PER3S WORKSHOP, PARIS

Argonne NATIONAL LABORATORY

TEN YEARS OF MOCHI DATA SERVICES FOR HPC: A RETROSPECTIVE

MATTHIEU DORIER

Mathematics and Computer Science Division
Argonne National Laboratory

May 23, 2025

PARALLEL FILE SYSTEMS

A successful example of data service

PARALLEL FILE SYSTEMS

A successful example of data service

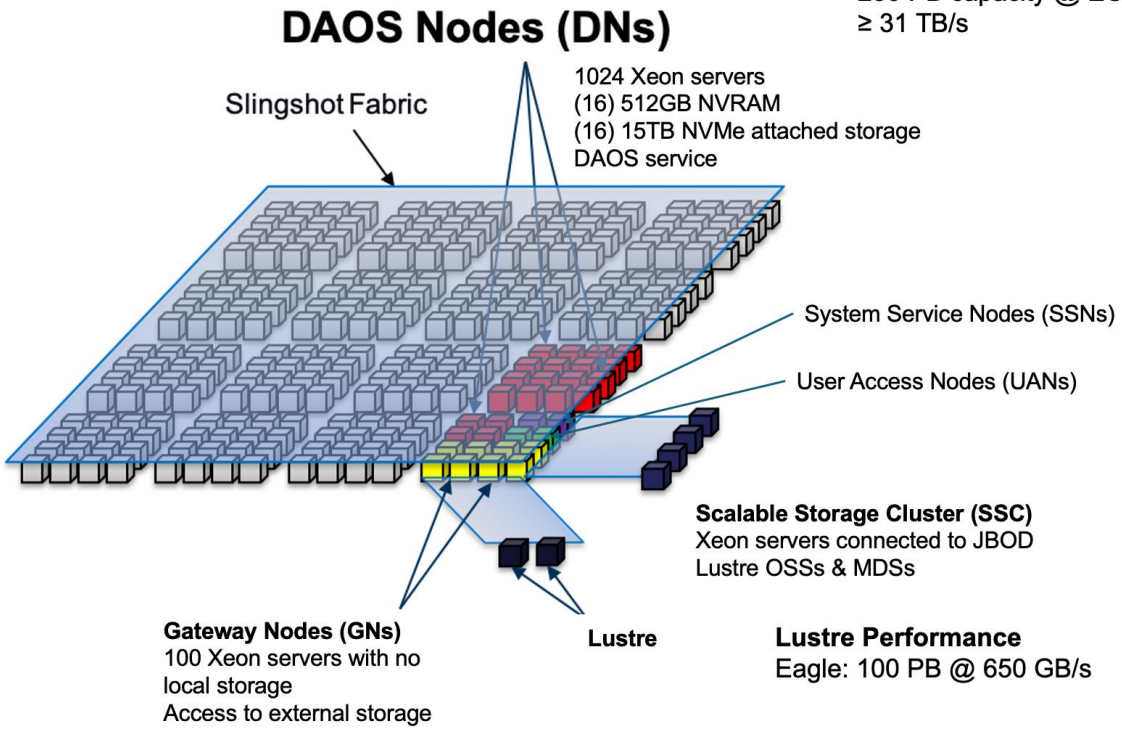
- A platform- or facility-wide file system must present a **general-purpose API** (usually POSIX files and directories).
- **Conservative semantics** are needed for the set of applications that *might* need it (e.g., directory locking for concurrent renames “just in case”).
- The software must be **complex** to manage concurrent storage, network, and server access, redundancy, security, high concurrency, and much more.
- The Unix/Linux OS model calls for file systems to be **closely tied to the operating system** for coherent access control.

Against all odds: parallel file systems are incredibly successful!
Could we use something different?

