Leveraging Flexible Storage System Components for HPC Research

NHR Workshop at DKRZ

2024-05-08

Prof. Dr. Michael Kuhn

michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Background

- My group conducts research and development on parallel systems
  - High performance computing
  - Storage and file systems
  - Data reduction techniques
  - I/O interfaces
  - Programming concepts
- We also offer a variety of courses for students
  - Parallel programming, parallel storage systems etc.

Problems

- We need graduates with storage system knowledge
  - Students we teach today are the ones doing research/development tomorrow
  - Finding suitable external candidates is even harder
- Storage system research/development requires special skill set
  - Many students are not familiar with system-level topics
- It is hard to find motivated and skilled students for storage topics
  - Anecdotal: Most students are not interested in storage topics

Problems...

- Wide variety of skills are relevant for storage systems
  - Operating systems, file systems, storage devices, networking etc.
  - C/C++/Rust, kernel programming, system administration etc.
- Storage topics require extensive training periods
  - Hard to fit into smaller courses or theses
  - Setup can take up significant portion of available time
- Students have a broad understanding after attending our courses
  - Full-fledges storage systems are more complex, however
- Many study courses do not include enough system topics
  - Number of systems groups seems to be shrinking

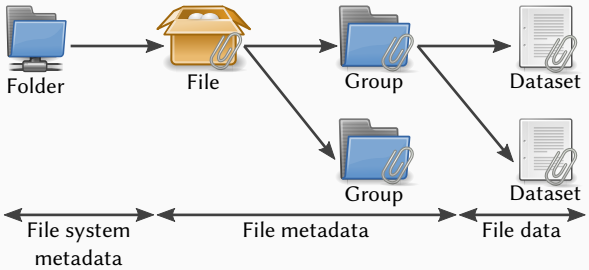
Outline

Self-Describing Data Formats

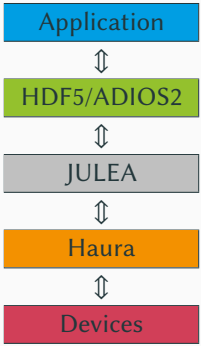
- Data is typically stored in parallel distributed file systems
  - Example: Summit (ORNL) with a capacity of 250 PB and a throughput of 2.5 TB/s
- Self-describing data formats (SDDFs) are widely used to exchange data
  - Data can be accessed and interpreted without prior knowledge

Self-Describing Data Formats

- Data is typically stored in parallel distributed file systems
  - Example: Summit (ORNL) with a capacity of 250 PB and a throughput of 2.5 TB/s
- Self-describing data formats (SDDFs) are widely used to exchange data
  - Data can be accessed and interpreted without prior knowledge

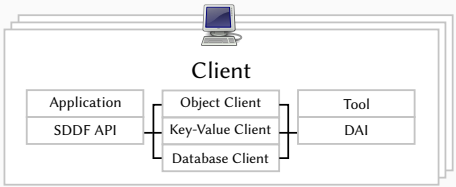


- JULEA is a flexible storage framework
  - Contains necessary building blocks for storage systems
  - Open source to be used in research and teaching[1]
- Facilitates rapid development and evaluation of prototypes
  - File systems are traditionally part of the operating system
  - Increased complexity and fragility of operating system approaches
- Support for a wide range of I/O interfaces
  - Objects, key-value, databases, HDF5, ADIOS2 etc.
  - Can access storage devices directly via Haura[2]



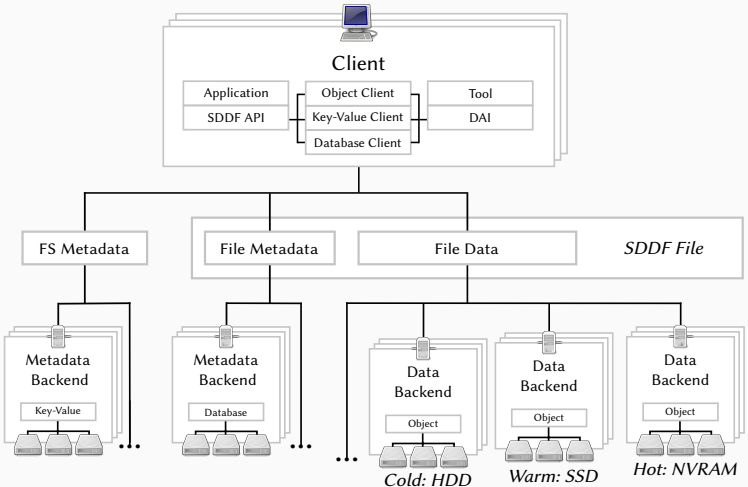
[1] https://github.com/parcio/julea
[2] https://github.com/parcio/haura

