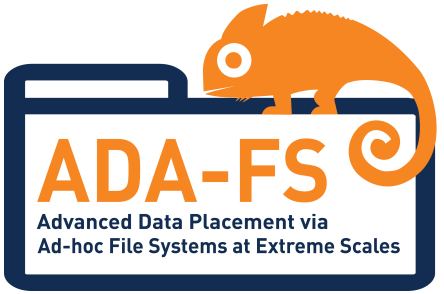
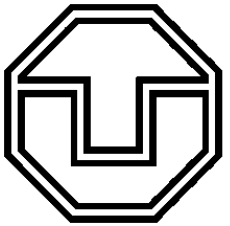


Advanced Data Placement via Ad-hoc File Systems at Extreme Scales (ADA-FS)

Mehmet Soysal, Marc-André Vef, Sebastian Oeste
Achim Streit, André Brinkmann, Michael Kluge, Wolfgang E. Nagel

JOHANNES GUTENBERG UNIVERSITÄT MAINZ



TECHNISCHE UNIVERSITÄT DRESDEN

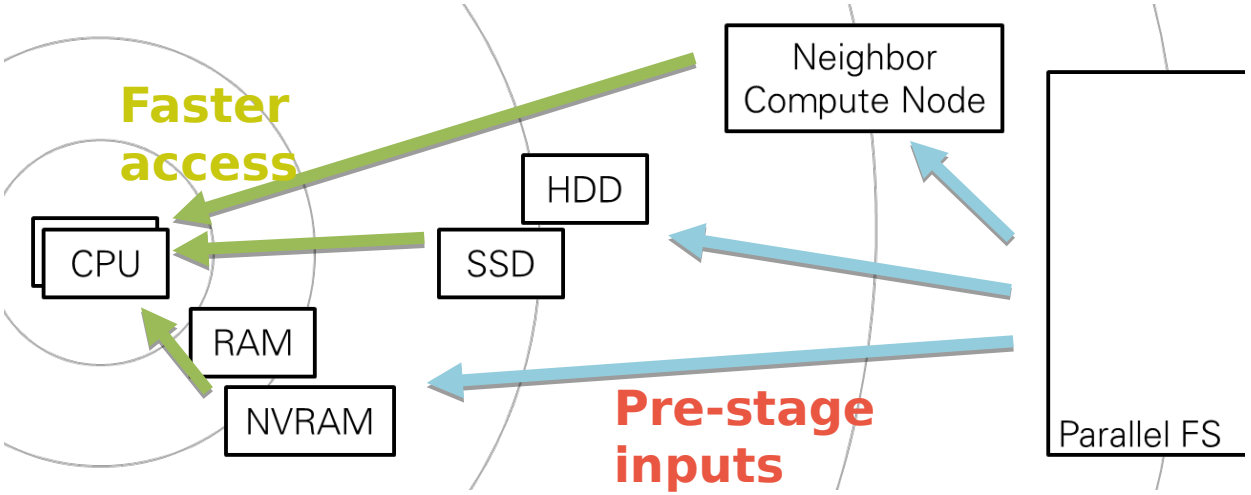
Karlsruhe Institute of Technology

The ADA-FS project

- New project in the second funding period of SPPEXA
- SPPEXA topics:
  - System software and runtime libraries
- Three project partners:
  - TU Dresden
    - Wolfgang E. Nagel (Principal Investigator), Andreas Knüpfer, Michael Kluge, Sebastian Oeste
  - Johannes Gutenberg University Mainz
    - André Brinkmann (Principal Investigator), Marc-André Vef
  - Karlsruhe Institute of Technology
    - Achim Streit (Principal Investigator), Mehmet Soysal

Project background

- The I/O subsystem is a generic bottleneck in HPC systems (bandwidth, latency)
- The shared Medium has no reliable bandwidth and IOPS
- Jobs can disrupt each other
- New storage technologies will emerge in HPC systems
  - SSD, persistent HBM (High Bandwidth Memory), NVRAM ...