ALTERNATIVES ARE EMERGING

Example: DAOS on Argonne's Aurora

DAOS Performance
230 PB capacity @ EC16+2
≥ 31 TB/s



ALTERNATIVES ARE EMERGING

Many more data services, for many more data models

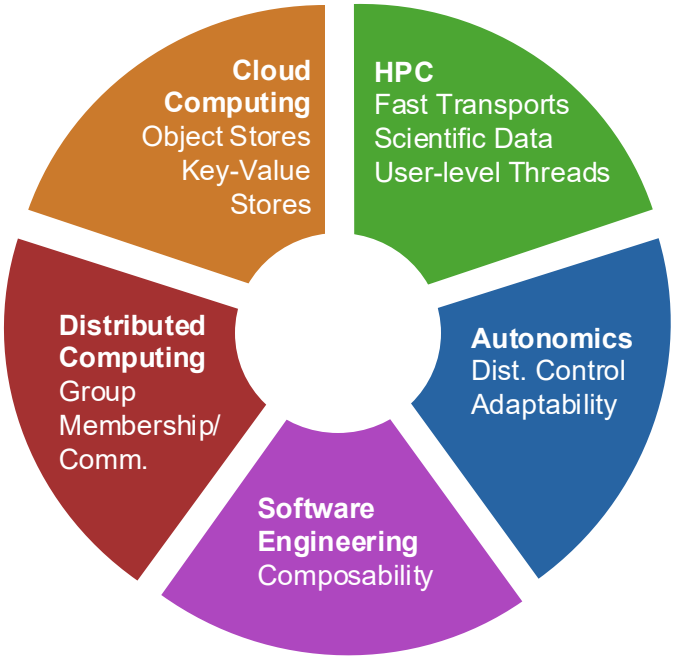
Redpanda

daos

ENTER… MOCHI

Mochi

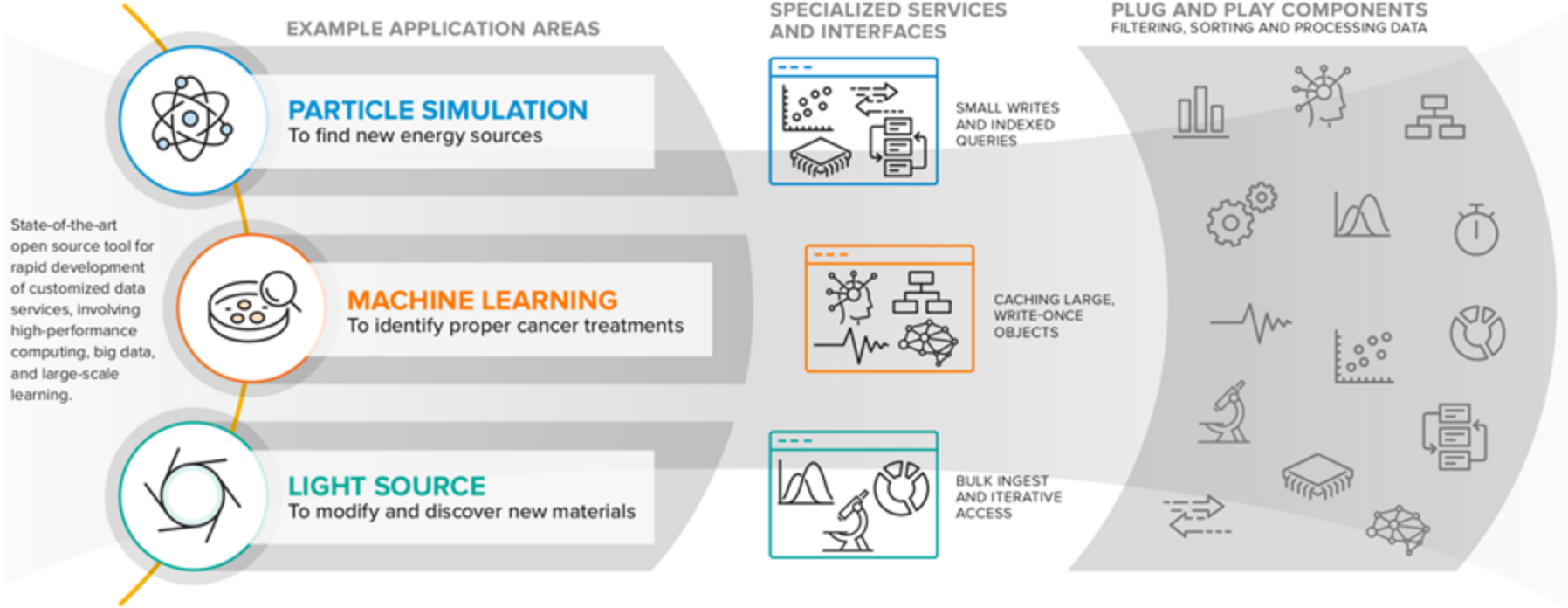
- Started in 2015 as an effort to explore what **composition** and componentization meant in the context of HPC storage services
- Grew into an effort to define a **methodology** and develop a set of **components** for building **HPC data services**
- Inspired by cloud computing, distributed computing, software engineering, autonomics, and HPC technologies



Objective: empower fast and innovative research and development in data management for HPC

MOCHI IN ONE PICTURE

Enable rapid development of data services tailored to applications



MOCHI'S TECHNICAL ROOTS

Mochi launched in 2015, but two key underpinnings predate it

Mercury

- HPC-oriented RPC framework
- Developed by ANL and The HDF Group
- Enables efficient access to native network transports for remote execution

