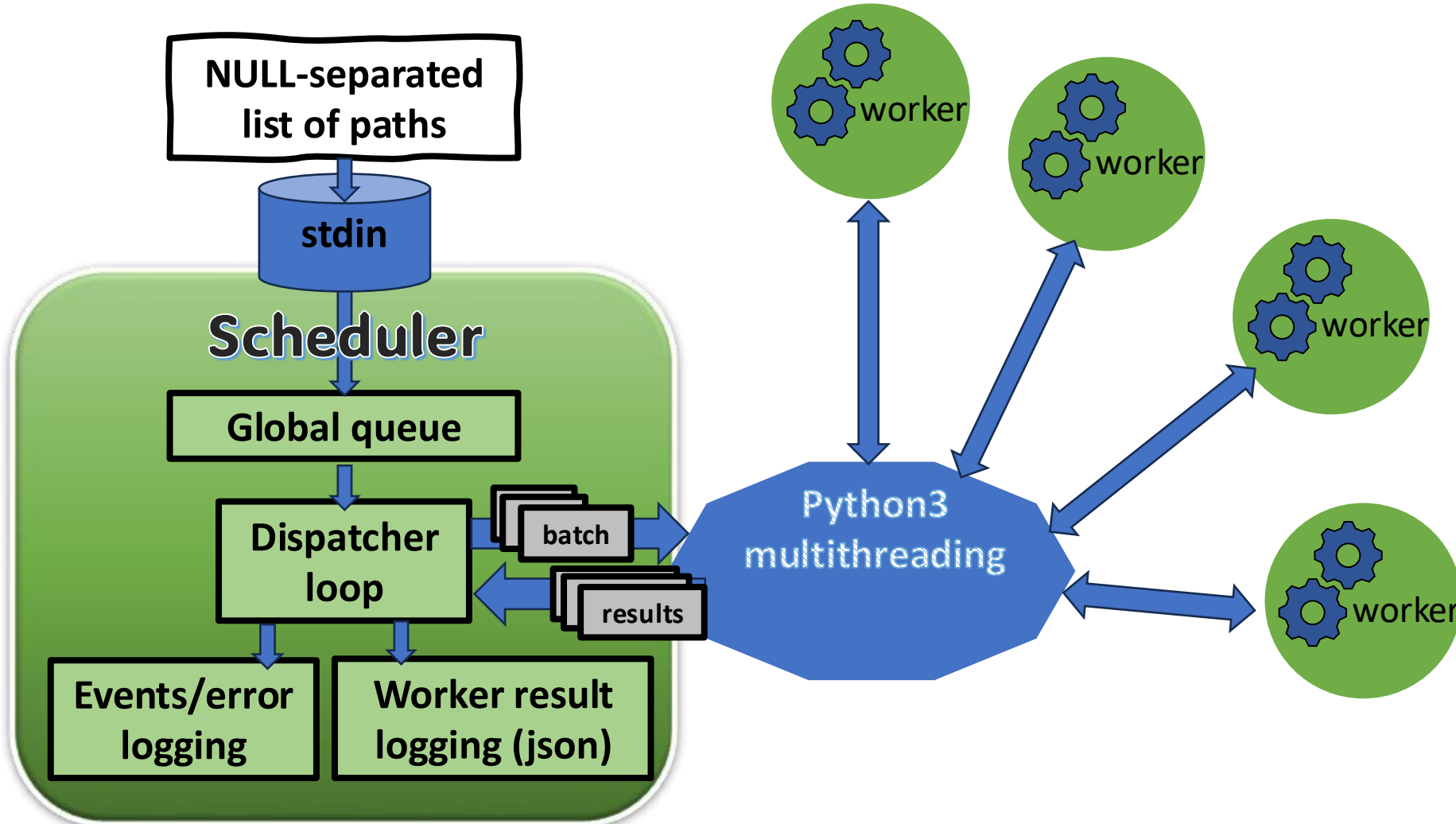
- process millions of files continuously
- use multiple workers without requiring MPI
- dynamically tune the workload profile by changing
 - worker count
 - batch size
- handle very long paths
- preserve filesystem metadata
- verify copied data
- survive individual file and filesystem errors
- produce machine-readable per-file status information
- support Lustre-specific operations such as OST migration
 - One scheduler, multiple filesystem operations

PROBLEM STATEMENT

We needed a tool that could:

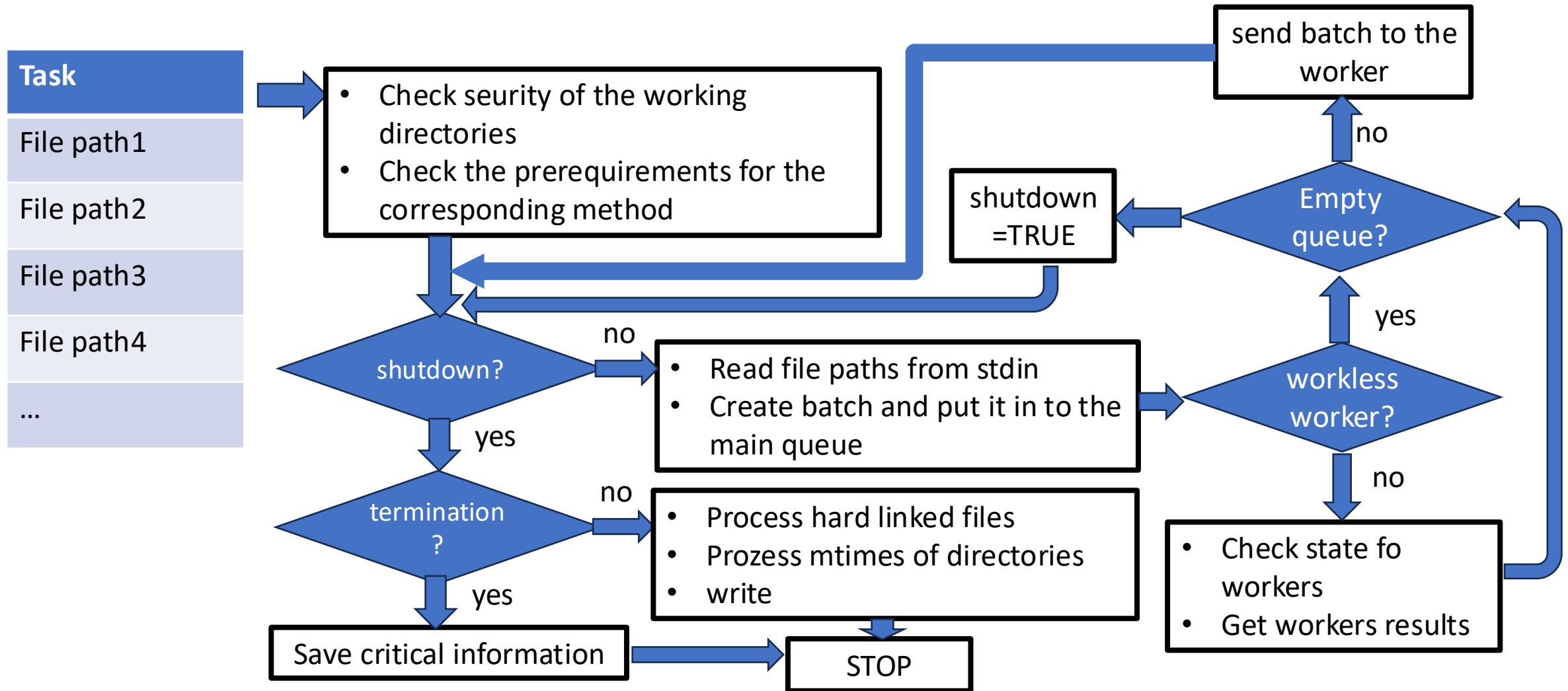
- process millions of files continuously 
- use multiple workers without requiring MPI 
- dynamically tune the workload profile by changing
 - worker count 
 - batch size 
- handle very long paths 
- preserve filesystem metadata
- verify copied data 
- survive individual file and filesystem errors 
- produce machine-readable per-file status information 
- support Lustre-specific operations such as OST migration 
→ One scheduler, multiple filesystem operations 

ARCHITECTURE

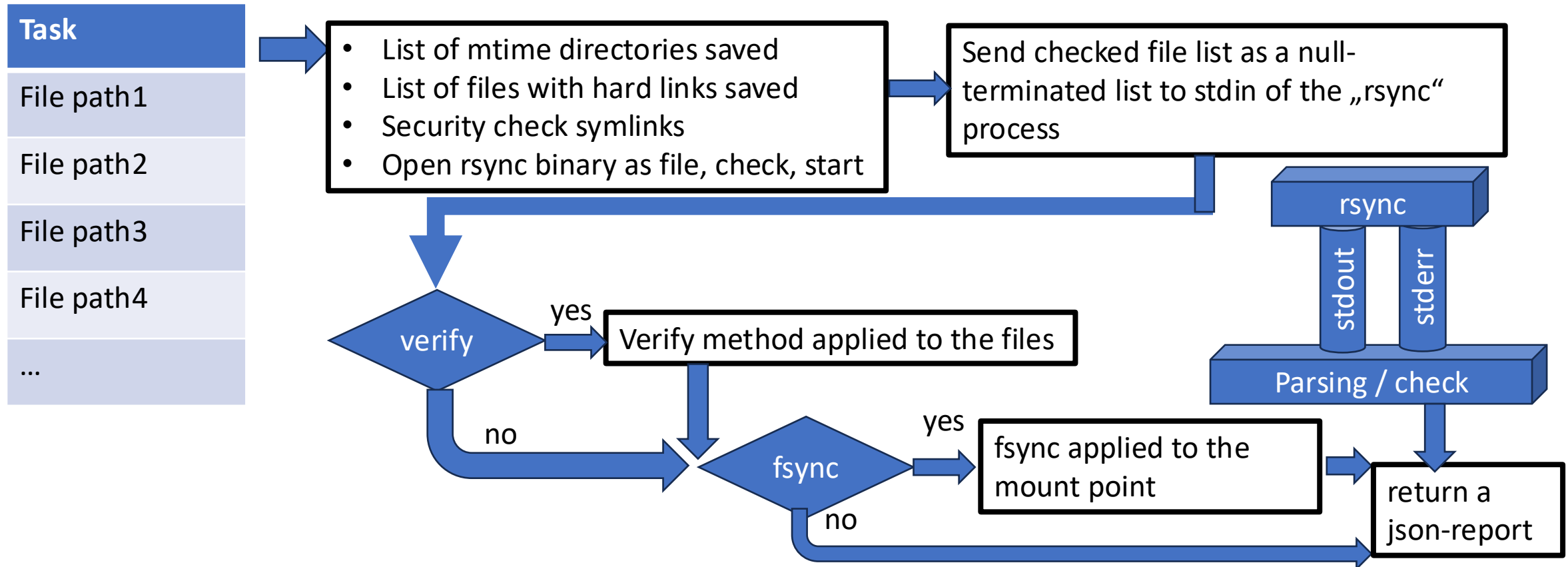


Operation type (selected at startup)
rsync process
internal copy function
External copy program
Internal diff function
Lustre OST migration helper