View on three HPC systems

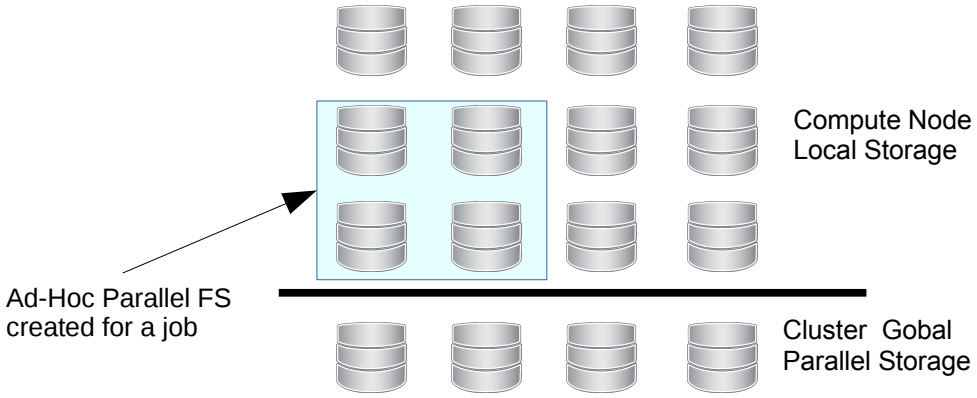
	# compute nodes N	Bisection Bandwidth	Global I/O Bandwitdh	Node Local Storage Bandwidth	Node Local Storage
Auroa @ANL	>50.000	500 Tbyte/s**	1 Tbyte/s**	>5 Gbyte/s*	NVRAM
Summit @ORNL	> 3.400	40 Tbyte/s**	2.5 Tbyte/s**	>5 Gbyte/s*	NVRAM
ForHLR 2 @KIT	1.172	6 Tbyte/s	50 Gbyte/s	0.5 Gbyte/s	SSD

* assumed values ** announced values

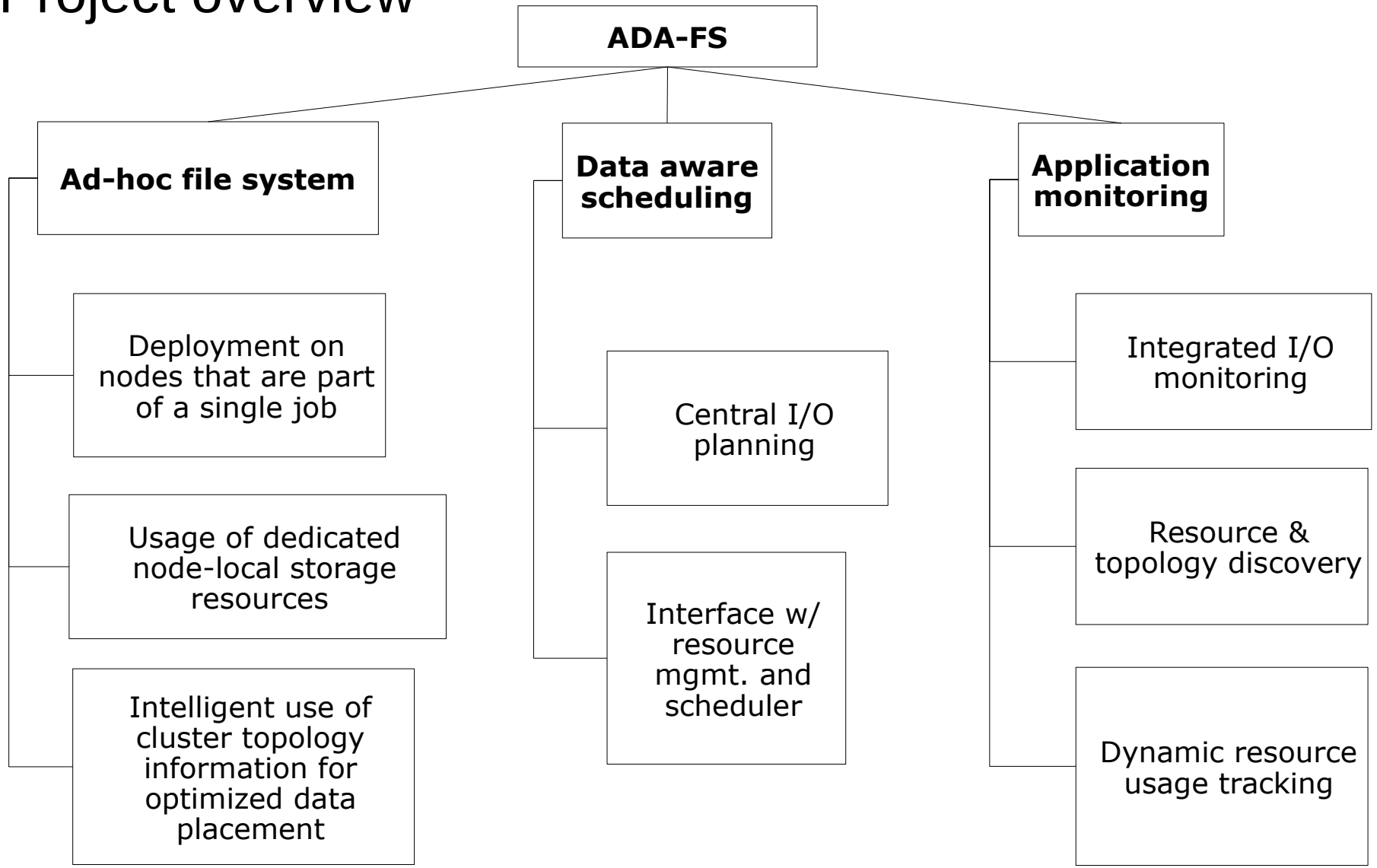
- Future systems are planned with high-speed node storage
- SSDs are already used in today's systems
- Higher bisection bandwidth within compute nodes