Jerome Soumagne et al., "Advancing RPC for Data Services at Exascale", 2020

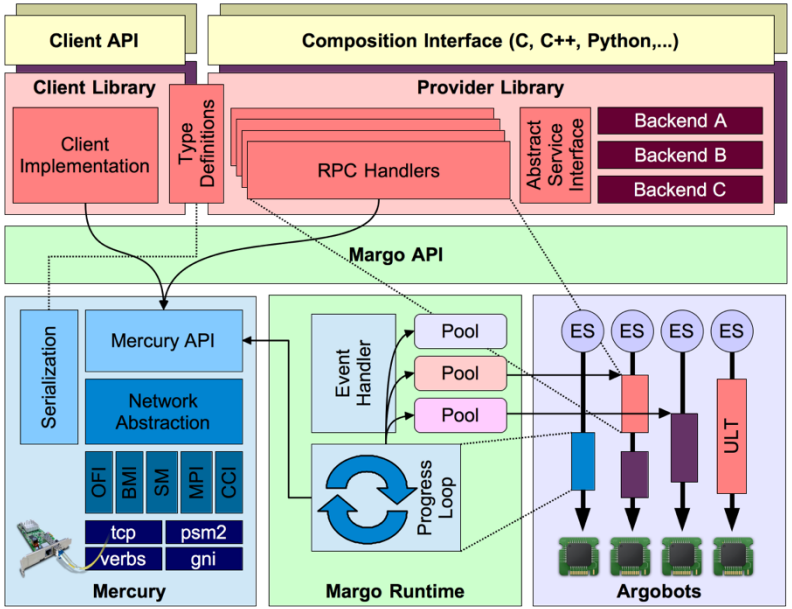
Argobots

- User-level threading framework
- Developed by ANL & collaborators
- Enables efficient management of concurrent, asynchronous execution paths

Sangmin Seo et al., "Argobots: A lightweight low-level threading and tasking framework", 2018

MOCHI'S COMPONENT MODEL

Simplifying component development



Margo (C) / Thallium (C++)

- Very easy to understand and program with
- Hides the Mercury progress loop
- No more callbacks! Everything is a ULT
- RPCs (Remote Procedure Calls) turned into ULTs
- Argobots takes care of scheduling to resources

Methodology

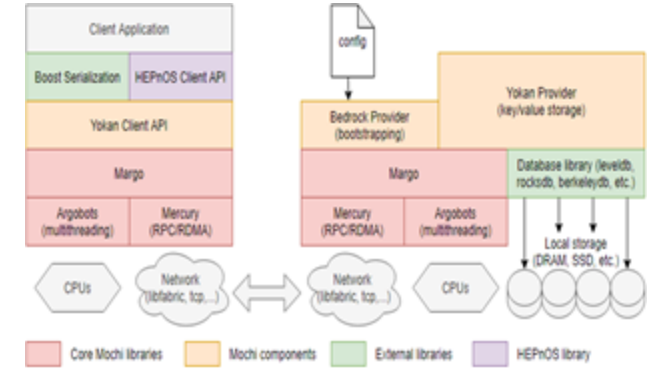
- Components provide a client and a server library
- Functionalities implemented in different ways
- Everything can be an RPC (even if everything executes in the same process or node)

MOCHI EXAMPLES

Ali *et al.*, "HEPnOS: a Specialized Data Service for High Energy Physics Analysis," *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) 2023.*

HEPnOS (below)

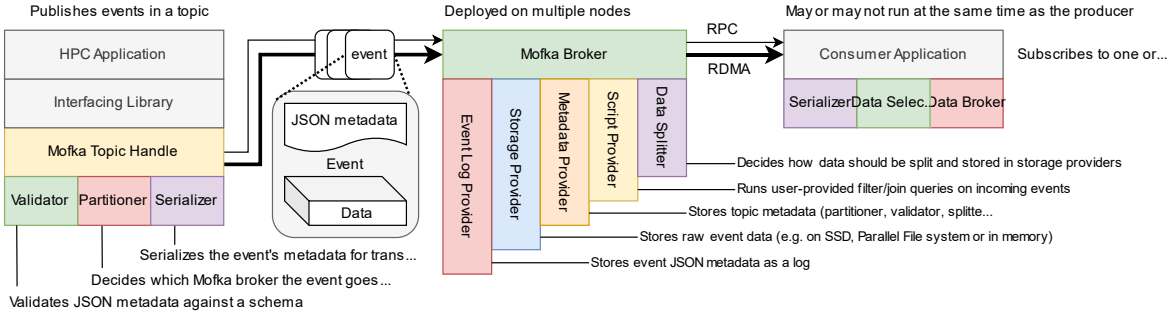
- Domain-specific service for HEP experiment analysis
- Presents hierarchical sorted data amenable to analysis