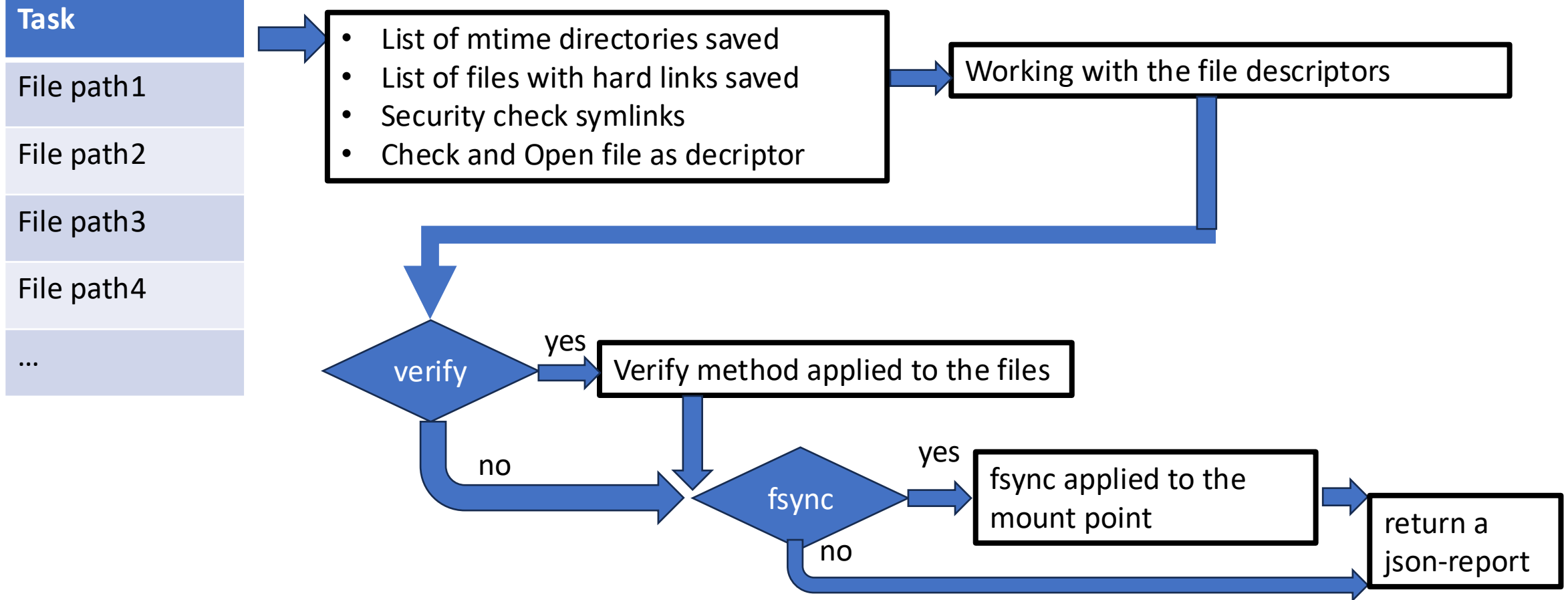
SCHEDULER WORKFLOW



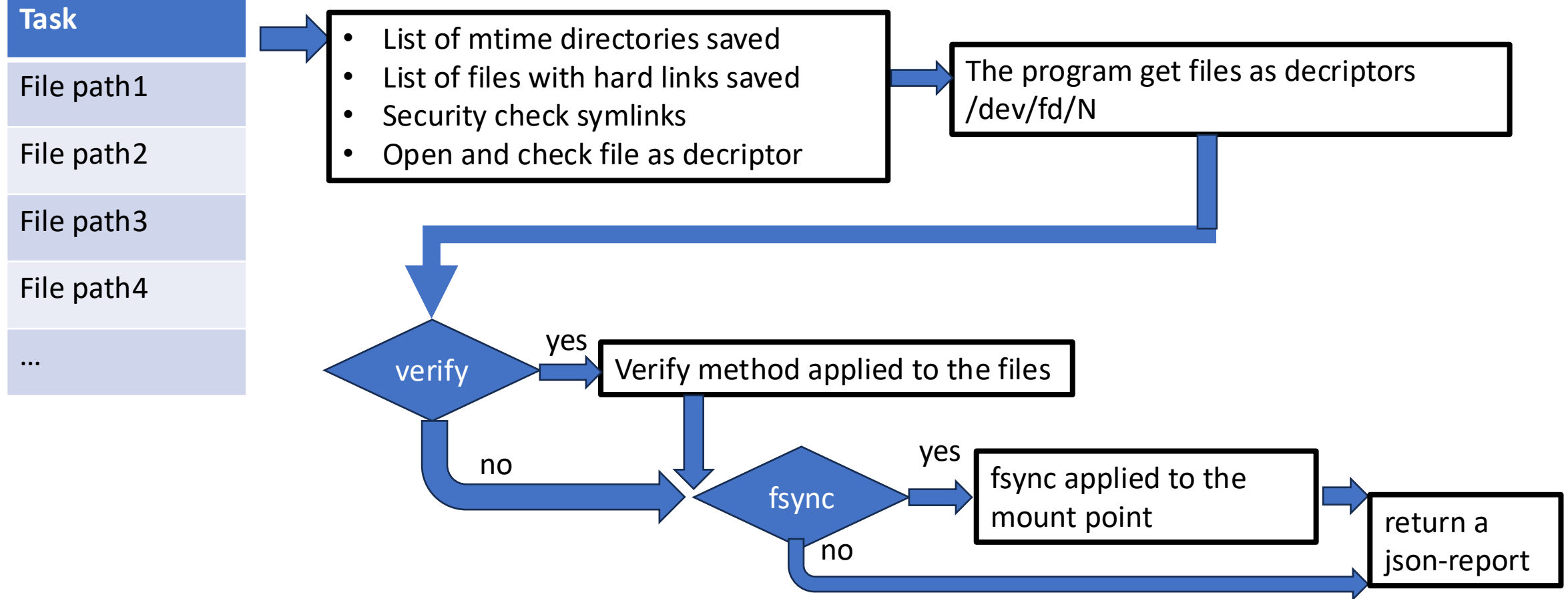
WORKER OPERATION RSYNC



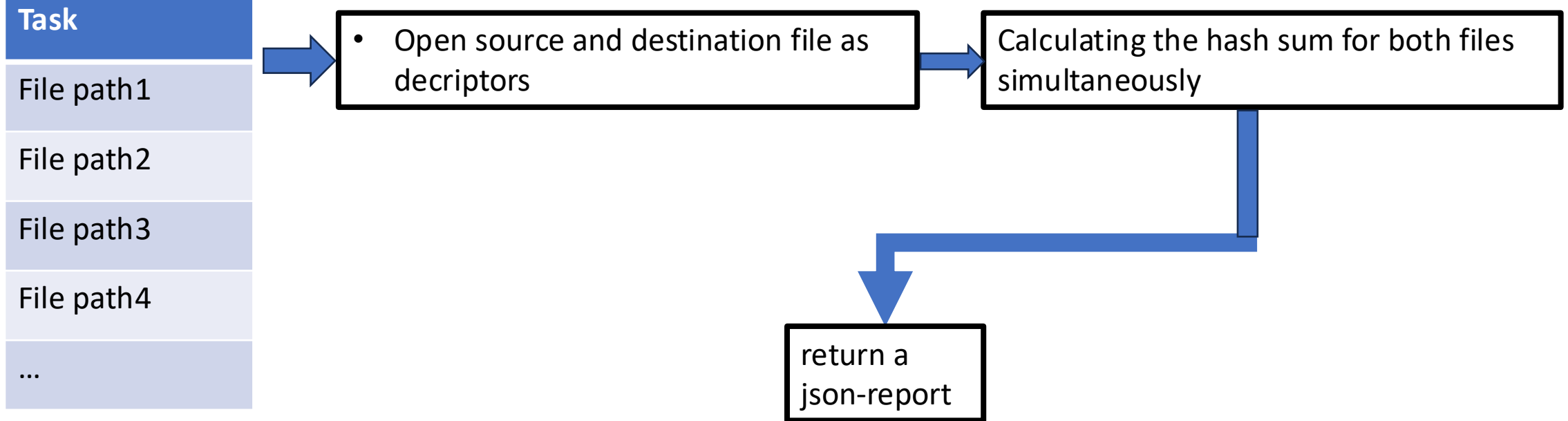
WORKER OPERATION INTERNAL COPY FUNCTION



WORKER OPERATION EXTERNAL COPY PROGRAM



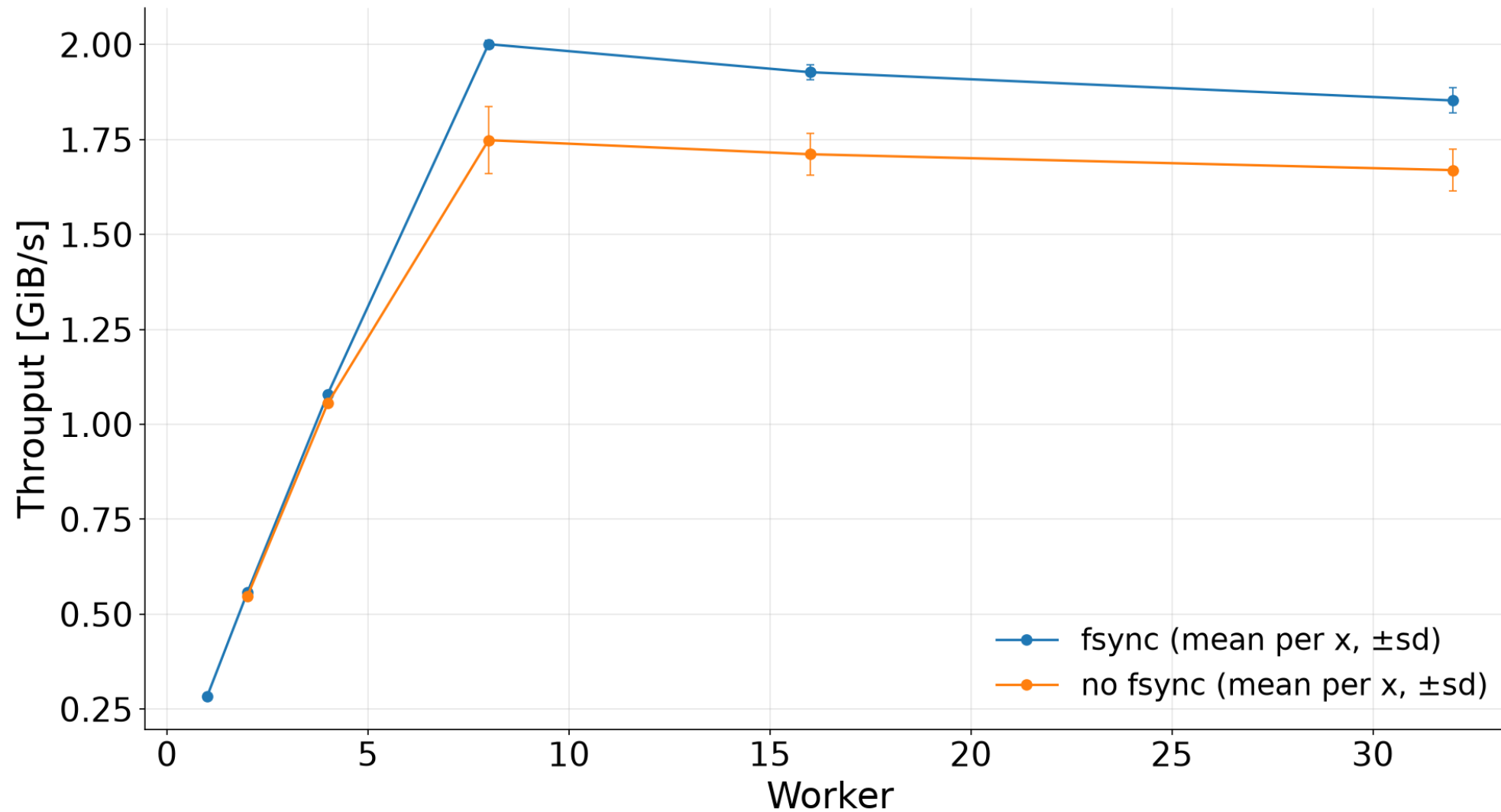