ADA-FS

- Proposed solution
  - Bring data closer to compute units
  - Create a private parallel file system for each job on the compute nodes
  - Tailor the file system to the application's requirements
  - Discover interconnect topology and distribute data accordingly
- Advantages
  - Dedicated bandwidth and IOPS
  - Independent to the global file system
  - Low latency due to SSD/NVRAM



Project overview



Ad-hoc file system

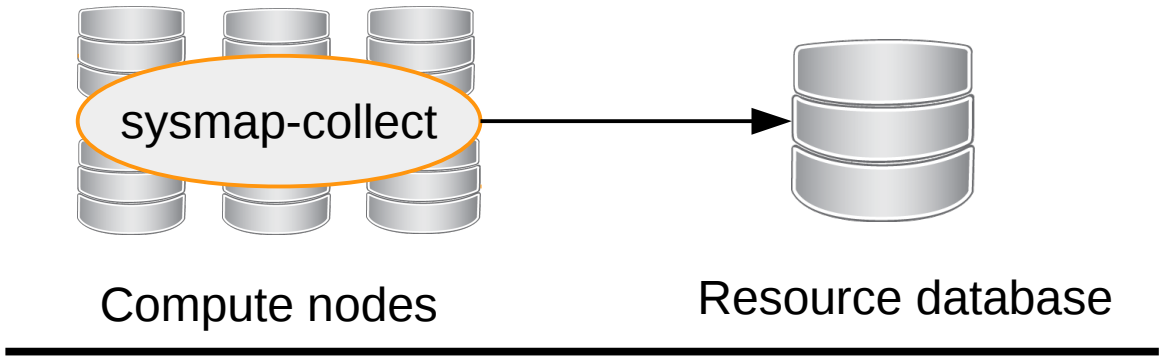
- Based on the Fuse library
- Challenges
  - Startup time
  - Metadata and data scalability
  - Fully utilize fast storage technologies (e.g., NVRAM)
- Solutions
  - Metadata scalability through key-value stores
  - Distribute data across all available disks
  - Use cluster topology information for improved data placement
  - Eventual consistency
  - Relaxed POSIX semantics
  - Optimize file system configurations for the running application

Data aware scheduling and data management

- Interaction with the existing batch environment – no replacement of existing components
- Challenges
  - Bad wall-clock prediction
  - Plan the exact time to stage the data to the ad-hoc file system
  - Data has to be managed during data staging
- Solutions
  - Use machine learning for better wall-clock prediction
  - Pre-stage data using RDMA (NVMe)
  - Manage data with “workpools” with a global identifier
  - Tight interaction with the existing scheduler and the resource manager

