Karlsruhe Institute of Technology

On-demand File Systems

Production use, use cases and lessons learned

Mehmet Soysal
Steinbuch Centre for Computing (SCC)

Overview

- Motivation / Background
- Integration into HPC batch system
- Generic Benchmarks
- Use Cases
  - OpenFOAM, NAStJA, FLAPS
- Summary and Conclusion

Motivation / Background

- ADA-FS project within SPPEXA
  - ZDV@Mainz, ZIH@Dresden, SCC@KIT
  - Tackling the I/O bottleneck on HPC Systems
  - Use of on-demand file systems (ODFS)
- At SCC we have:
  - traditionally node-local storage (HDD, SSD, NVMe …)
  - heavy I/O use cases

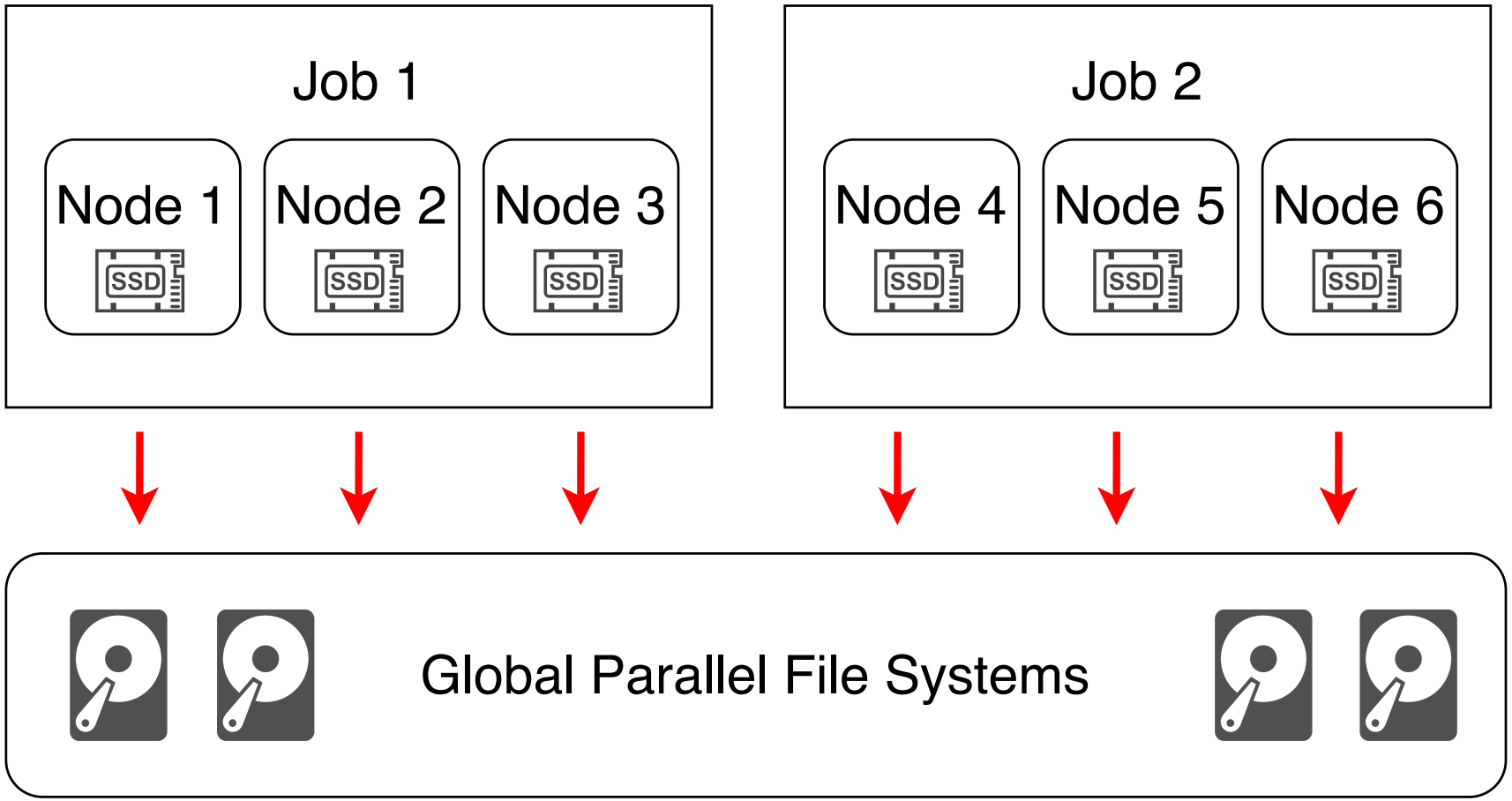
- We needed a solution to handle heavy I/O until use cases are optimized

Motivation / Background (1)

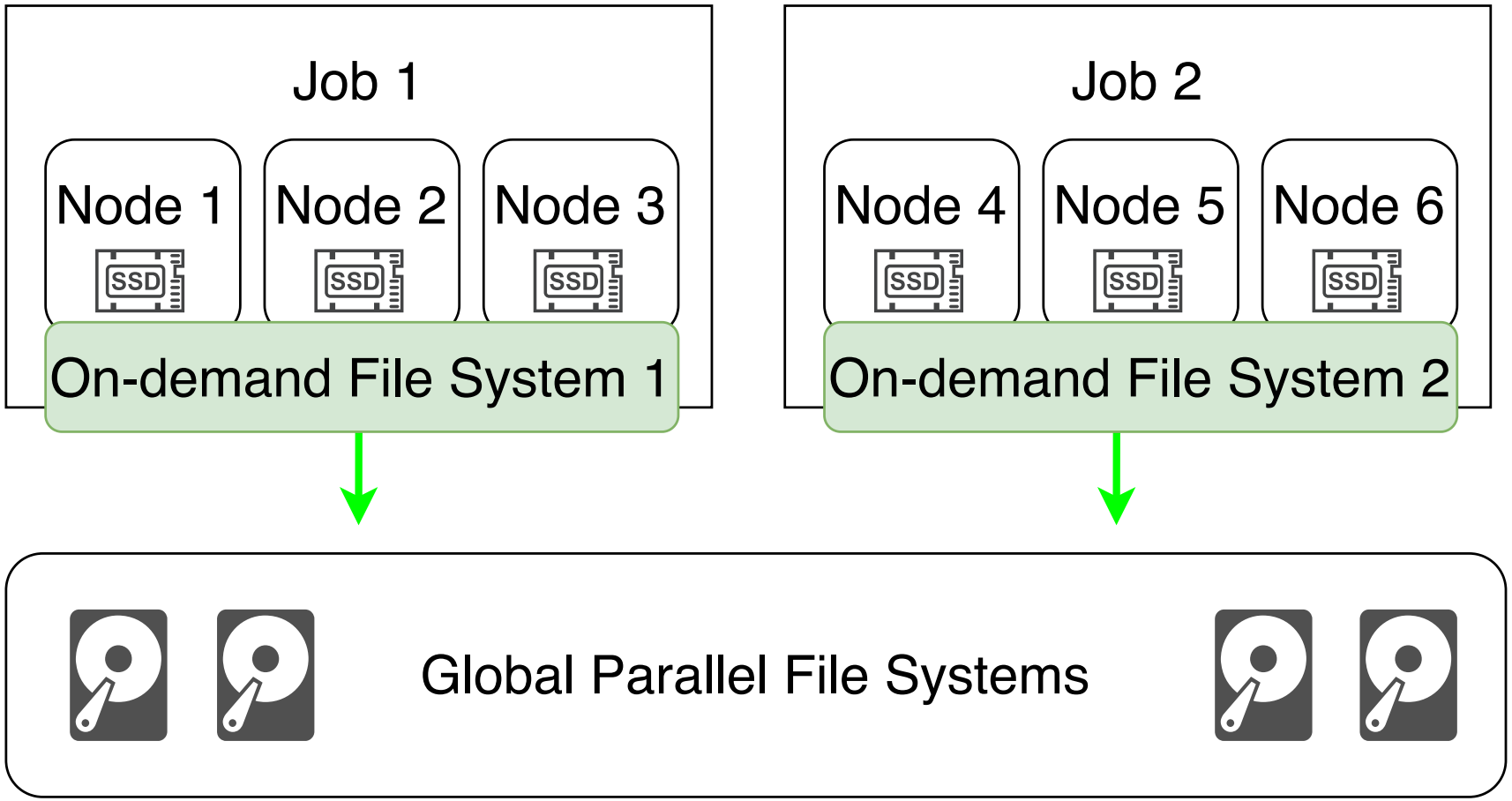
- ForHLR II
  - Compute nodes with 400GB SATA-SSD
  - ~550/450 MB/s R/W per SSD
- Horeka
  - Compute nodes with 1TB NVMe-SSD
  - ~3000/1000 MB/s R/W per NVMe