- We are using our own storage framework for most research
  - Easier for students to work with, easier for us to help them
  - Additional analyses with BeeGFS, Ceph, Lustre, OrangeFS etc.
- Why are we using our own storage framework instead of existing ones?
  - GPFS: Not open source
  - Lustre: Used in the past, too complex
  - BeeGFS: Problematic license (except for client, not really open source)
  - OrangeFS: Used in the past, not very flexible back then
  - DAOS: Back on our radar now that it does not require NVRAM anymore
  - JULEA predates newer approaches (first commit in 2010)

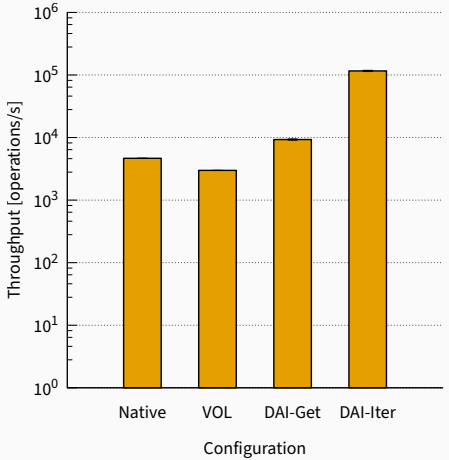


- Rethink the architecture of storage systems
  - Couple storage system and SDDFs
- Applications continue using existing interfaces
  - Fully transparent and backwards compatible
  - Native data formats can be exported to exchange data
- Data Analysis Interface (DAI) for efficient processing
  - No unified way to connect metadata and data across files
  - Example: “Average temperature over the last 12 months for all experiments”
- JULEA provides infrastructure and low-level interfaces

- CoSEMoS: 2019–2023
  - Generic approach for arbitrary data formats
  - Improve performance and data management
- Storage system understands structure of data formats
  - Optimized mapping and efficient access
- I/O requirements determine mapping to backends
  - Hot data on fast media etc.

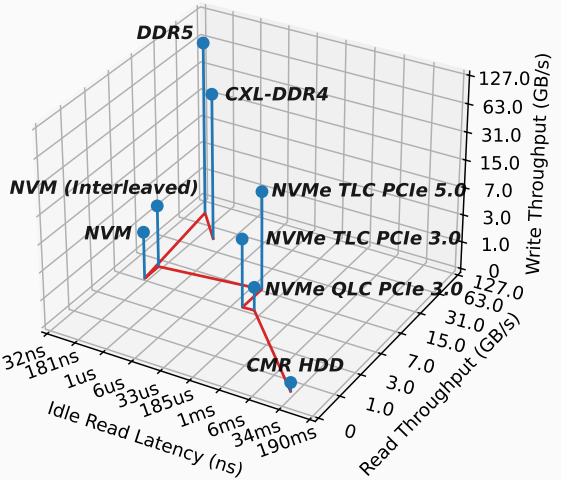


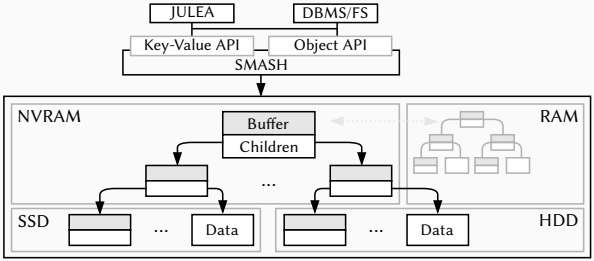
- Data Analysis Interface improves performance
  - Use case: Reading attributes from HDF5 file
- Native HDF5 and JULEA plugin
  - Inefficient, all accesses have to pass I/O stack
- DAI-Get (individual reads via DAI)
  - Faster by a factor of two
- DAI-Iterate (efficient query via DAI)
  - Faster by a factor of 25
- ADIOS2 version faster by a factor of up to 60,000



