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

Understanding I/O Performance Behavior (UIOP) 2017

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

Agenda

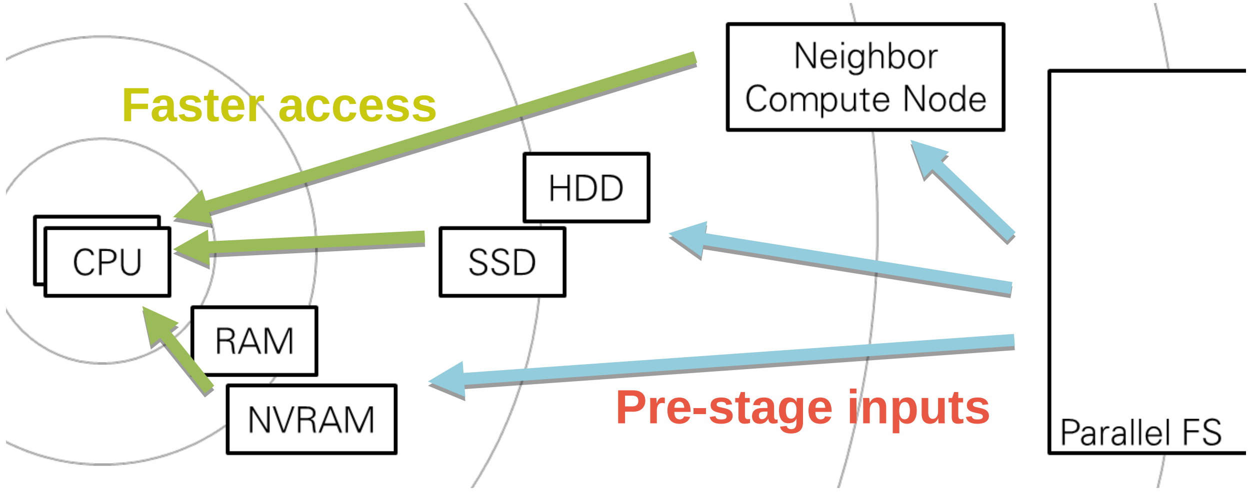
- Motivation
- ADA-FS (WIP) - Overview
- The ad-hoc file system
- Data aware scheduling
- Resource and topology discovery
- Monitoring
- Tracing in distributed file systems
- Conclusion
- Future work

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* (TUD)
    - Wolfgang E. Nagel (Principal Investigator), Andreas Knüpfer, Michael Kluge, Sebastian Oeste
  - *Johannes Gutenberg University Mainz* (JGU)
    - André Brinkmann (Principal Investigator), Marc-André Vef
  - *Karlsruhe Institute of Technology* (KIT)
    - Achim Streit (Principal Investigator), Mehmet Soysal

Motivation

- 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

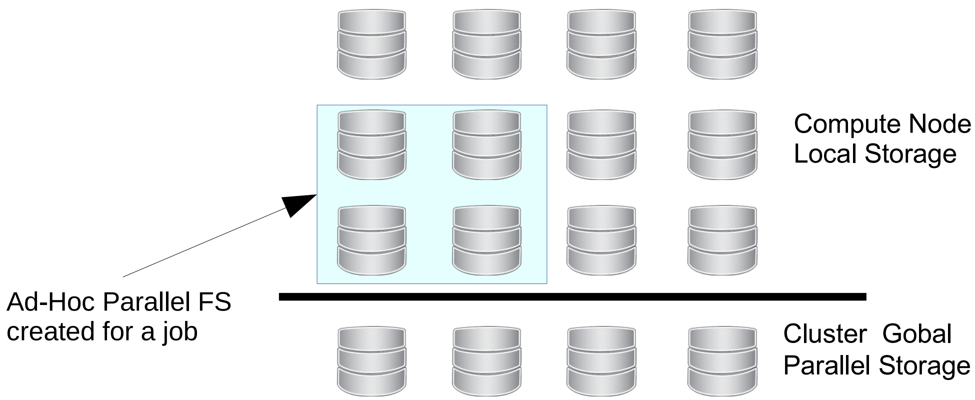
	# compute nodes N	Bisection Bandwidth	Global I/O Bandwitdh	Node Local Storage Bandwidth	Node Local Storage
Auroa @ANL	>50.000	500 Tbyte/s**	1 Tbyte/s**	>5 Gbyte/s*	NVRAM
Summit @ORNL	> 3.400	40 Tbyte/s**	2.5 Tbyte/s**	>5 Gbyte/s*	NVRAM
ForHLR 2 @KIT	1.172	6 Tbyte/s	50 Gbyte/s	0.5 Gbyte/s	SSD