- Motivation / Background
- **Integration into HPC batch system**
- Generic Benchmarks
- Use Cases
  - OpenFOAM, NAStJA, ML/DL (Gromacs)
- Summary and Conclusion

Using Global Parallel File System



Using On-demand File System

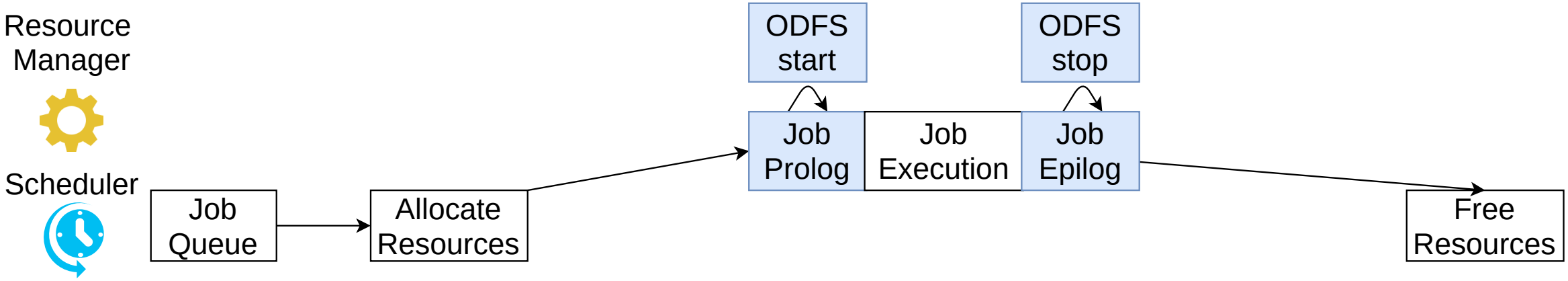


Integration into Batch System

- BeeGFS / BeeOND is used as ODFS for production use
  - Other FS planned, e.g., GekkoFS[1] …
- Only on-demand – minimal changes to system usage
  - Request on job submission “--constraint BEEOND”
- Different implementations
  - MOAB + (SLURM) + data staging
  - SLURM + Burst Buffer
  - SLURM + Pro-/Epilog
- Loopback device as storage

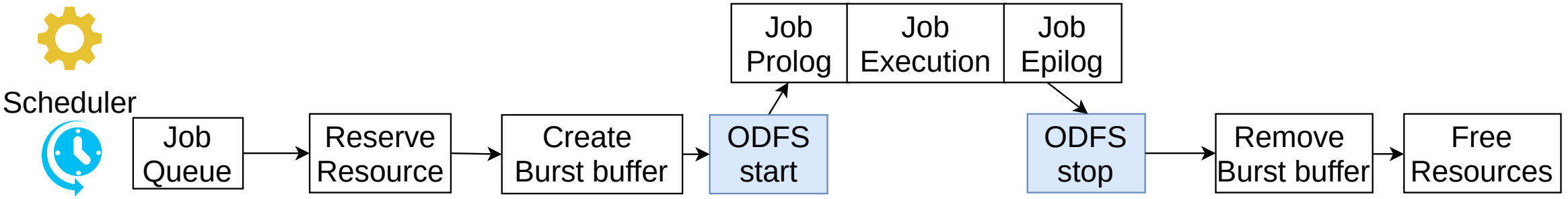
1) Vef et al. “GekkoFS - A Temporary Distributed File System for HPC Applications”

Integration using Job Pro-/Epilog



- Simple integration
  - ODFS start/stop in pro/-epilogue
- No data staging (User must move data within job)
- Very robust

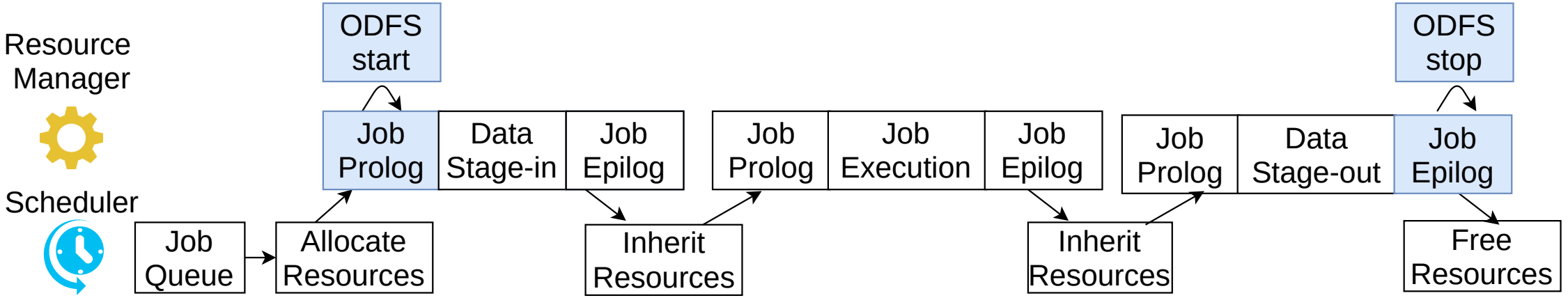
Extending Slurm Burst Buffer



- Extending the Burst Buffer plugin
  - Only proof of concept
- Creation of a reservation for a burst buffer

1) Bachelorthesis : Valentin Voigt “**Entwicklung eines on-demand Burst-Buffer-Plugins für HPC-Batch-Systeme”**