WORKER OPERATION INTERNAL DIFF FUNCTION



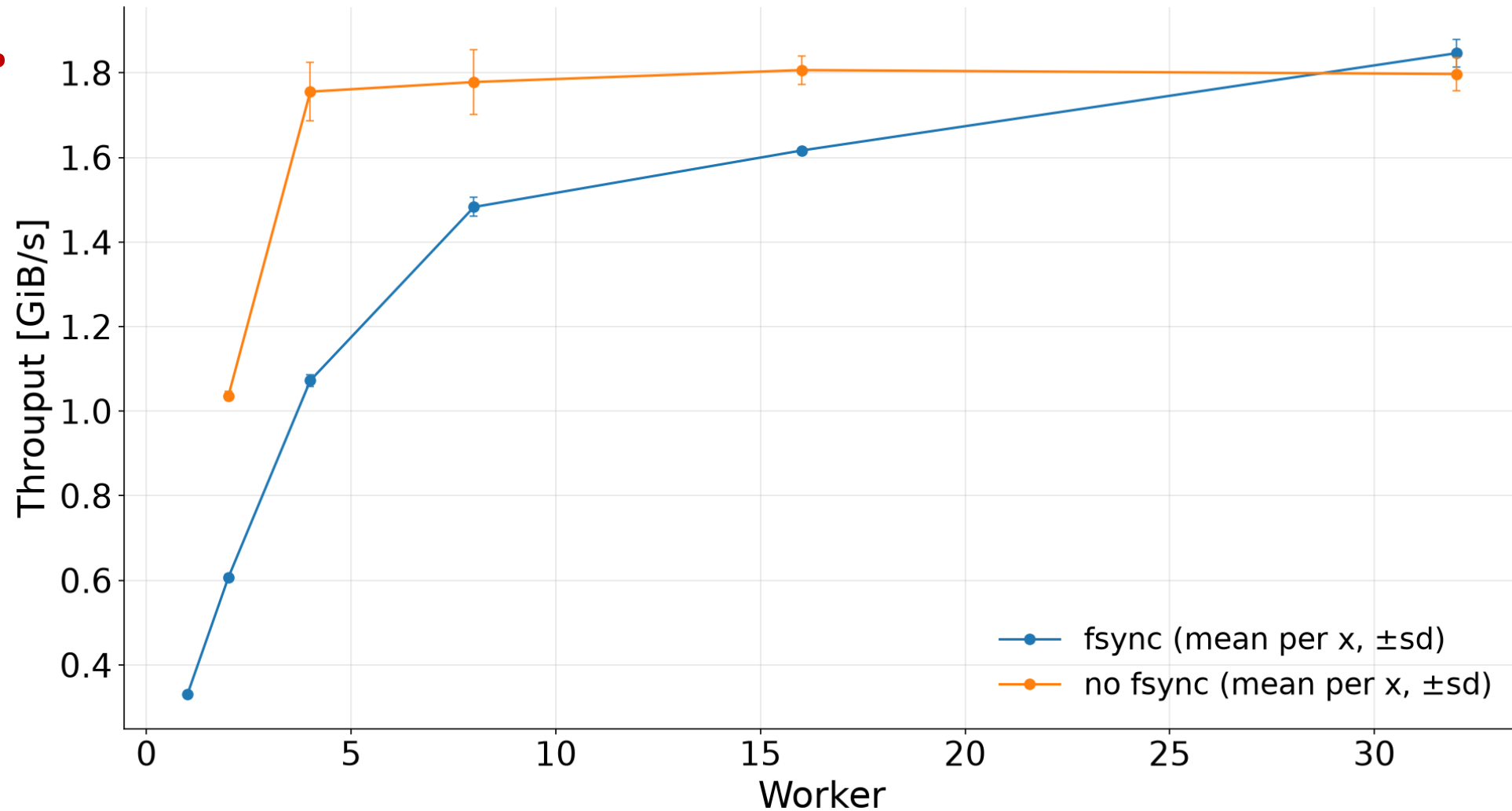
BENCHMARKING: LUSTRE STORAGE

Metadata storage	Object storage
HA pair 2x PowerEdge R650	HA pair 2x PowerEdge R650
RAID Controller Dell PowerVault ME5024	EMC Storage Expansion Enclosure ME484
SSD RAID based MDT+MGT	ZFS DRAID based OST