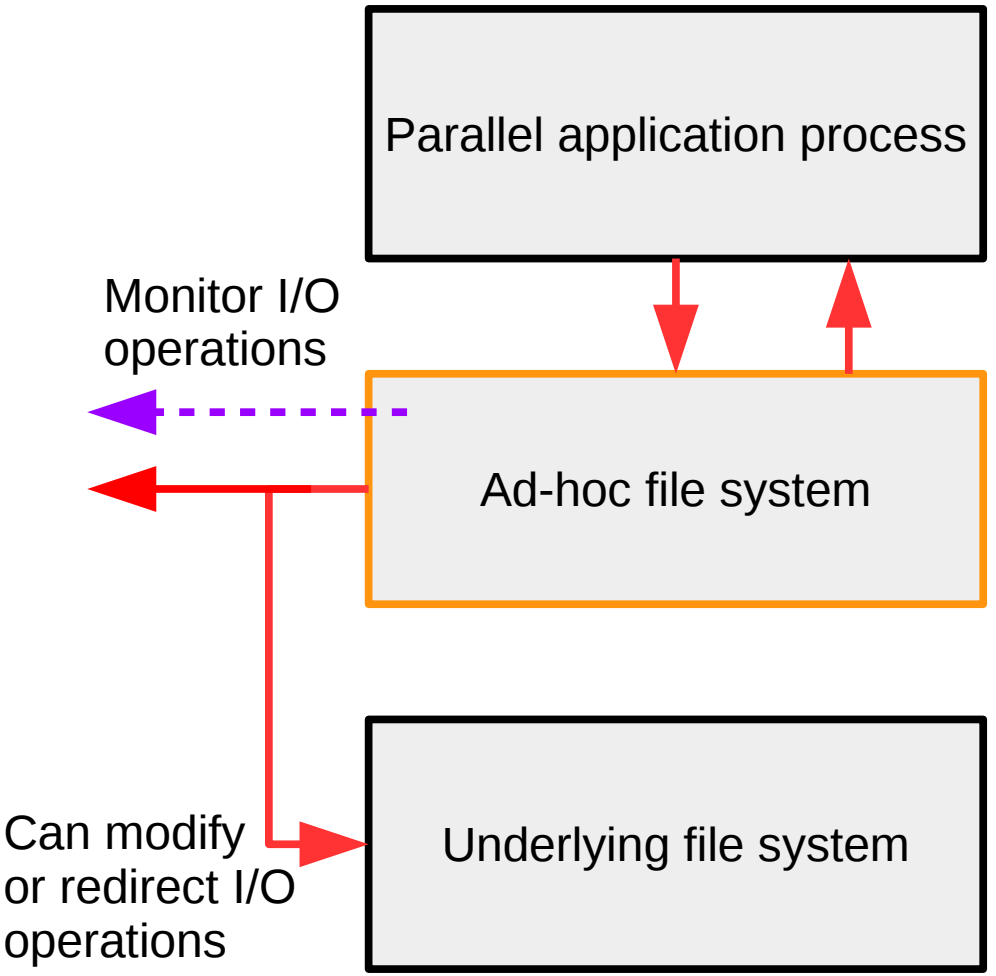
Resource and topology discovery

- Discover node-local storage types, sizes, and the reliable local speed
  - Store information in a central resource database
  - Use tools to fetch information of a number of nodes
- Expectation
  - Detailed knowledge where the job will run
  - Provide a better foundation for data-staging and scheduler decisions