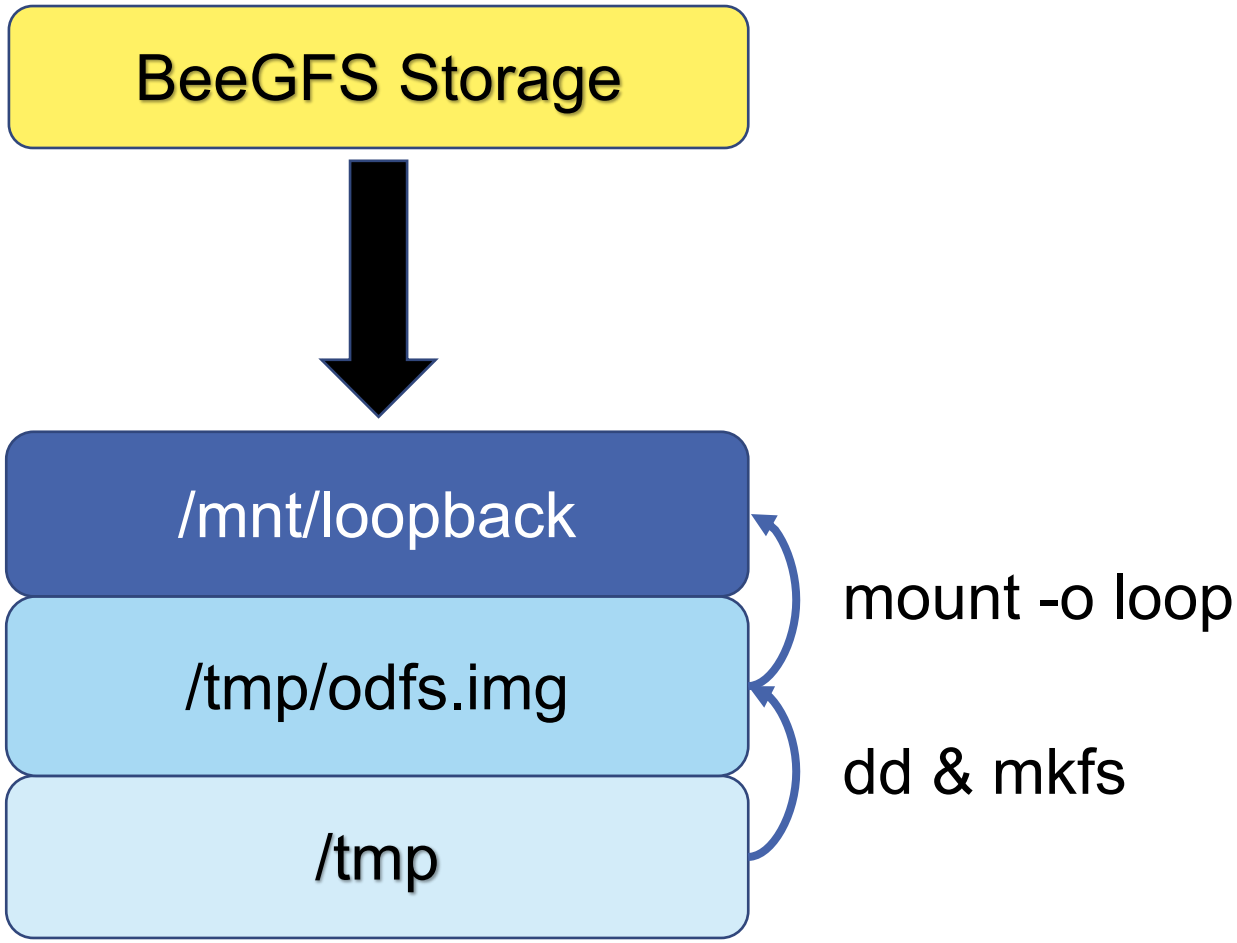
Moab and Data Staging



- MOAB's data staging feature extended
  - Job is split in three sub-jobs
- MOAB offers special feature to inherit a resource allocation

Loopback device

- Using loopback device as storage target
  - Control disk usage by img file size
  - Use different file system (xfs, ext4 …) on img file
  - Fast and simple cleanup
  - Speed boost through memory buffering[1]



Yamamoto et al., Analysis and elimination of client evictions on a large scale lustre based file system. Presentation at the Lustre User Group 15 (LUG'15), 2015.

Summary – Integration

- The simplest solution is usually the best (Pro-/Epilogue)
- One storage target per node
- Using loopback device as storage target
- Pre-create directories with different stripe settings

- Motivation / Background
- Integration into HPC batch system
- **Generic Benchmarks**
- Use Cases
  - OpenFOAM, NAStJA, ML/DL (Gromacs)
- Summary and Conclusion

Generic Benchmarks

- Start/Stop time
- Read/Write Throughput
- Metadata performance
  - Multiple Metadata Server (MDS)
- Topology awareness

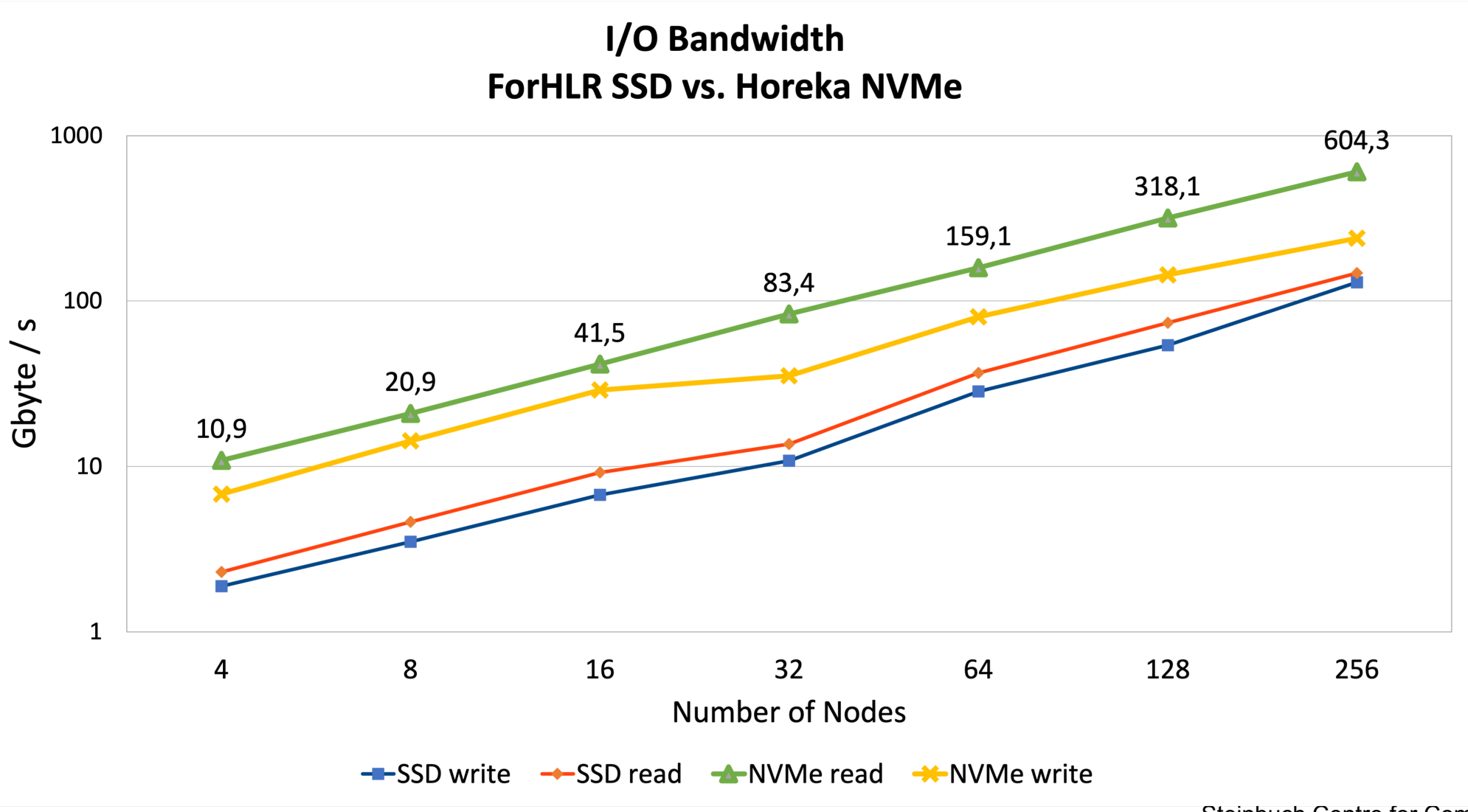
Startup and Shutdown of BeeGFS

Nodes	8	16	32	64	128	256
Startup (s)	10.2	16,7	29,3	56,5	152,1	222,4
Shutdown (s)	11,9	12,1	9,4	15,9	36,1	81,1