mgt 50Gb 2x mdt 7 TB	4x OST 291TB draid3:6d:21c:1s
Infiniband bond 2x100Gbit (backup mode) Ethernet bond lacp(L3+4) 2x100Gbit	Infiniband bond 2x100Gbit (backup mode) Ethernet bond lacp(L3+4) 2x100Gbit

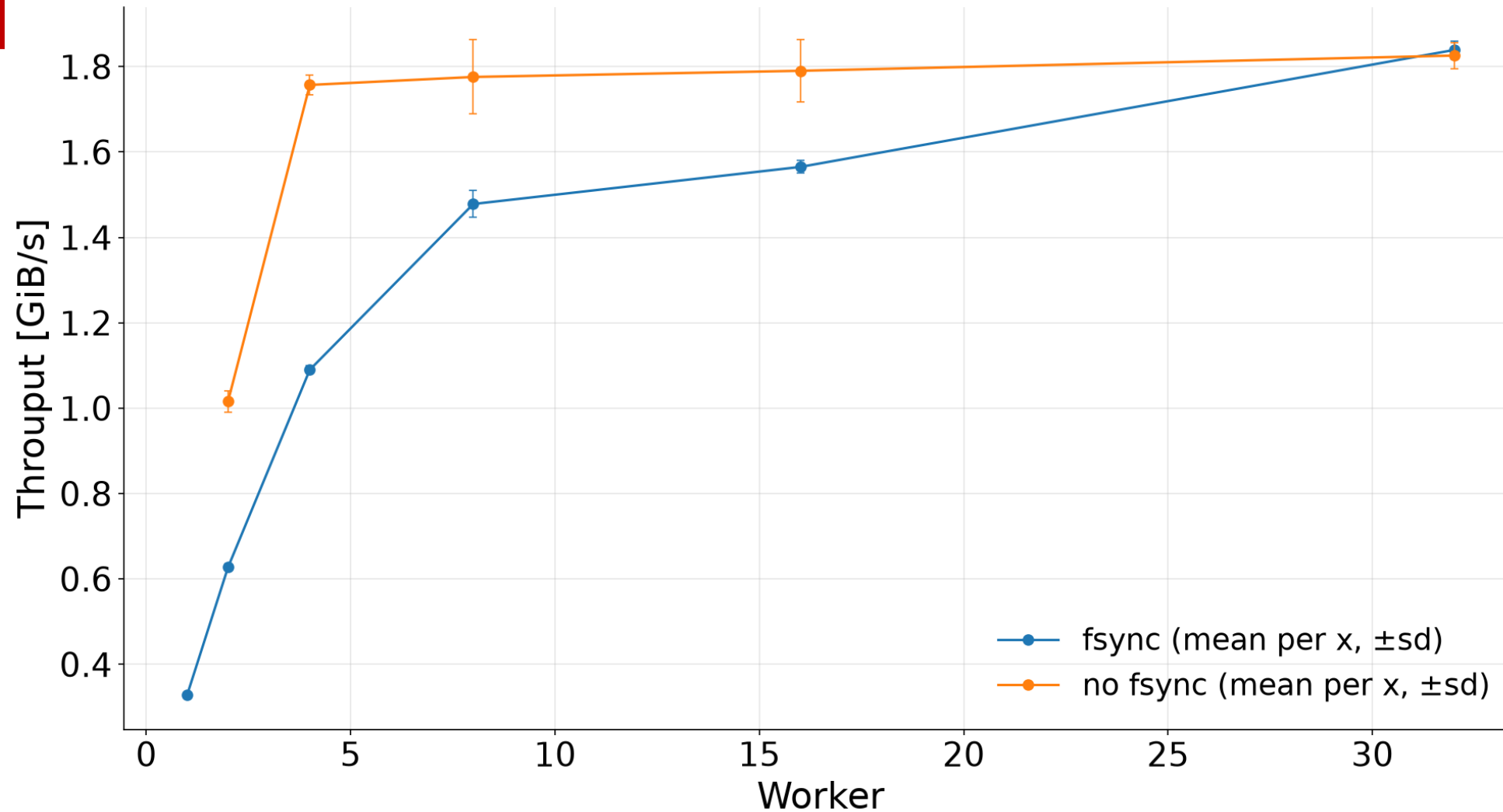
SCALING RSYNC OPERATION (1.18 TIB)