Mofka (below)

- Streaming event service
- Analogous to Kafka but tailored to scientific computing

https://mofka.readthedocs.io



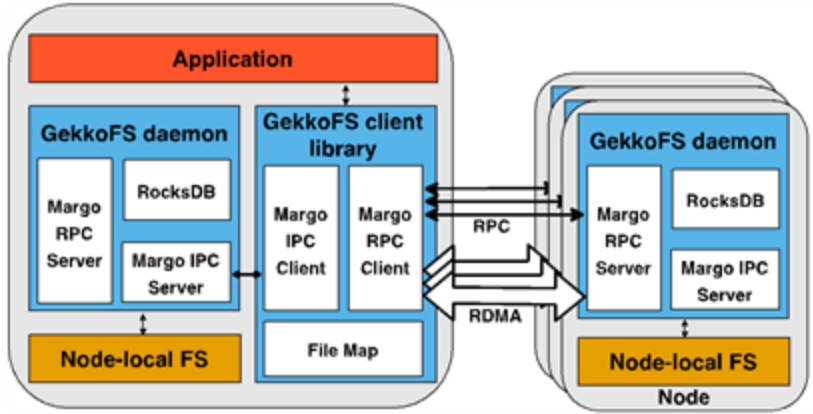
SUCCESS STORIES FROM MOCHI

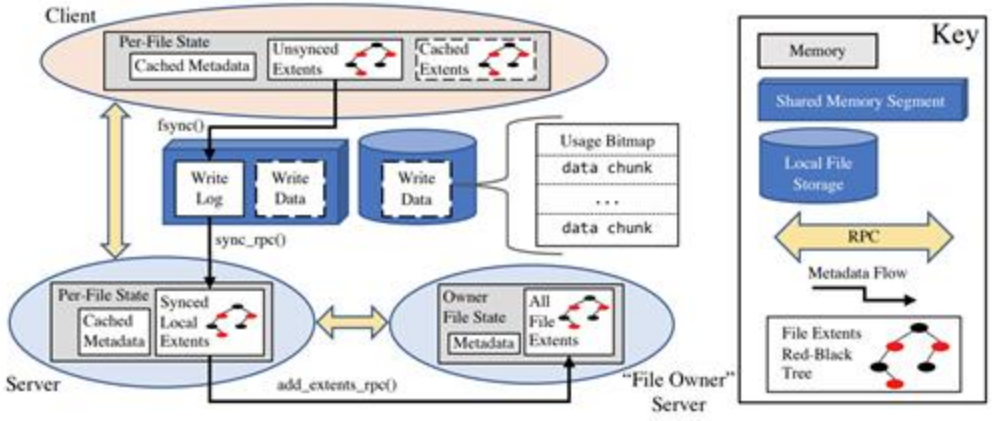
USER LEVEL FILE SYSTEMS

UnifyFS (below)

- Transient file system
- Emphasis on use of local storage during writes
- Delayed visibility via *laminate* operation

[Unify22]



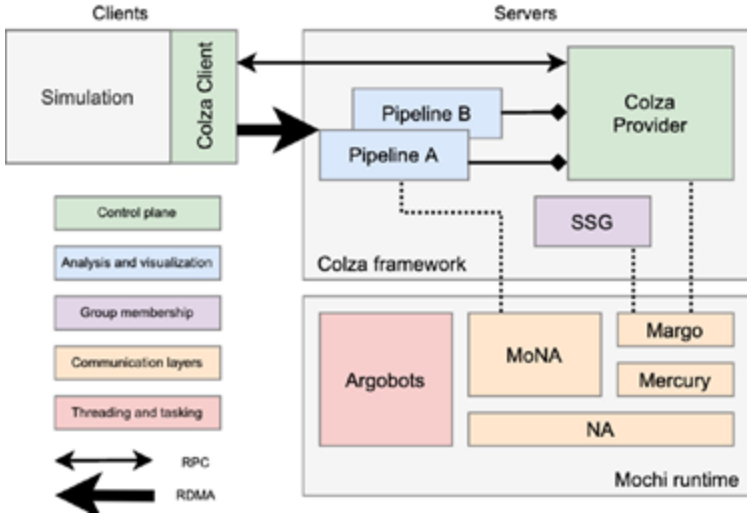


GekkoFS (above)

- Transient file system
- Sharded data
- Relaxed consistency in data and metadata paths