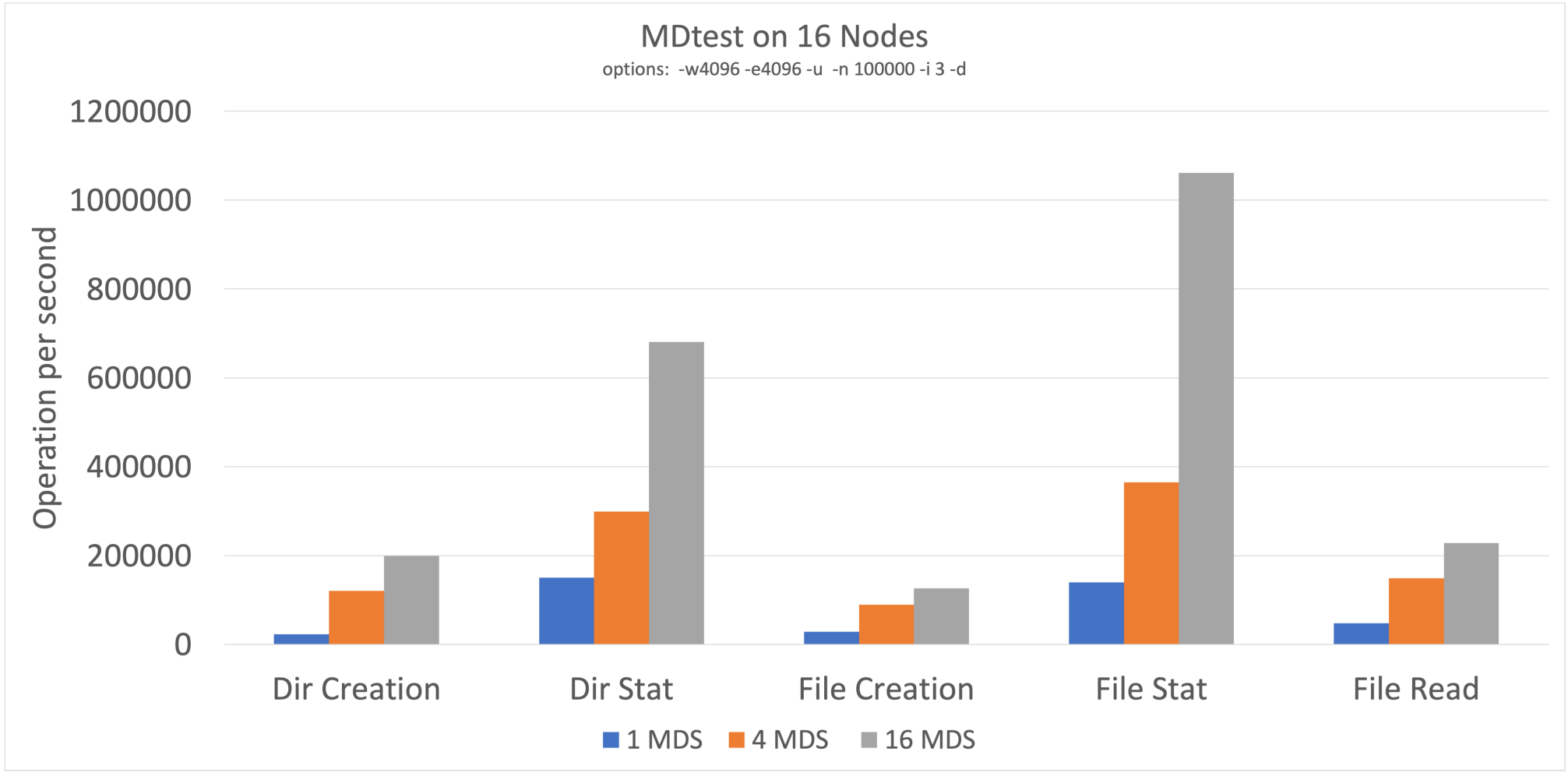
- Results above with beeond script using pdsh (clean start/stop)
- Probably to long for very short jobs
- With optimization possible to start on 256 nodes within a minute[1]
- Shutdown can be done within seconds (“stoplocal”)

1) Mehmet Soysal “Speeding up beeond startup” https://groups.google.com/g/fhgfs-user/c/g8ysFS35Ucs/m/E-RtxKyiCAAJ

R/W Throughput – ForHLR & Horeka



Metadata Performance (Horeka)