SCALING INTERNAL COPY OPERATION (1.

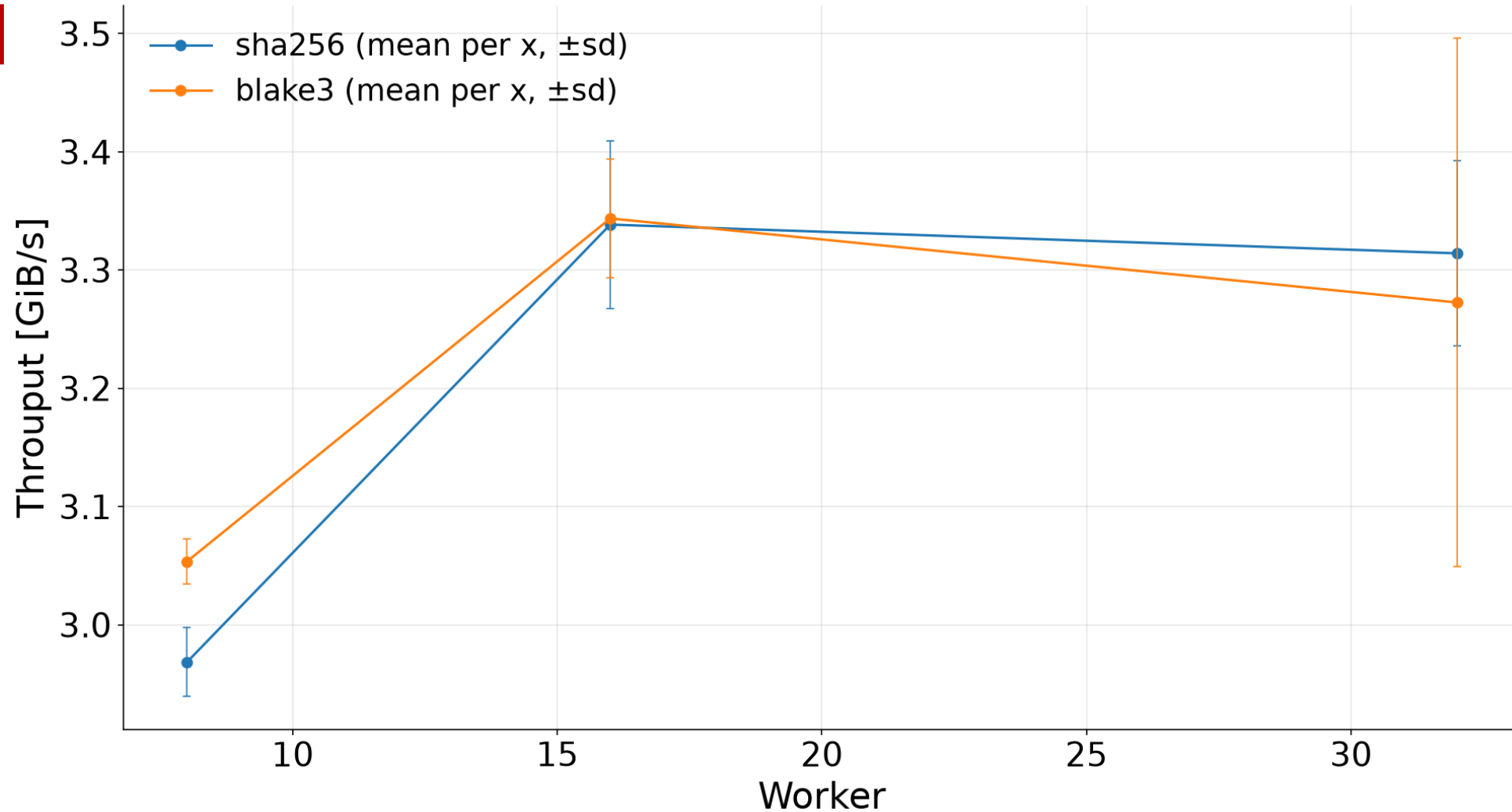


SCALING LINUX CP PROGRAMM OPERATION (1)