[Vef18]

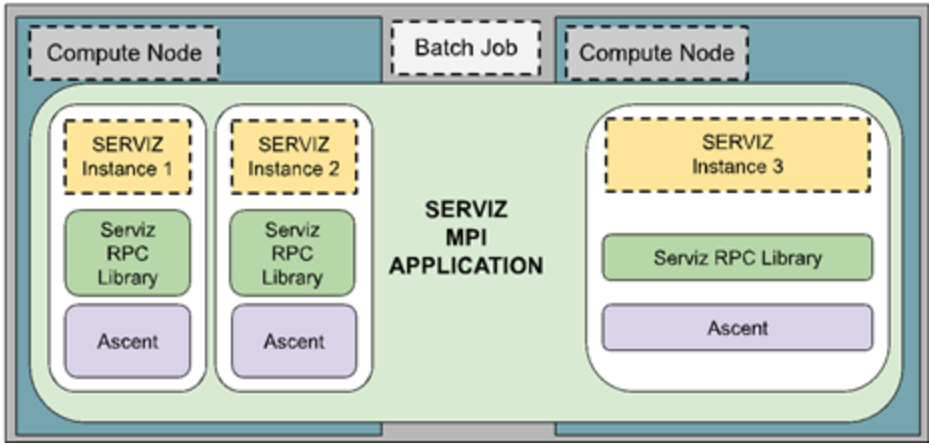
IN SITU DATA ANALYSIS



Colza

- Enables elastic in situ via addition/removal of visualization nodes
- Couples to Catalyst for visualization
- Replaces VTK comm. with Mochi

[Dorier22]

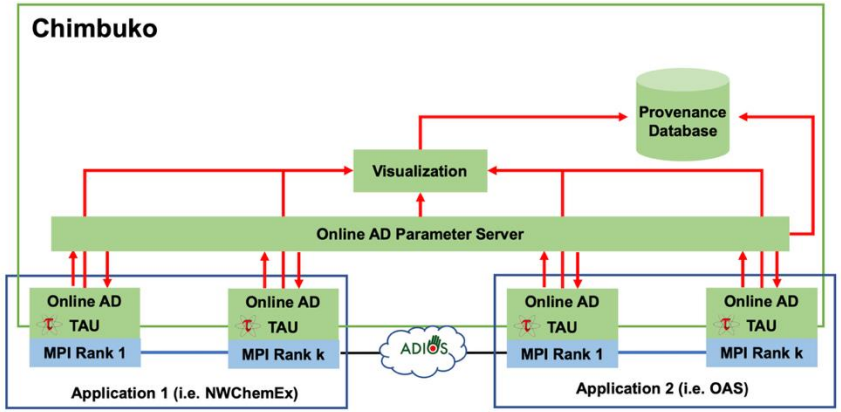


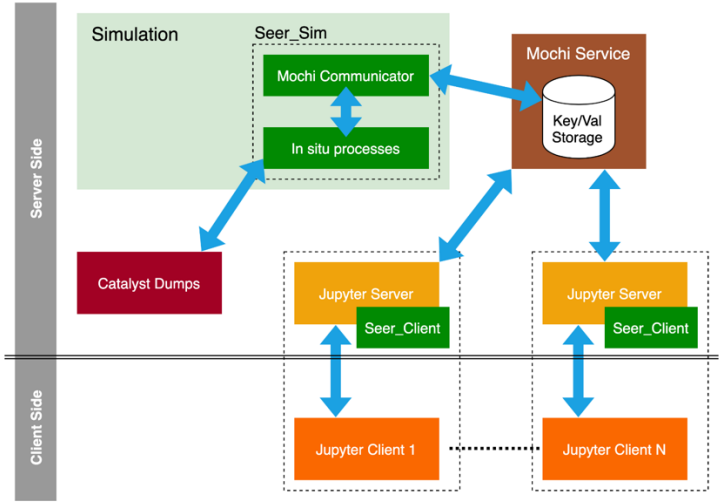
SERVIZ

- Visualization as a service, support for multiple applications
- Coupling to Ascent to leverage VTK ecosystem
- Ascent continues to use MPI

[Ramesh22]

PERFORMANCE DATA SERVICES





Chimbuko

- Performance trace data captured via TAU
- Local anomaly detection (AD) filters trace
- Provenance database allows real-time monitoring and analysis