* 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

ADA-FS

- **Ad-hoc file system**
  - Deployment on nodes that are part of a single job
  - Usage of dedicated node-local storage resources
  - Intelligent use of cluster topology information for optimized data placement
- **Data-aware scheduling**
  - Central I/O planning
  - Interface w/ resource mgmt. and scheduler
- **Application monitoring**
  - Integrated I/O monitoring
  - Resource & topology discovery
  - Dynamic resource usage tracking

Issues with today‘s distributed file systems

- Big players: Spectrum Scale (GPFS), Lustre, Ceph (not ready for prod.)
- Not designed for Exascale
  - E.g., Spectrum Scale was designed for Gigascale
- Heavy communication
- Intricate locking mechanisms
- Metadata is not scalable (rigid data structures)
- Applications produce too much load on file systems
- Providing POSIX I/O semantics results in such issues

Ad-hoc file system design

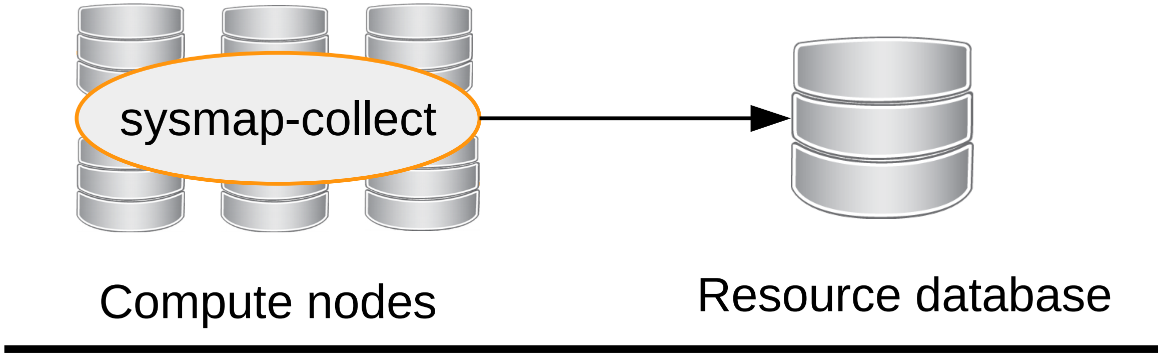
- Based on the Fuse library
- Eventual consistency
- Relaxed POSIX I/O semantics
  - Ignore metadata fields, such as mtime, atime, filesize …
  - Simplify file system protocols
  - No locking
- Scalable metadata approaches
  - Use Key-Value stores
  - No rigid data structures (e.g., directory blocks)
  - Last writer wins
- Distribute data across disks (taking locality into account)
- Use cluster topology information for improved data placement
- 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