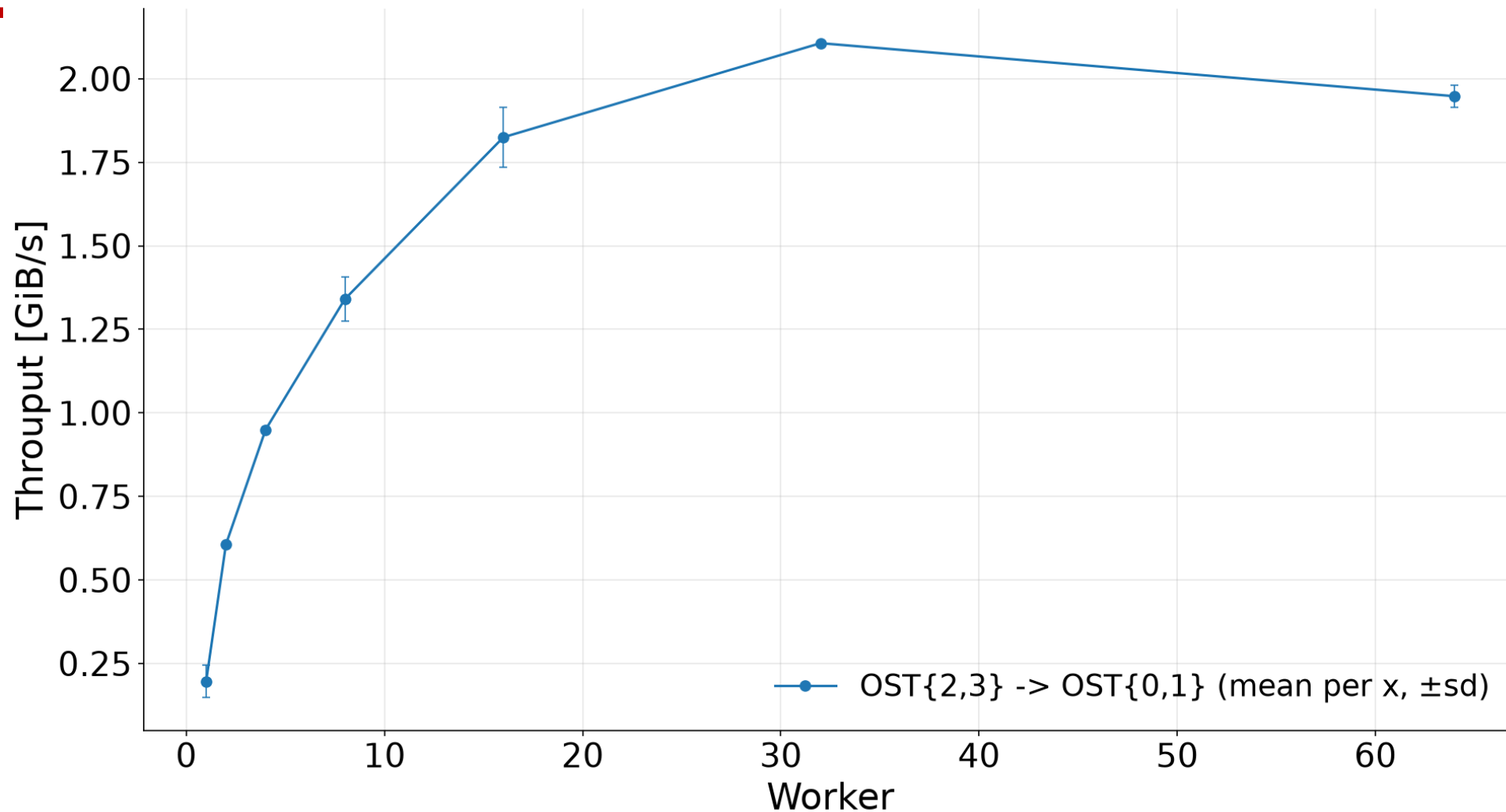
SCALING PARALLEL VERIFICATION (1.18

TI



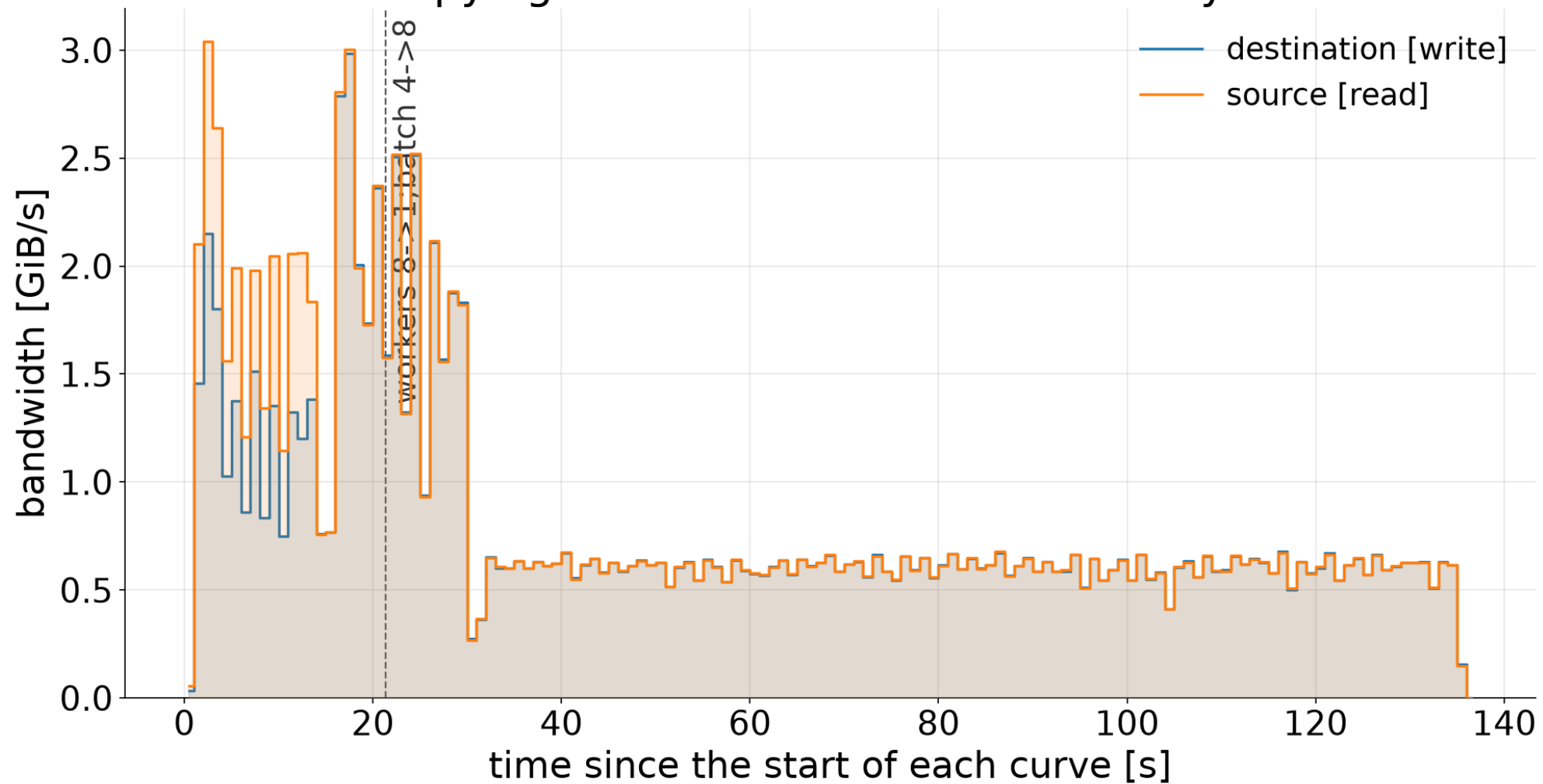
SCALING LUSTRE OST MIGRATION (1.18)