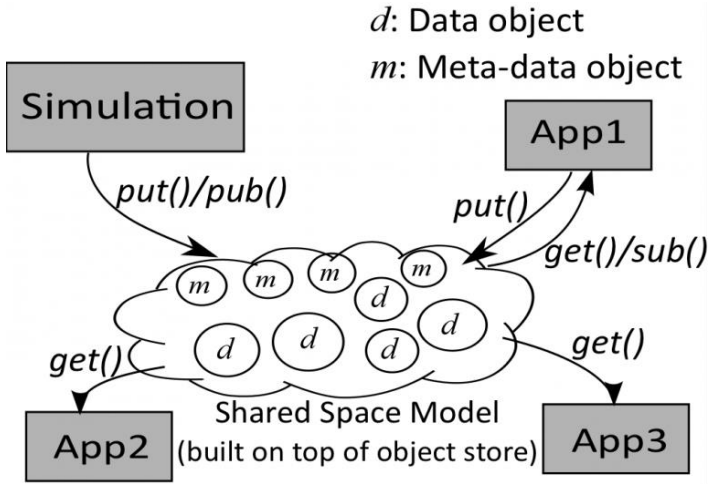
[Kelly20]

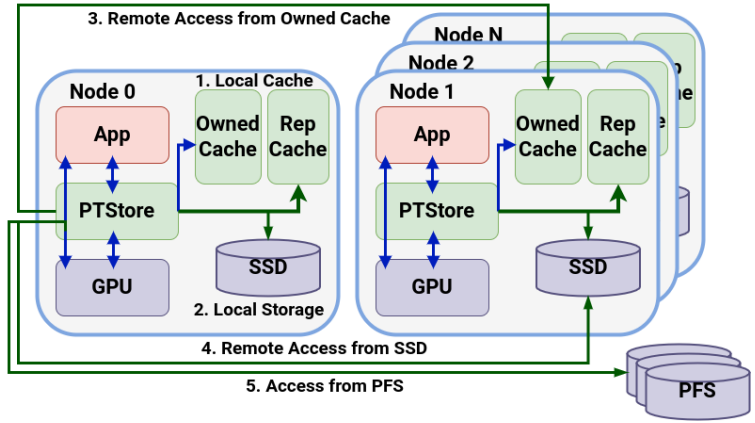
SEER

- Combines performance and scientific data visualization
- Allows multiple users to attach to simulation and adjust analysis on the fly
- Computational steering

[Grosset20]

ALTERNATIVE DATA MODELS





DataSpaces

- N-dimensional data model
- Coupling parallel applications in workflows

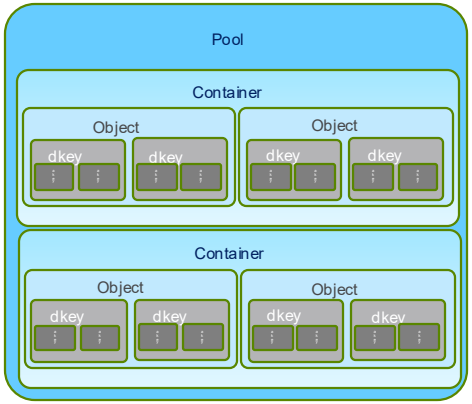
[Docan12]

DataStates, EvoStore, PTStore

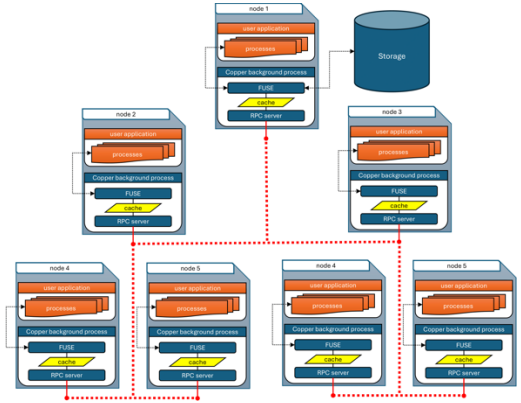
- Data services for AI (model checkpointing, distributed caching, etc)

[Nicolae20, Underwood23]

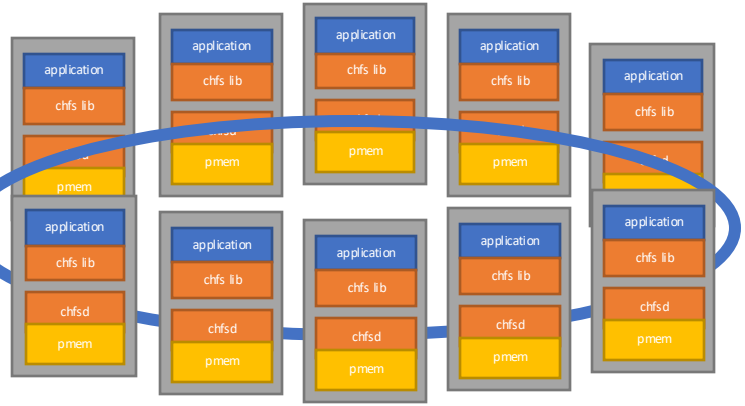
…AND THERE ARE MORE!



