Programming Models and Engines for Computational Storage

1 Contact Information and Supervisory Team

Main Contact	Jakob Luettgau (SRP, Inria) jakob.luettgau@inria.fr https://jakobluettgau.com/
Laboratory	Institut national de recherche en sciences et technologies du numérique (Inria) Inria Centre at Rennes University, France
Research Group	KerData (Scalable Storage for Clouds and Beyond; leader: Gabriel Antoniu) https://www.inria.fr/en/kerdata https://team.inria.fr/kerdata/
Supervisory Team	Jakob Luettgau, PhD (Inria, France), KERDATA Marcello Traiola, PhD (Inria, France), TARAN

2 Context and Overview

The volume of globally produced data is projected to reach more than 180 zettabytes by 2025 [Dat]. This poses challenges to data-driven organizations and research that routinely produce and access large volumes of data for analysis, search, and increasingly machine learning applications. With current computer architectures this means routinely transferring large volumes between distributed storage systems and compute systems spanning edge, cloud and supercomputing facilities.

Unfortunately, data movements between storage and compute resources are a common bottleneck, often limiting the performance and driving the cost of executing data-driven applications. This cost primarily stems from the large amounts of network, storage and compute infrastructure as well as the energy needed to operate it. As a result data and compute infrastructure today has a notable environmental footprint: Data centers are currently accounting for about 3% of of the worlds total electricity usage [DOE24] and are projected to double by 2026 [Ele24].

To address both the performance and the energy challenges, **computational storage architectures** are being researched. These architectures combine storage and low-power computing capabilities: instead of moving the data, they allow moving the computational tasks. This allows for reducing the load on the network [LT21, SJP24] and freeing CPUs in the host system to perform other useful work [ZMRA21]. Computational storage also allows processing the data in parallel which can be much faster and more energy efficient [ZMRA21, GL21].

Research on computational storage is concerned with developing suitable programming models and designing algorithms and hardware acceleration to offer high performance while reducing the computations' energy footprint. Therefore, it becomes essential to identify the most common data—and energy-intensive tasks in applications that can be computed directly in the computational storage, thus minimizing the data movement across nodes. A hybrid approach is widely anticipated: besides computational storage functions for common tasks — such as data aggregation and reduction, compression, duplication, and erasure coding, or search through regular expressions — computational storage devices may provide re-configurable acceleration through FPGAs. This heterogeneity necessitates taking a workflow perspective to allow the co-design of programming models, runtime systems, execution engines, and suitable hardware acceleration to optimally support a particular application.

3 Internship objectives and approach

This project aims to investigate methods to offload data-intensive processing tasks to computational storage systems and devices.

1. As a first step, the student shall become familiar with the programming interfaces and data formats used to interact with computational storage devices and self-describing data formats.

2. This will allow the student to analyze and explore performance optimization and hardware acceleration of different workloads.

Real workloads in the context of numerical weather prediction and climate simulation, high-energy physics, and machine learning will be studied to identify candidates for task that can be offloaded to computational storage engines. We will will evaluate and characterize these candidates when executed in containers runtimes (e.g., Podman, Apptainer) or in hardware accelerated environments (e.g., based on RISC-V using Chipyard and Verilator).

The student will learn state-of-the-art tools and standards, such as:

- Self-Describing Data Formats (HDF5[HDF19], NetCDF[Uni19]) and numerical data processing libraries (Numpy, Pandas[pdt20], Dask[Roc15])

- Storage Networking Industry Association (SNIA) Computational Storage API [Com23]

- Podman [eaHWB+18, Con24] and Apptainer [App, KcB+21] for Container-based Computational Storage Engines

- Chipyard [ABG+20] and Verilator[SWGe24, Ver] for low-level accelerated Computational Storage Functions

We will build upon previous work and active research of the members in the supervisory team and international collaborators in the USA and Germany. As such this work will be complementary to ongoing collaborations with Argonne National Laboratory, the German Climate Computing Center (DKRZ) and the University of Magdeburg (OVGU). DKRZ and Argonne National Laboratory will provide the workloads.