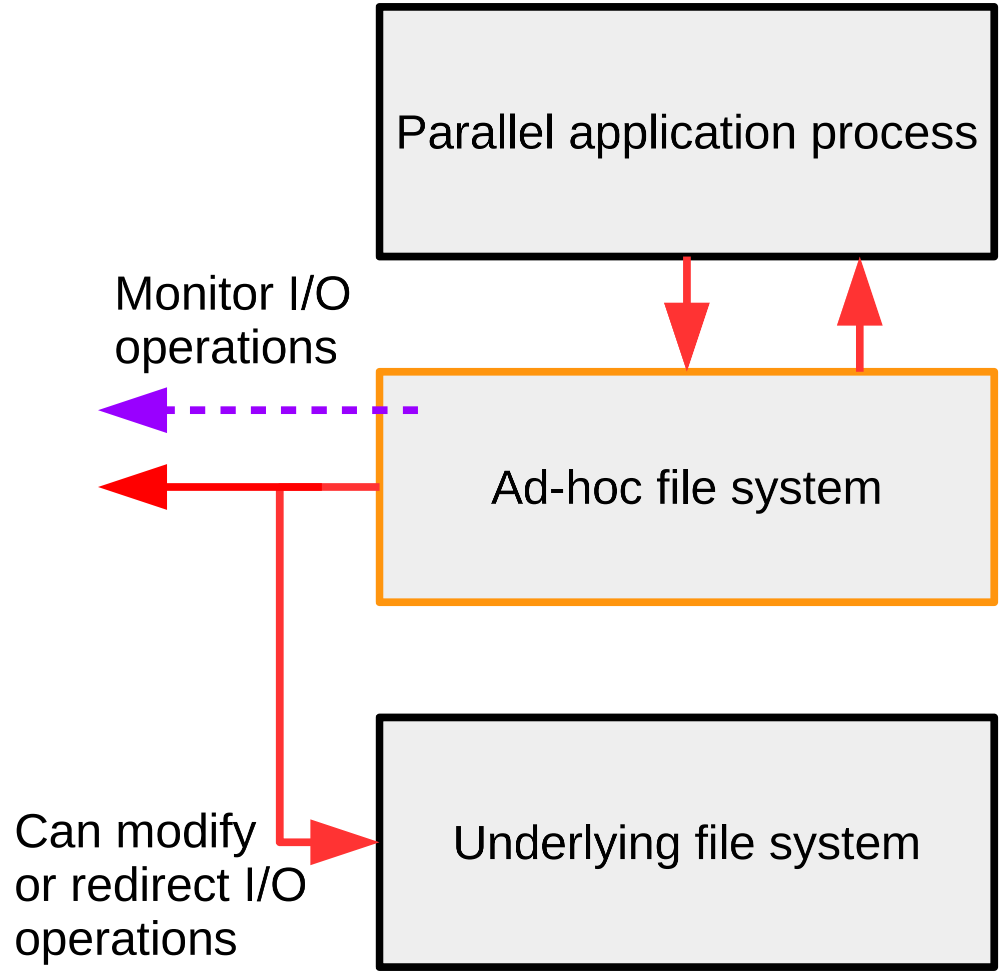
Resource and topology discovery

- 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



Towards flexible and practical tracing in distributed systems

- JGU visited IBM research for a summer internship in 2016
- Investigation of IBM Spectrum Scale's tracing framework
- Over 1 million lines of code with 20,000+ static tracepoints
  - Sets: subsystems (60) and priority (0-14)
  - Two tracing modes: blocking and overwrite
- Issues that arose after >20 development years
  - Sets of tracepoints are impractical
  - Developer tend to care only about two levels: default and not-default
  - Too many tracepoints even in the default levels
  - Either severe performance degradation (blocking) or information loss (overwrite)
  - Excess collection of byproduct traces