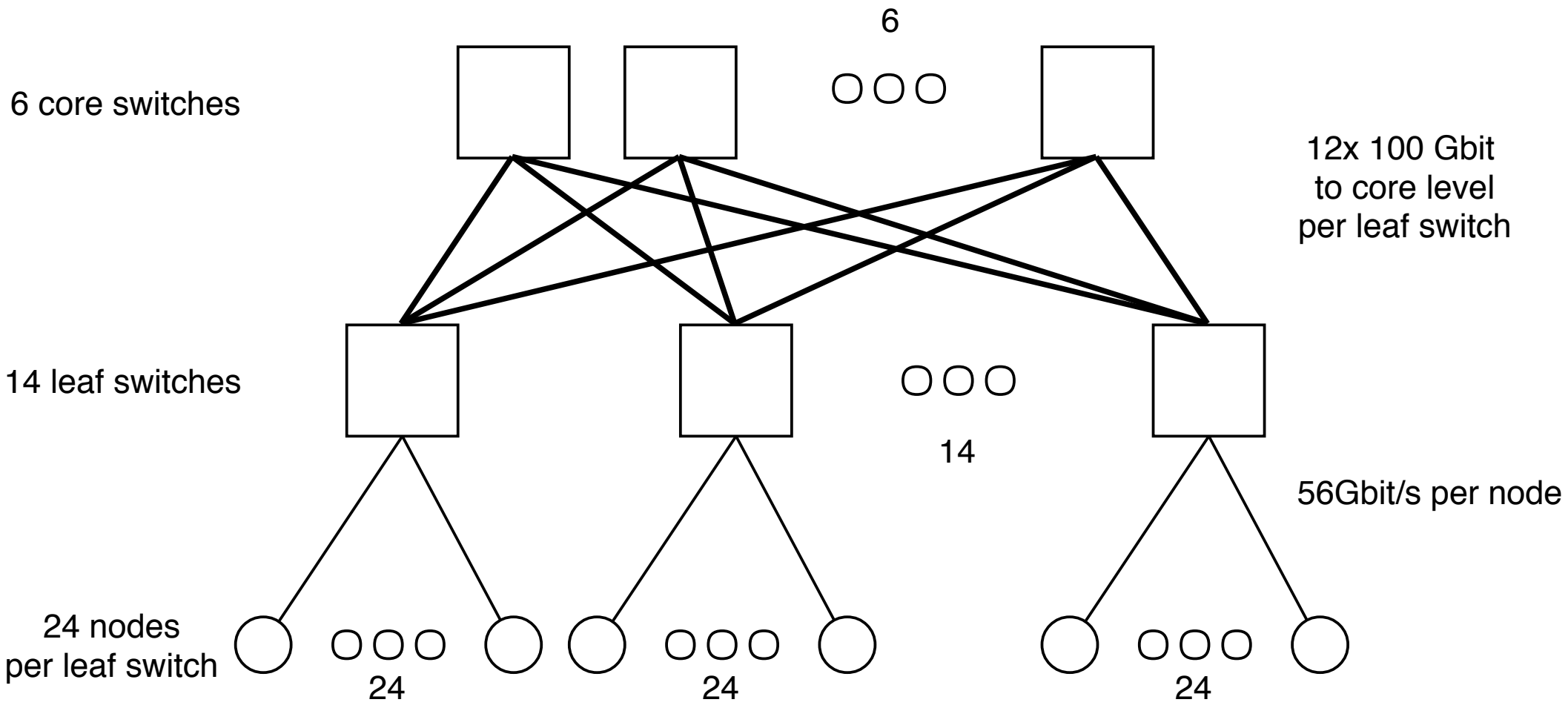


Topology awareness

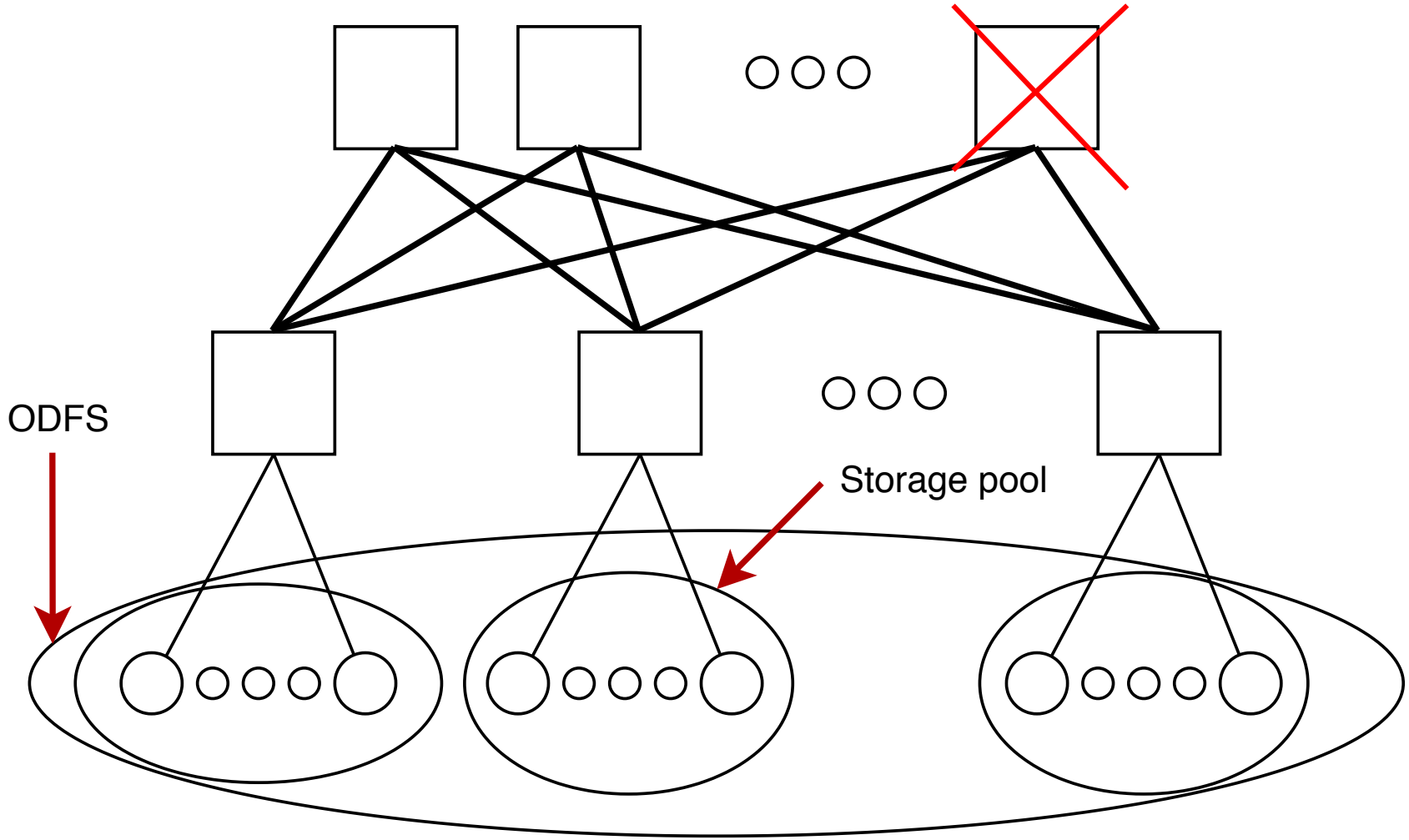
- What if we do not have a fully-non blocking interconnect?
  - Large Systems have a “weaker” interconnect (Dragonfly, Tofu, Hypercube …)
  - Cost of interconnect
- BeeGFS offers storage pools