T



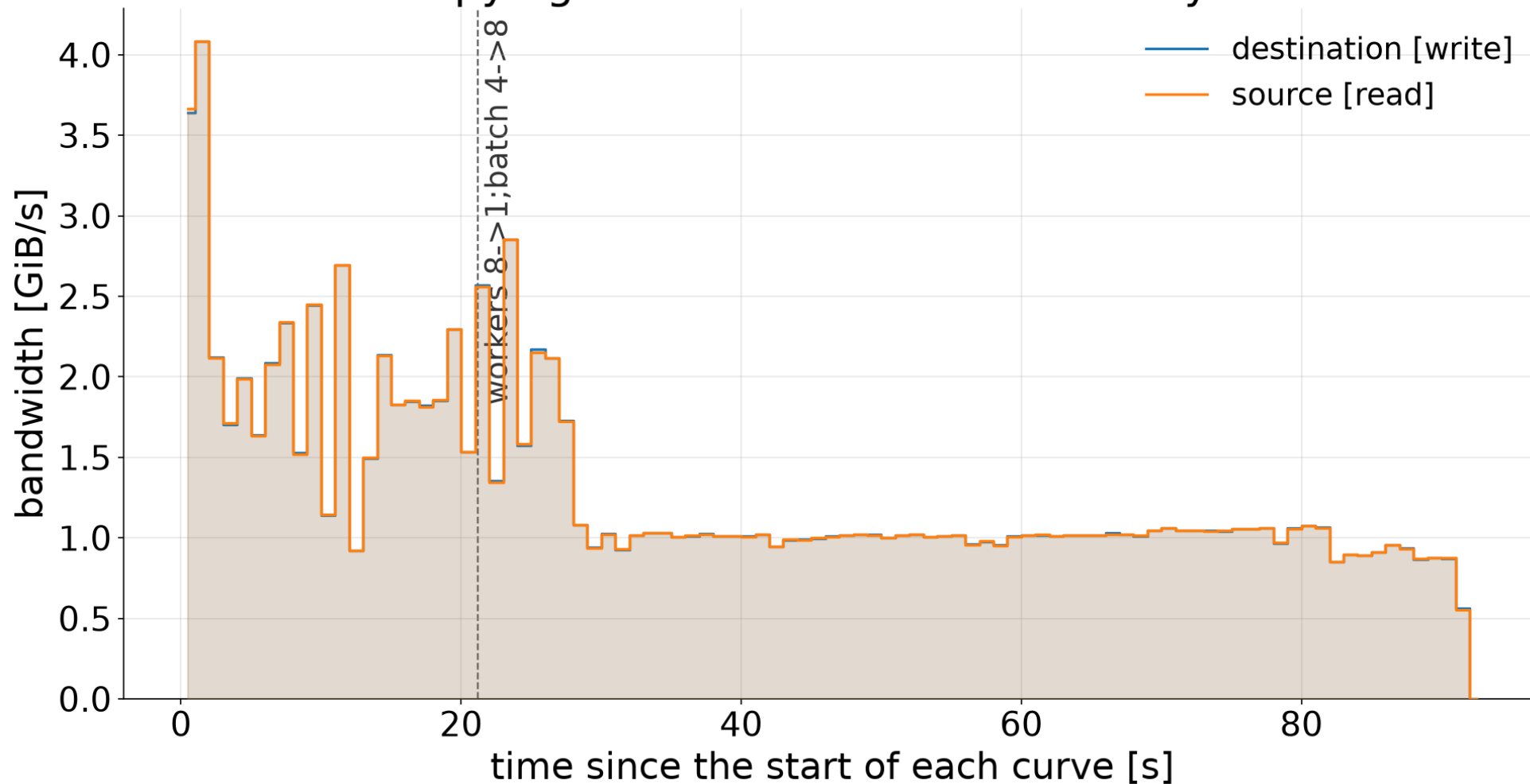
DARSHAN DXT I/O TRACES

copying 121 GiB with 8 worker with fsync



DARSHAN DXT I/O TRACES

copying 121 GiB with 8 worker no fsync



CONCLUSION

- One scheduler for different file operations
- Effective parallel scaling up to 8–16 workers per node in our tests
- Worker count and batch size allow dynamic workload tuning
- Different backends can use the same scheduling framework
- Beyond a certain point, performance is limited by the storage system rather than the scheduler

THANK YOU FOR YOUR ATTENTION

We would like to express my sincere thanks to my colleagues for interesting discussions and ideas.