Monitoring

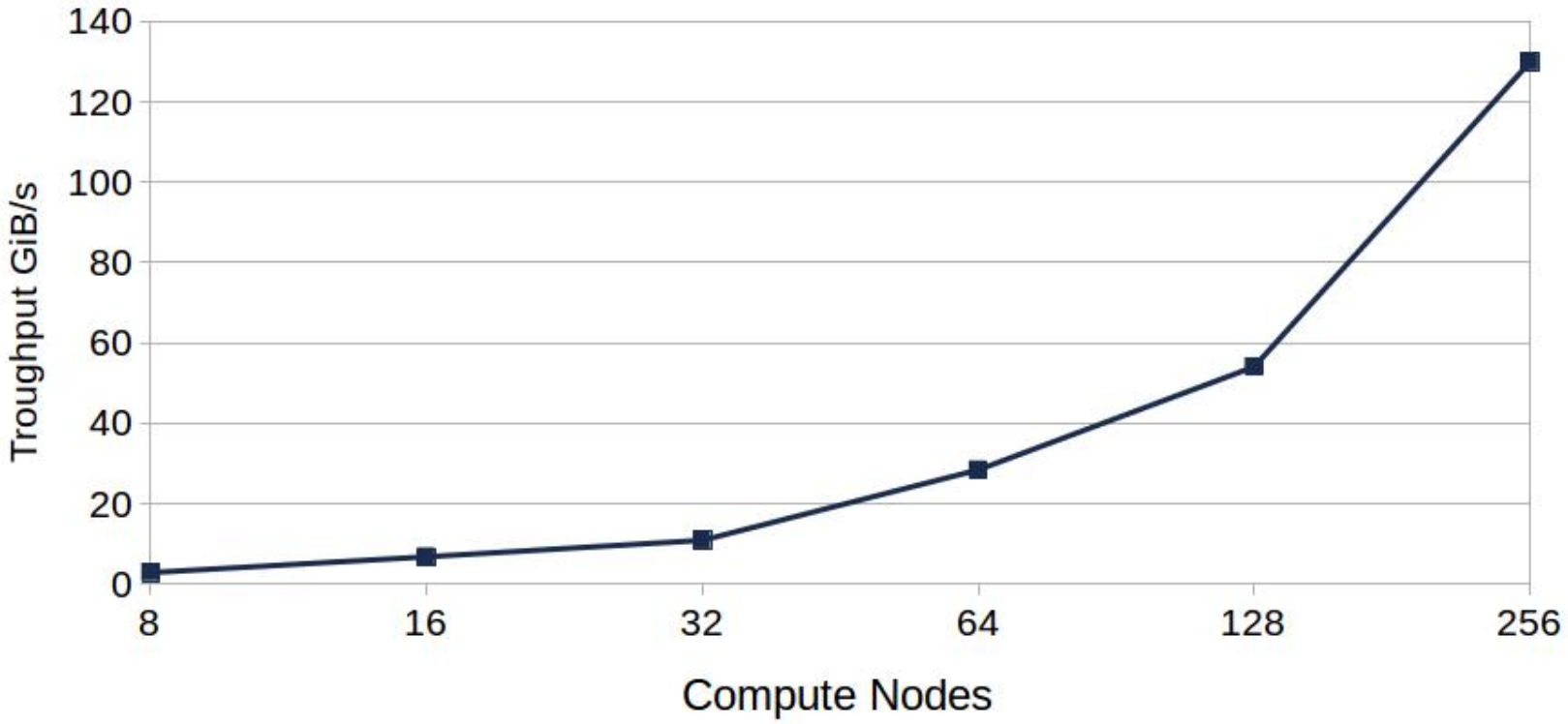
- Record I/O behavior
- Predict I/O phases
- Generalize I/O for application types
- Learn I/O characteristics
- Generate hints for the I/O planer



Initial benchmarks

- ForHLR II @ KIT
- Compute nodes with Infiniband FDR (56 Gbit/s) Fat-Tree
- Global I/O throughout 50 GByte/s
- Node local SSD (400Gb) approx. R/W 600/400MB
- BeeGFS used as private parallel file system
- 256 computes nodes used for benchmark
- First benchmark: IOZone write throughput
- Second benchmark: super_sph