- => Storage pools for topology awareness

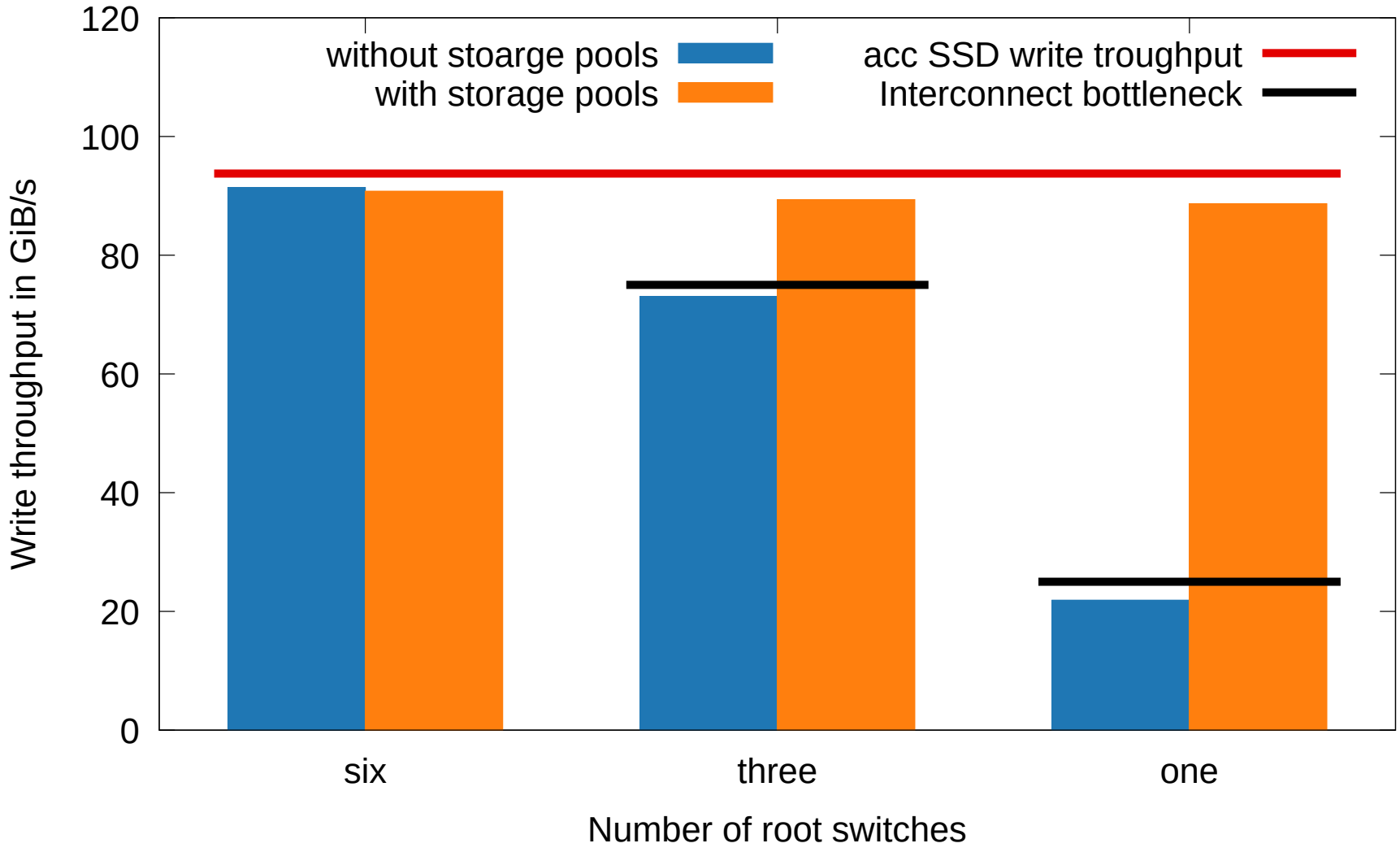
Small Island ForHLR II



Storage Pool per Leaf Switch



Comparison w/o Storage Pools



Summary – Benchmarks

- BeeGFS scales well
- Bandwidth scales almost linear
- MDS performance scales with multiple MDS
- BeeGFS has the same “weakness” like Lustre
  - MDS bottleneck
  - Working in same directory => slow performance
  - Need to choose right strip size/count
  - Care about file size / stripcount / OST capacity

- Motivation / Background
- Integration into HPC batch system
- Generic Benchmarks
- **Use Cases**
  - **OpenFOAM, NAStJA, FLAPS**
- Summary and Conclusion

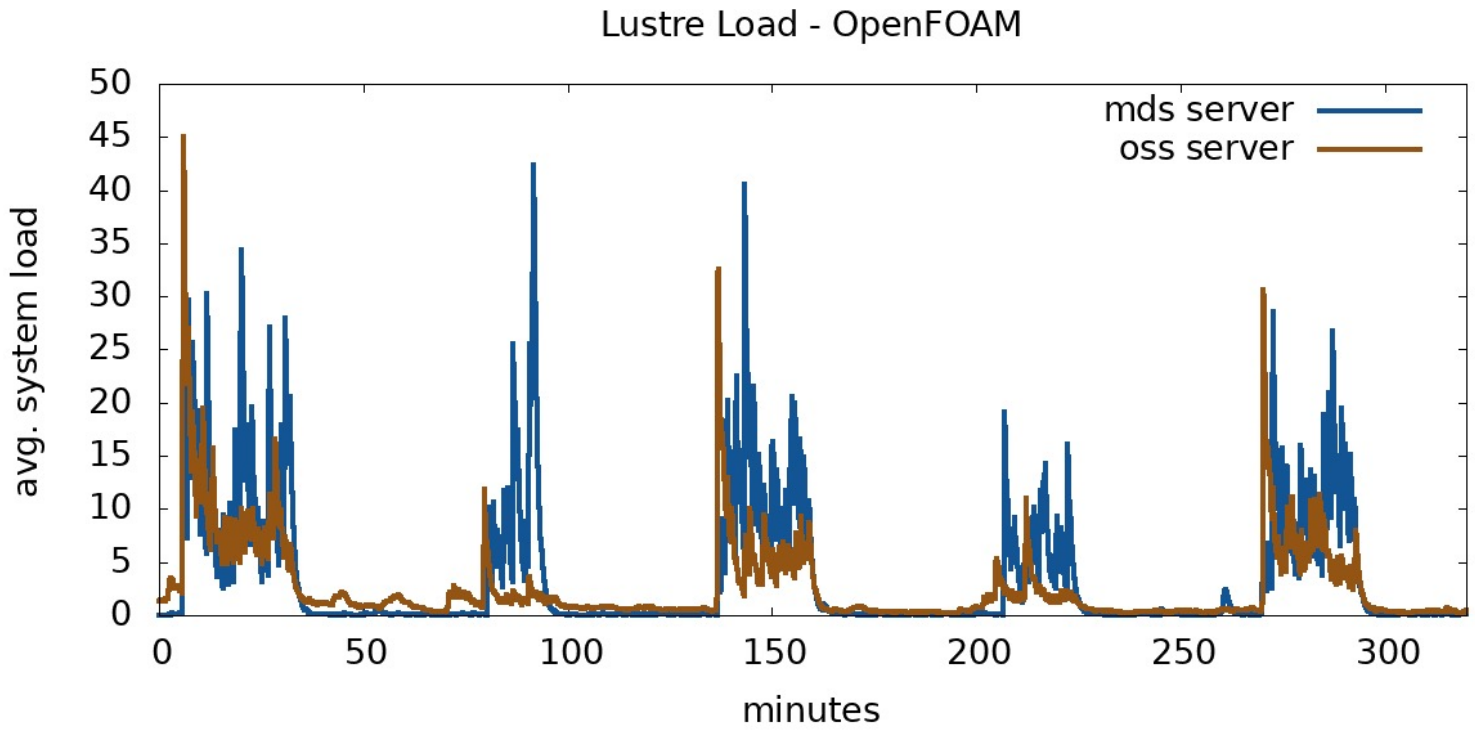
Use Cases

- OpenFoam[1]
  - 240 nodes (20 Cores)
  - ~450k files / 120 GB per snapshot
- NAStJA[2] – Stencil Code solver
  - 240 nodes (20 Cores)
  - 4800 files / 4800MB per snapshot
- FLAPS[3] - dynamic particle swarm optimization
  - ML/DL + Gromacs
  - 50 nodes
  - Heavy metadata

1) Zirwes et al. “Automated Code Generation for Maximizing Performance of Detailed Chemistry Calcu- lations in OpenFOAM”. In: High Performance Computing in Science and Engineering ’17.
2) Berghoff et al. “Massively Parallel Stencil Code Solver with Autonomous Adaptive Block Distribution”. In: IEEE Transactions on Parallel and Distributed Systems 29.10 (2018)
3) Weiel et al. “Dynamic particle swarm optimization of biomolecular simulation parameters with flexible objective functions.” Nat Mach Intell 3, https://doi.org/10.1038/s42256-021-00366-3