Required skills

- Good knowledge/understanding of computer architectures and embedded systems

- Strong programming knowledge (Python, C/C++)

- HW design: Hardware synthesis flow, C++ code for HW (basic understanding of High Level Synthesis)

- Basic familiarity with container environments (Docker, Podman, ...)

- Knowledge of Design Space Exploration approaches is a plus

Languages: Proficiency in written English is required; fluency in spoken English is required.
Relational skills: The candidate will work in a research team, where regular meetings will be set up. The candidate has to be able to present the progress of their work in a clear and detailed manner.
Other values appreciated: Open-mindedness, strong integration skills, and team spirit.

Most importantly, we seek highly motivated candidates.

References

[ABG+20] Alon Amid, David Biancolin, Abraham Gonzalez, Daniel Grubb, Sagar Karandikar, Harrison Liew, Albert Magyar, Howard Mao, Albert Ou, Nathan Pemberton, Paul Rigge, Colin Schmidt, John Wright, Jerry Zhao, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. Chipyard: Integrated design, simulation, and implementation framework for custom SoCs. *IEEE Micro*, 40(4):10–21, 2020.

[App] Apptainer - Portable, Reproducible Containers. https://apptainer.org/.

[Com23] Computational Storage API v1.0, October 2023.

[Con24] Containers/podman. Containers, September 2024.

[Dat] Data growth worldwide 2010-2025. https://www.statista.com/statistics/871513/worldwide-data-created/.

[DOE24] DOE: Data Centers and Servers. https://www.energy.gov/eere/buildings/data-centers-and-servers, 2024.

[eaHWB+18] Matt et. al Heon, Dan Walsh, Brent Baude, Urvashi Mohnani, Ashley Cui, Tom Sweeney, Giuseppe Scrivano, Chris Evich, Valentin Rothberg, Miloslav Trmač, Jhon Honce, Qi Wang, Lokesh Mandvekar, Adrian Reber, Eduardo Santiago, Sascha Grunert, Nalin Dahyabhai, Anders Bjorklund, Kunal Kushwaha, Sujil Ashwin Sha, Yiqiao Pu, Zhangguanzhang, Matej Vasek, and Podman Community. Podman - : A tool for managing OCI containers and pods. Zenodo, January 2018.

[Ele24] Electricity 2024 - Analysis and Forecast to 2026, 2024.

[GL21] Tianhan Gao and Wei Lu. Machine learning toward advanced energy storage devices and systems. *iScience*, 24(1):101936, January 2021.

[HDF19] HDF Group. HDF5: Hierarchical Data Format, 2019.

[KcB+21] Gregory M. Kurtzer, cclerget, Michael Bauer, Ian Kaneshiro, David Trudgian, and David Godlove. Hpcng/singularity: Singularity 3.7.3. Zenodo, April 2021.

[LT21] Corne Lukken and Animesh Trivedi. Past, Present and Future of Computational Storage: A Survey, December 2021.

[pdt20] The pandas development team. Pandas-dev/pandas: Pandas. Zenodo, February 2020.

[Roc15] Matthew Rocklin. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In *Python in Science Conference*, pages 126–132, Austin, Texas, 2015.

[SJP24] Sushama Annaso Shirke, Naveenkumar Jayakumar, and Suhas Patil. Design and performance analysis of modern computational storage devices: A systematic review. *Expert Systems with Applications*, 250:123570, September 2024.

[SWGe24] Wilson Snyder, Paul Wasson, Duane Galbi, and et al. Verilator, September 2024.

[Uni19] Unidata. NetCDF: Network Common Data Form, 2019.

[Ver] Veripool. https://www.veripool.org/verilator/.

[ZMRA21] Zichen Zhu, Ju Hyoung Mun, Aneesh Raman, and Manos Athanassoulis. Reducing Bloom Filter CPU Overhead in LSM-Trees on Modern Storage Devices. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)*, pages 1–10, Virtual Event China, June 2021. ACM.