Outline

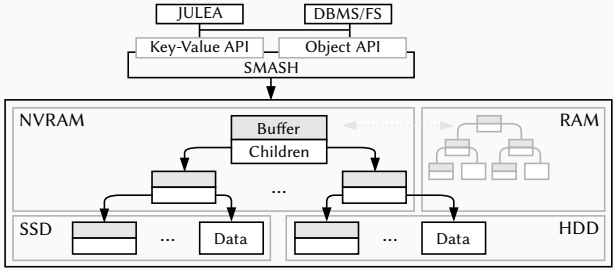
- Many factors shape limitations
  - Latency, bandwidth, capacity, granularity, cost, etc.
  - Storage accesses during computation
- Diverse options for placing data
  - High-bandwidth memory has very limited capacity
  - Data needs to be stored on multiple media
  - Not a strict hierarchy

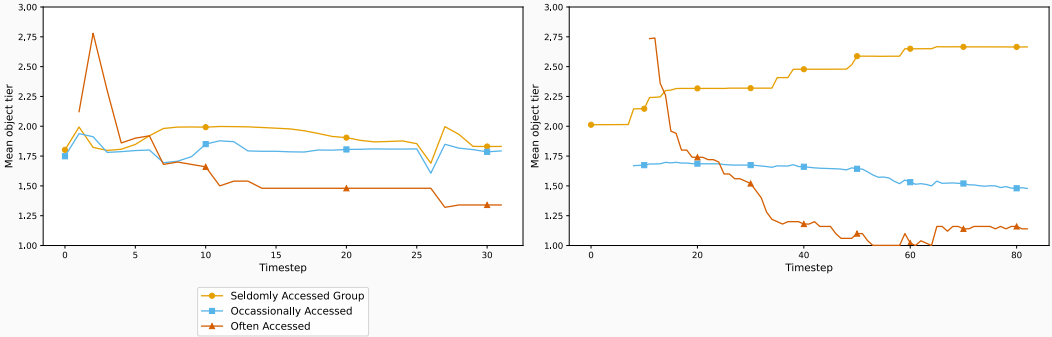




- Haura: Data store for scientific applications
- Distribute tree over a variety of storage and memory technologies
  - Exploit their unique characteristics in the process
  - Different data structures depending on storage technology
- Speed up workflows by optimizing for different data access needs

- SMASH: 2022–2025
  - Heterogeneous storage landscape
  - Traditional storage devices and non-volatile memory technologies
- Object and key-value store
  - Data placement and migration
  - Eliminating volatile caches
  - Data reduction techniques
- HPC and DBMS use cases
  - HPC workflows via JULEA
  - DBMS via SMASH interfaces





- Least frequently used policy (left): More time spent performing migrations
- Reinforcement learning policy (right): Can classify data more accurately

Outline

Ideas

- JULEA is not meant as competition for production file systems
  - Relatively easy to understand and therefore to get working with
- Offer a playground for new and interesting technologies
  - Rust is pretty popular for systems development (and with students)
  - Python bindings to make it easier to work with
  - Ready-made containers to keep setup overhead low

Ideas...

- JULEA and Haura work on their own
  - We are currently working on integration tests for the full stack
  - Next step will be running real-world applications on top of it
- We still need to streamline the setup process
  - Dependencies are installed using Spack
  - Dev containers should make setup easier and faster
- Better telemetry is needed to understand performance behavior
  - Existing tracing frameworks mostly capture application behavior

Outline

Summary

- JULEA and its clients cover high-level I/O interfaces for applications
- Haura covers efficient low-level storage and tiering
- Modular approach makes it easy to prototype new ideas
- Do you know a student interested in doing a PhD in parallel systems? ☺

References

[Duwe and Kuhn, 2021] Duwe, K. and Kuhn, M. (2021). **Dissecting Self-Describing Data Formats to Enable Advanced Querying of File Metadata.** In Wassermann, B., Malka, M., Chidambaram, V., and Raz, D., editors, *SYSTOR '21: The 14th ACM International Systems and Storage Conference, Haifa, Israel, June 14-16, 2021*, pages 12:1–12:7. ACM.

[Kuhn and Duwe, 2020] Kuhn, M. and Duwe, K. (2020). **Coupling Storage Systems and Self-Describing Data Formats for Global Metadata Management.** In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1224–1230.

[Wünsche et al., 2023] Wünsche, J., Karim, S., Kuhn, M., Broneske, D., and Saake, G. (2023). **Intelligent Data Migration Policies in a Write-Optimized Copy-on-Write Tiered Storage Stack.** In Acquaviva, J., Ibrahim, S., and Byna, S., editors, *Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems, CHEOPS 2023, Rome, Italy, 8 May 2023*, pages 17–26. ACM.