Solution

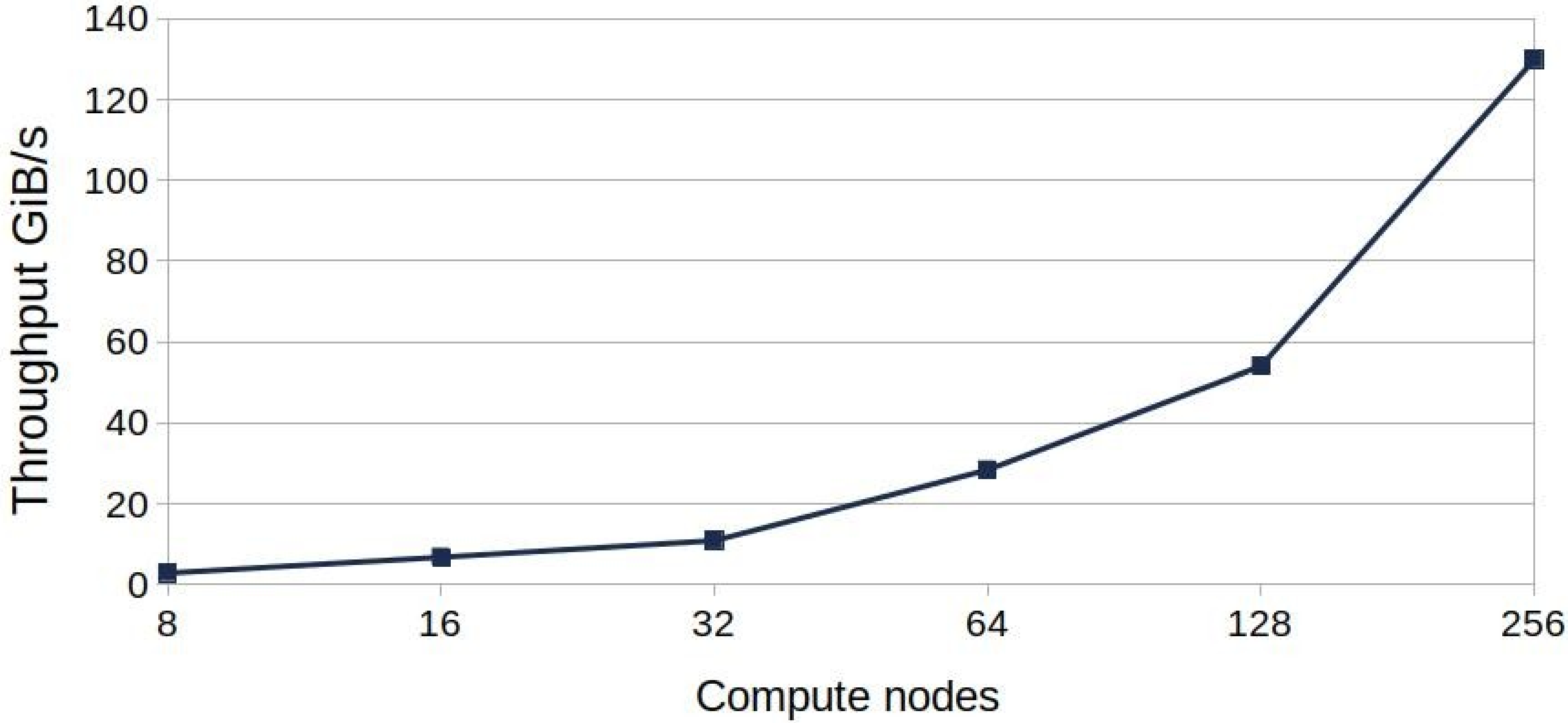
- Consider: dynamic instrumentation vs. static instrumentation
- Idea: Combine the flexibility of dynamic instrumentation with the low overhead of static instrumentation
- Evaluating a tracing prototype at IBM allowed several insights
  - Use a ring-buffer for trace collection
  - Allow fine-grained control over all tracepoints
  - A simple bitmap is sufficient
  - Allow flexible and user-defined tracepoint sets

Preliminary evaluation with BeeGFS

Experimental setup

- Hardware
  - ForHLR II @ KIT
  - Compute nodes with Infiniband FDR (56 Gbit/s) Fat-Tree
  - Global I/O throughput 50 Gbyte/s
  - Node local SSD (400Gb) approx. R/W 600/400MB
  - 256 used compute nodes
- Environment
  - BeeGFS as a job-temporal private parallel file system
  - Use the IOZone benchmark to evaluate write throughput

IOzone write throughput



- Observation: Write throughput is limited by SSD performance

Conclusion and future work

- Fully exploiting fast storage technologies will become more important
- Sparse related work on job-temporal file systems
- Write performance looks promising even with POSIX compliant systems (BeeGFS, ...)

Next:

- Application analysis across three clusters (JGU, TUD, KIT)
  - I/O phase categorization
  - Which POSIX functions are mostly used
  - Which hints can be used for improved data placement
- Analysis of the impact of Fuse and other file system configurations
- Evaluation of the performance and applicability of a minimal file system
- Investigation of efficient data staging methods

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>

- Or visit ada-fs.github.io

Thank you
Questions?