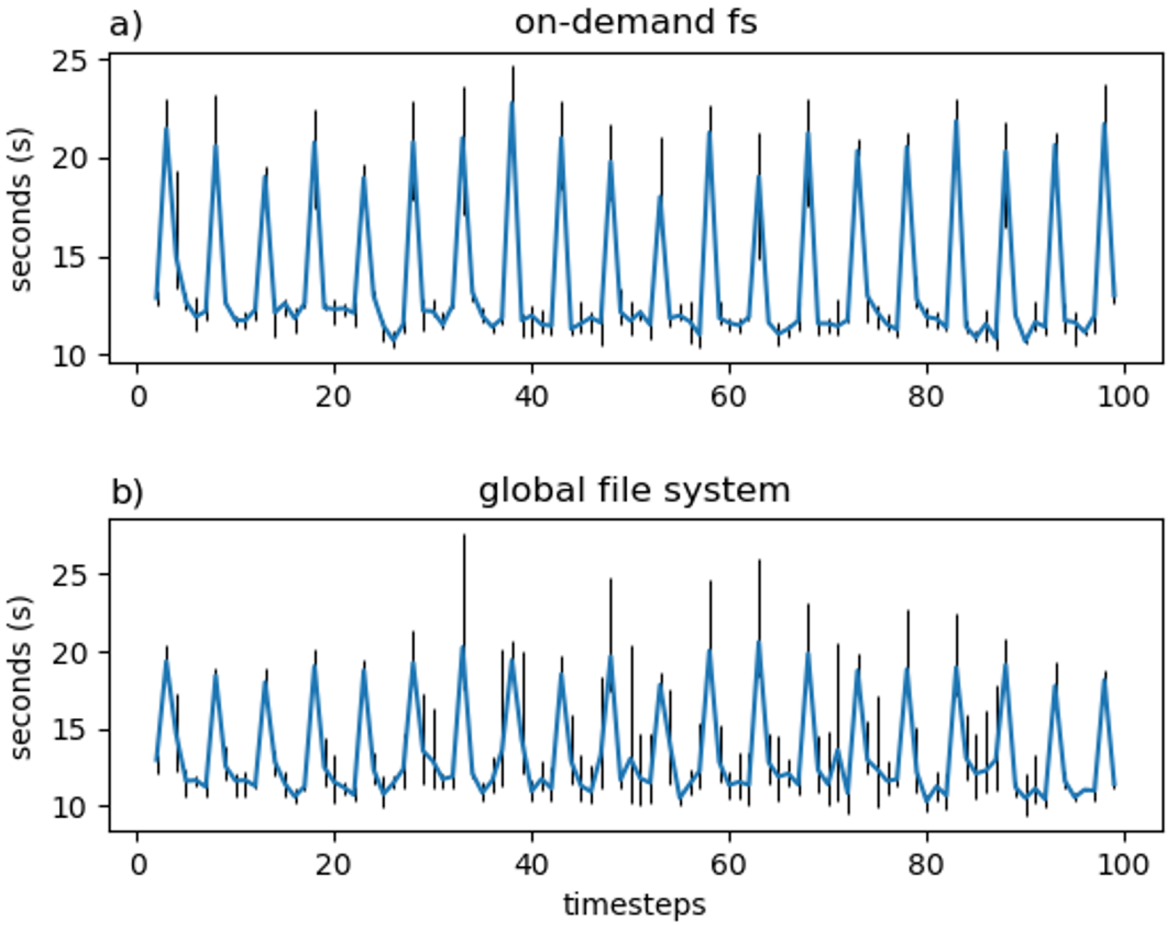
OpenFOAM – Lustre Server Unix Load

- 5 short runs
- Average unix load of MDS/OSS
- Whole system is getting “laggy”



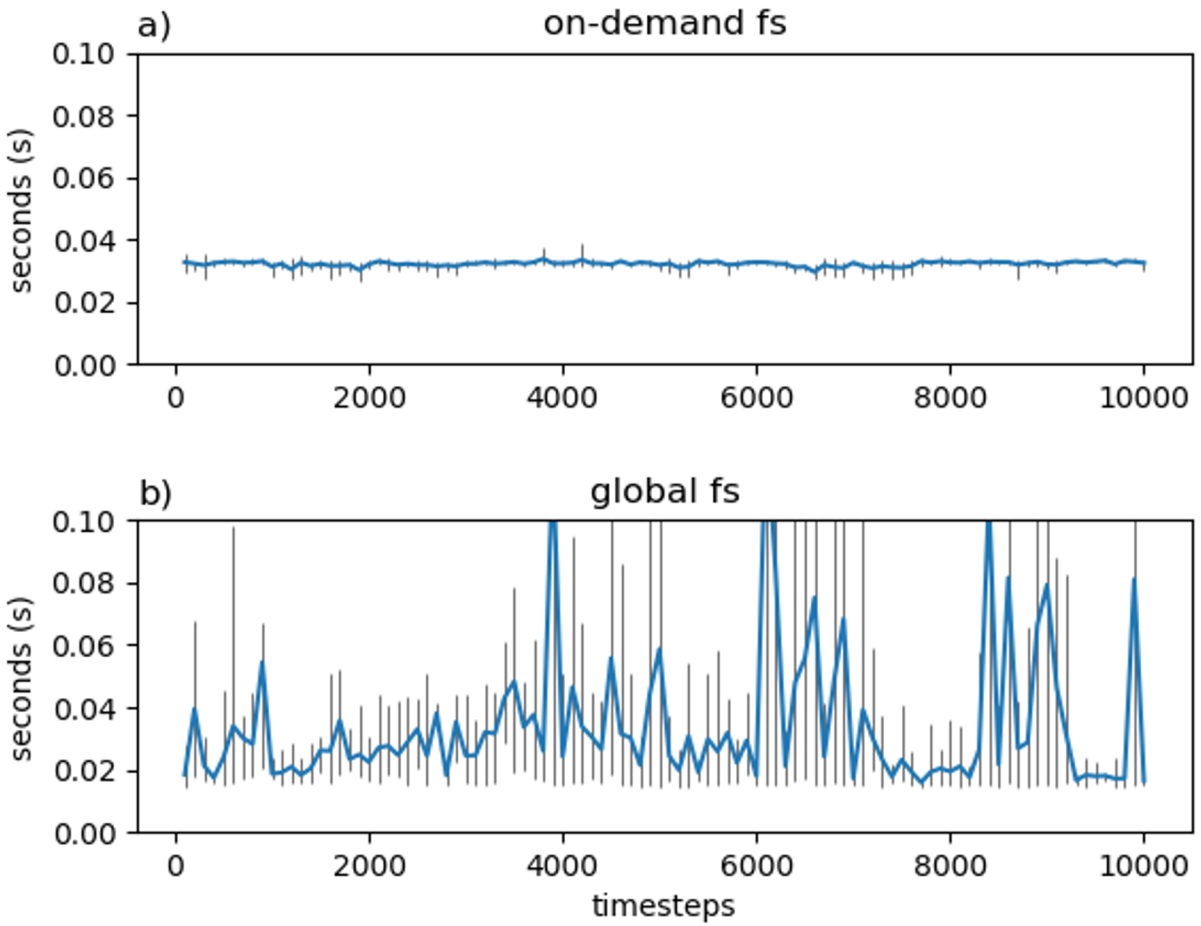
OpenFOAM – Runtime per Timestep

- Blue line average of 5 runs
- 100 timesteps ~ 30 min
- Snapshot every 5 timesteps
- Less variation when using ODFS (black bars)
- Simulation takes slightly more time when using ODFS