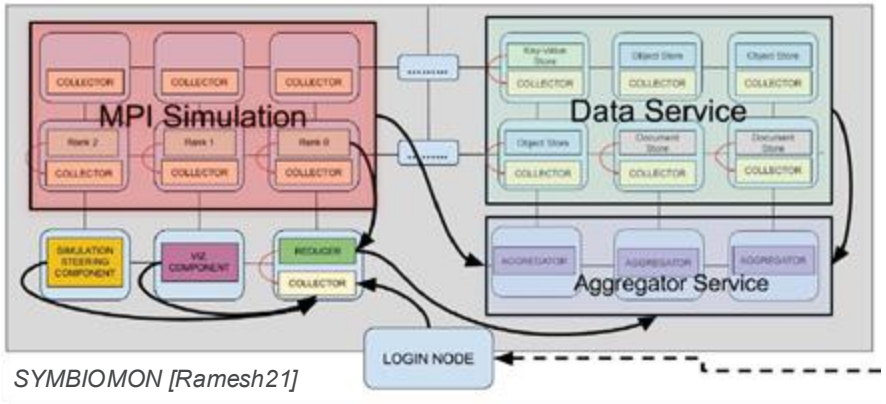
DAOS [Liang20]



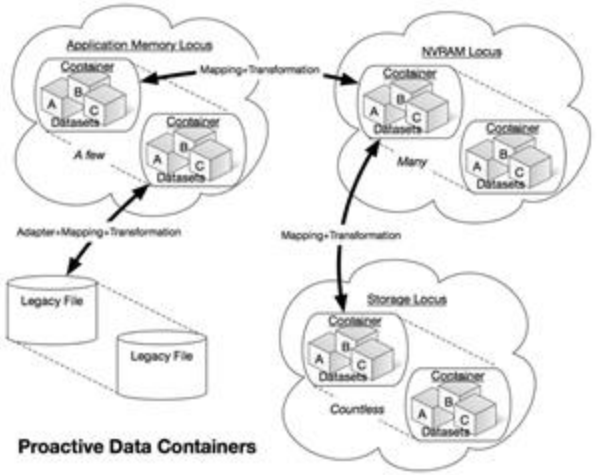
Copper [Lewis2024]



CHFS [Tatebe22]



SYMBIOMON [Ramesh21]



PDC [Tang18]

WHAT WE COULD HAVE DONE DIFFERENTLY

ARGONNE AS A TRACK RECORD IN OPEN SOURCE SOFTWARE

Understanding what the community needs is important

SOME COMPONENTS ARE MORE POPULAR THAN OTHER

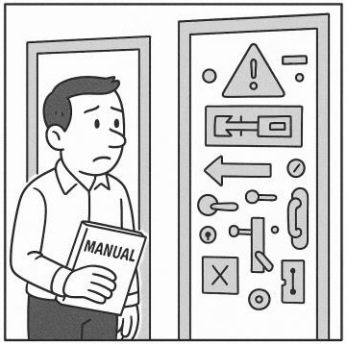
We should have understood why earlier

	Mercury	Argobots	Margo	Thallium	ABT-IO	Yokan / SDSKV	Warabi / Bake	Flock / SSG	Bedrock
Mofka (and other)	■	■	■	■	■	(Yokan)	(Warabi)	(Flock)	■
DeltaFS	■								
DAOS	■	■							
CHFS	■	■	■						
UnifyFS	■	■	■						
DYAD	■	■	■						
Copper	■	■	■	■					
OpenFAM	■	■	■	■					
Cargo	■	■	■	■					
Seer	■	■	■			(Yokan)			
GekkoFS	■	■	■		■				
Datastates-AI	■	■	■	■				(SSG)	
Chimbuko	■	■	■	■		(Sonata)			

UNDERSTANDING COMPONENT ADOPTION

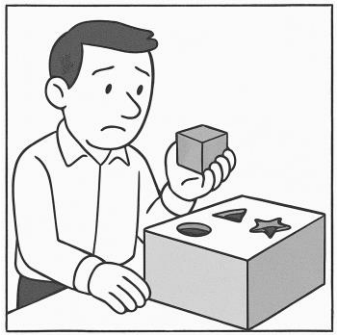
Why are some components not more widely adopted?

Lack of documentation?



Too complex?

Missing features?