IOzone write throughput

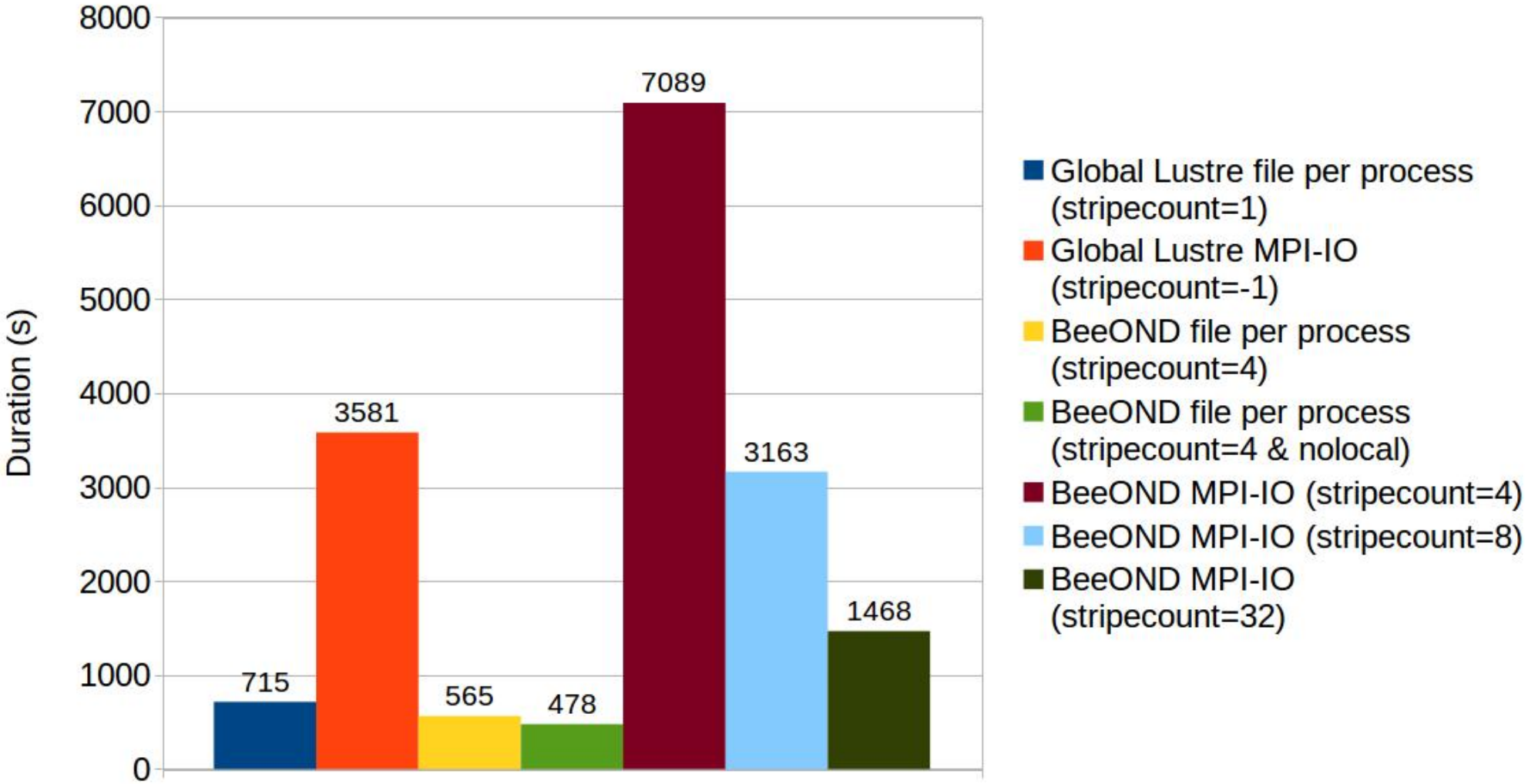


- Observation: Write throughput is limited by SSD performance

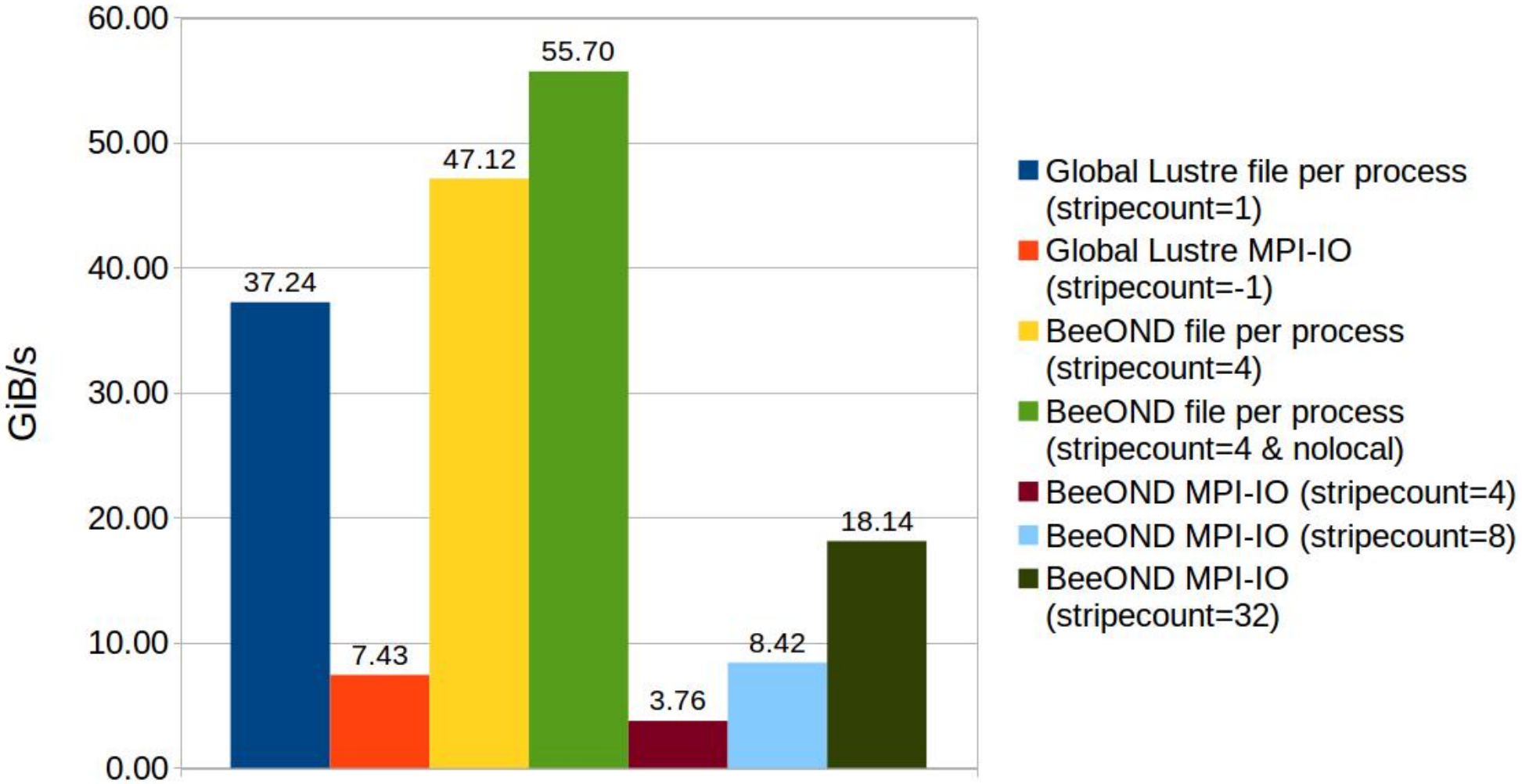
User application - super_sph

- Simulation for *Smoothed Particle Hydrodynamics*
- Developed at the “Institut für Strömungsmaschinen” @KIT
- Scales up to 15,000 Cores and 10^9 particles
- First implementation with file-per-process (HDF5) with data-gathering as post-processing
- Now writing directly to time steps using MPI-IO
- File-per-process method causes heavy load on the PFS
- MPI-IO is slower than the file-per-process method but impacts the global PFS less
  –> no post-processing required
- Benchmark
  - 10^{10} particles and 50 time steps
  - 26 TB of data

Writing runtime



Data write throughput



Conclusion of benchmarking a user application

- With good settings of the private parallel file system → good results
- Many optimizations are possible
- Changing user application is not required

But:

- Application must be profiled and analyzed to tailor the private file system
- Data need to be copied back to the global PFS
- Users should not need to worry about the file system configurations (e.g., strip count)

We are looking for more real world applications

- Applications with high IOPS or large I/O footprints
- Applications that expose high metadata loads on the parallel file system
- Please contact us:

ada-fs-all@fusionforge.zih.tu-dresden.de
Michael Kluge <michael.kluge@tu-dresden.de>
Sebastian Oeste <sebastian.oeste@tu-dresden>
Marc-André Vef <vef@uni-mainz.de>
Mehmet Soysal <mehmet.soysal@kit.edu>

Acknowledgments

- Samuel Braun from the “Institut für Strömungsmaschinen” for supplying the code to simulate the I/O part of super_sph with different methods
- DFG for funding the ADA-FS project under SPPEXA
- Steinbuch Centre for Computing (SCC) @KIT for granting access to ForHLR II

Thank you
Questions?