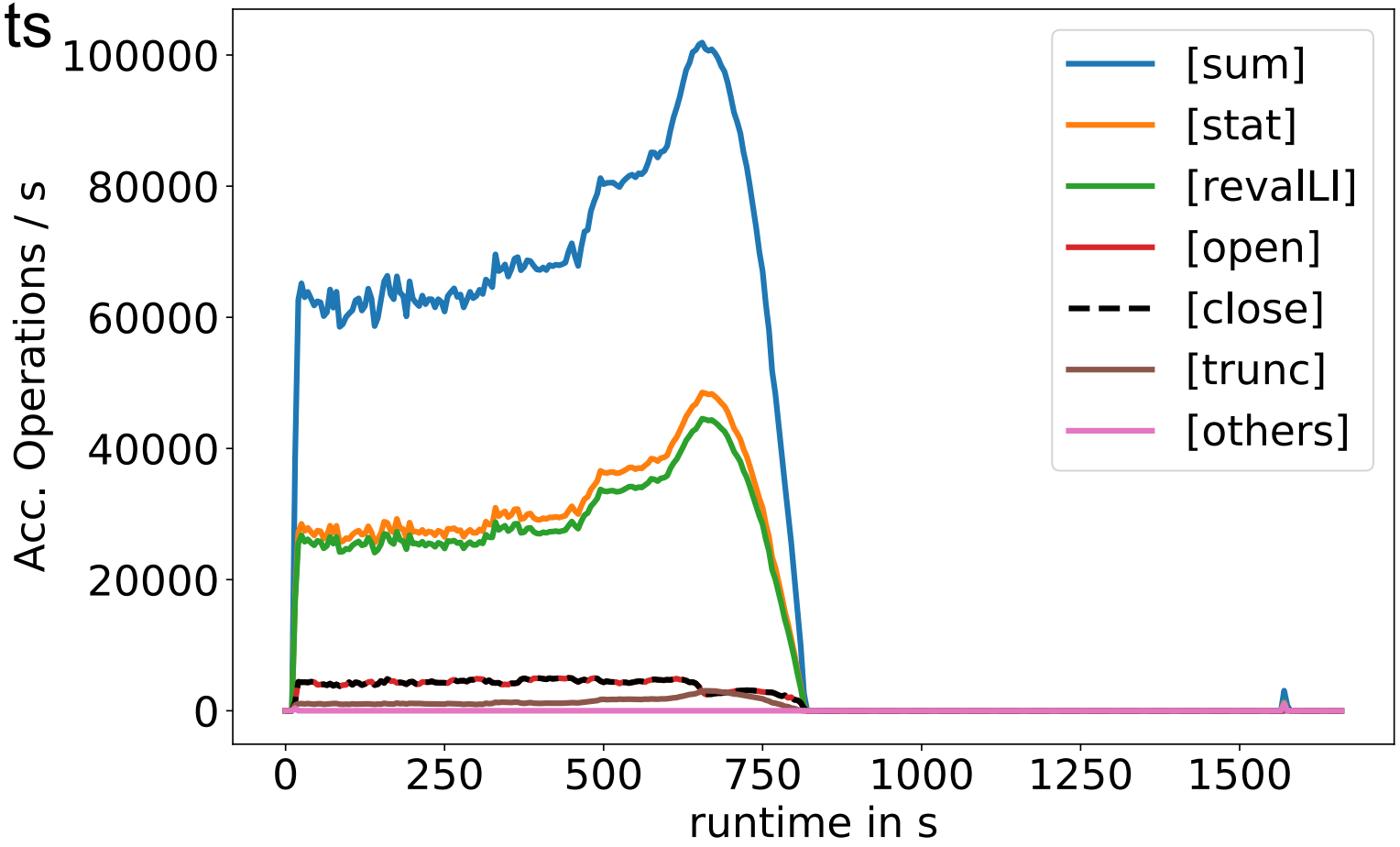
NAStJA – Runtime per Timesteps

- Snapshot every timestep
- Almost no variation when using ODFS
- Very high interference on ODFS

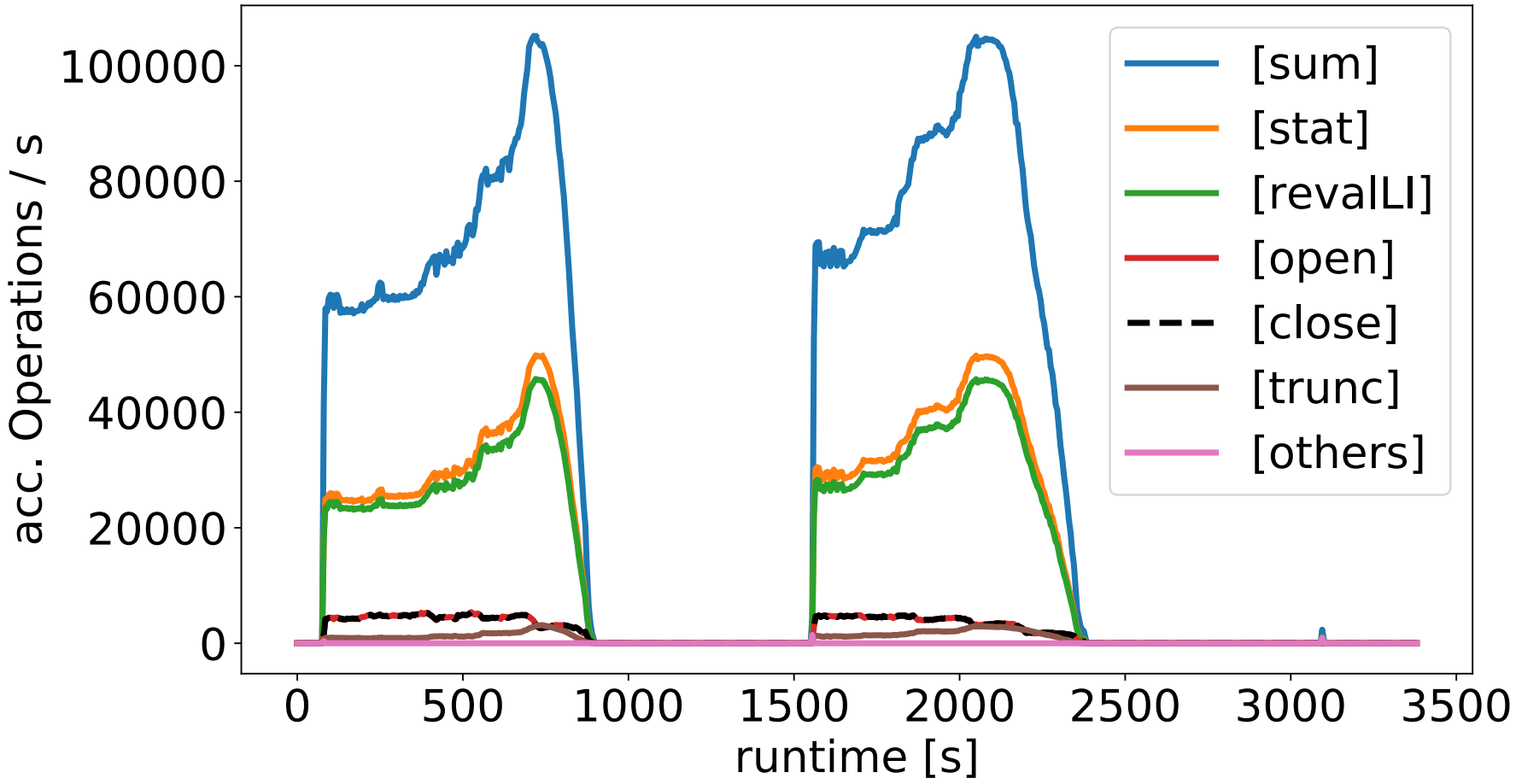


FLAPS – One Generation

- ML/DL of gromacs results
- High metadata op/s
- A lot of stat operations
- One job slowed down the whole system



FLAPS – Two Generations



Summary – Use Cases

- Effect on application hard to predict
  - Some running slower and some faster[1]
- Metadata performance more important
- Easy to use for users – no code changes needed
  - Data staging is handled by users
- Some use cases are enforced to use ODFS

1) Versick et al. “Performance gains in an ESM using parallel ad-hoc file systems” EGU 2020

- Motivation / Background
- Integration into HPC batch system
- Generic Benchmarks
- Use Cases
  - OpenFOAM, NAStJA, FLAPS
- **Summary and Conclusion**

Summary & Conclusion (1)

- ODFS (BeeGFS) is easy to integrate
- Startup time is acceptable
- SSD wear leveling should be observed
  - We did not notice any problems
- Loopback device improves performance
  - Especially with small files
- BeeGFS scales well (Metadata performance ok)
  - Multi MDS
  - Topology awareness with storage pools possible

Summary & Conclusion (2)

- Impact on application hard to predict
- Observation: Add one additional node to job
  - Use this additional node for MDS
  - mpirun –nolocal
- Observation: Metadata performance more important than bandwidth
  - Added options for multi metadata servers
- User handles staging staging
  - Concurrent data staging possible with less interference[1]

- **Advantages of ODFS outweigh any disadvantages**
  - Most important: Load reduction of the global parallel file system

1) Soysal et al. “Using On-Demand File Systems in HPC Environments” HPCS 2019 DOI: 10.1109/HPCS48598.2019.9188216

Thanks for your attention

Questions ?

Scripts available: https://github.com/mehsoy/ODFS-tools

Backup Slides