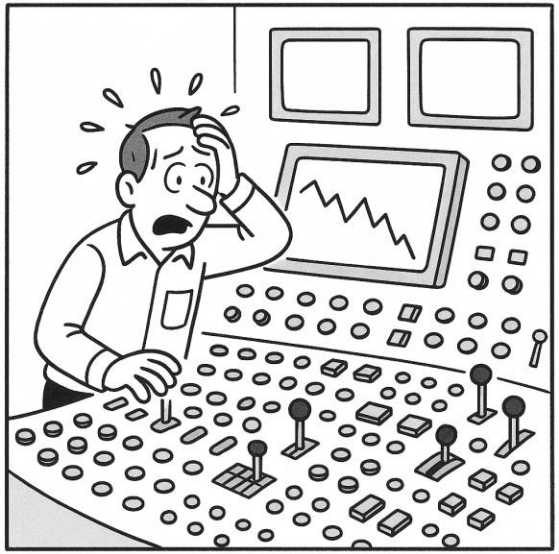


API not right?

- Mochi has **great documentation** for Margo and Thallium, documentation for other components is **in progress**
- Margo and Thallium are **very easy to use**, some users jump to implementing services and **don't look beyond them**
- We assumed that **missing features = users will contribute**, in practice users will **implement their own thing**
- **Reusable components** = more generic API, but users may **need something specific**

MORE COMPONENTS = MORE TUNING

Knowing what configuration works best is a hassle



- Each component may have multiple implementations
- Each implementation has its own set of parameters
- Scheduling and thread placement options are infinite
- Composition choices affect performance

We should provide tools to better understand how to tune a Mochi service, as well as auto-tuning tools

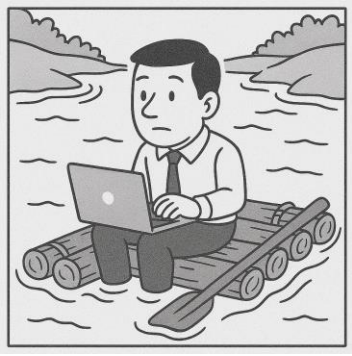
- HPC Storage Service Autotuning Using Variational-Autoencoder-Guided Asynchronous Bayesian Optimization, Dorier et al. 2022 (Cluster)
- **We are working on such tools!**
- Importantly: they should be intuitive enough

FOCUSING ON MORE COMPLEX ASPECTS?

Because a key/value component is easy to write yourself

Resilience

Auth^2 / Encryption



Consistency

Introspection

- Resilience is difficult to implement correctly, we can provide methodologies, APIs, and tools
- Authentication, authorization, and encryption allow multi-user services
- Consistency involves protocols such as Raft, two-phase commit, etc. which Mochi could provide
- Introspection would allow users to understand the performance of their services better

THE FUTURE OF MOCHI

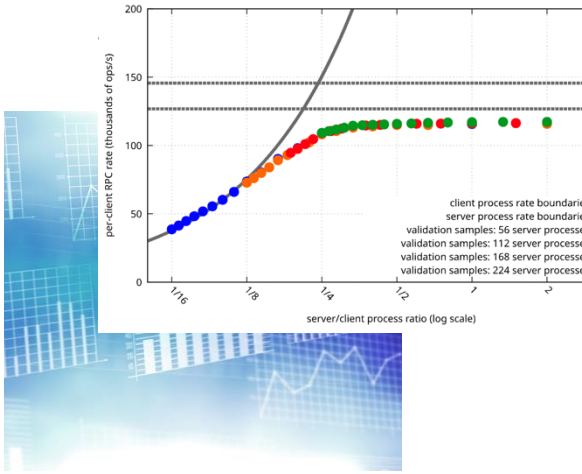
THE FUTURE OF MOCHI

New directions have opened up

Data services for AI

Energy efficient data movements

[Carns2025] (CUG)



Enhanced performance characterization

THE TEAM

Phil Carns (PI), Matthieu Dorier, Amal Gueroudji, Rob Latham,
Shane Snyder, and Rob Ross (former PI)
Argonne National Laboratory

Tyler Reddy, Kyle Roarty, Galen Shipman,
and Qing Zheng
Los Alamos National Laboratory

George Amvrosiadis, Chuck Cranor, and Ankush Jain
Carnegie Mellon University

* also long-time contributions from Jerome Soumagne of HPE, formerly Intel, formerly The HDF Group

THANK YOU!

THIS WORK WAS SUPPORTED BY THE U.S. DEPARTMENT OF ENERGY, OFFICE OF SCIENCE, ADVANCED SCIENTIFIC COMPUTING RESEARCH, UNDER CONTRACT DE-AC02-06CH11357.