Composing HPC Micro-Services to Build Application-Tailored Distributed Object Stores

Matthieu Dorier
mdorier@anl.gov
Argonne National Laboratory

SIG-IO-UK Workshop - Reading, UK, June 6th, 2018

Mochi Project

Software Defined Storage

DOE project 2015-present

Argonne NATIONAL LABORATORY

Los Alamos NATIONAL LABORATORY EST. 1943

HDF The HDF Group

CARNEGIE MELLON UNIVERSITY PITTSBURGH PENNSYLVANIA 1900

Existing storage systems provide diverse features

- Data distribution
- Indexing methods
- Access semantics
- Transactions and locking
- Fault tolerance, replication

But they are not tailored to each application individually

However, they build on similar components

- RPC mechanism
- Threading/tasking management
- Storage management
- Metadata management
- Group membership

Let's split these building blocks and recompose them according to each application's needs

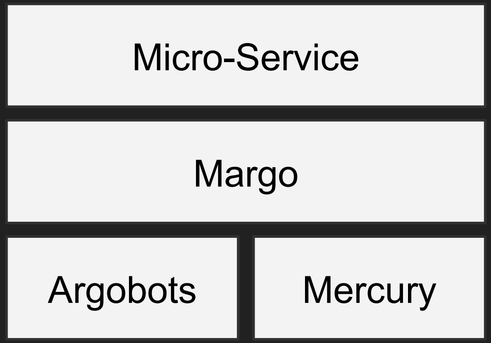
Composing HPC Microservices

- Formalize composition
- Unify single-process, multi-process, single-node, and multi-node designs
- Maximize efficient use of resources (network, storage)

Mochi building blocks

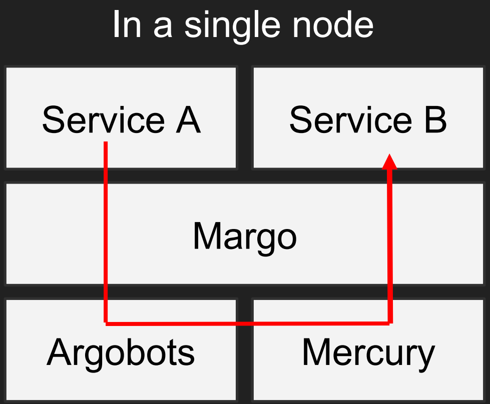
- MERCURY: RPC library with RDMA support and many network backends
- ARGOBOTS: Threading/tasking framework
- MARGO: Higher-level, ARGOBOTS-enabled MERCURY interface
- BAKE: RDMA-enabled data transfer to local storage (e.g. SSD, NVRAM)
- SDSKV: Key/Value store backed by LevelDB or BerkeleyDB
- SSG: Scalable Service Groups, group membership management
- MDCS: Lightweight diagnostic component
- PLASMA: Distributed approximate k-NN database
- POESIE: Enables running Python and Lua interpreters in Mochi services
- THALLIUM: C++14 wrapper for Margo
- Python wrappers: Py-Margo, Py-Bake, Py-SDSKV, Py-SSG, Py-Mobject, etc.

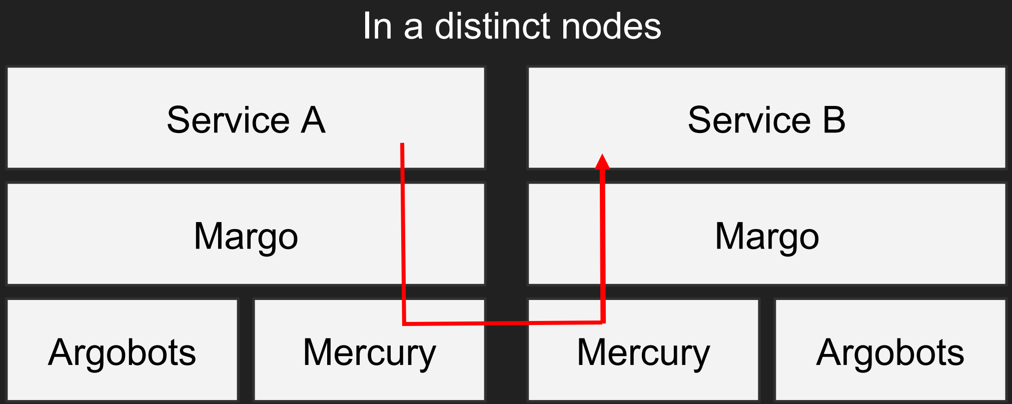
Mochi micro-services



- Mercury: RPC/RDMA
- Argobots: Threading/Tasking
- Margo: Mercury+Argobots

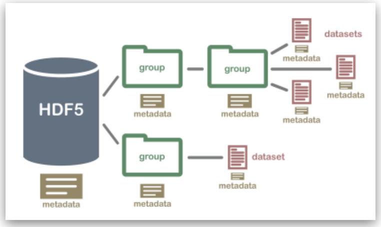
Different deployments; same code!

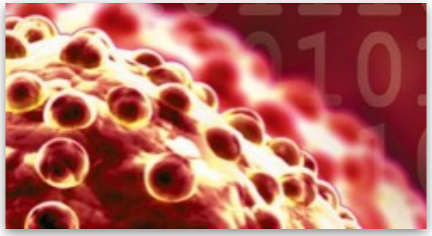




Different users

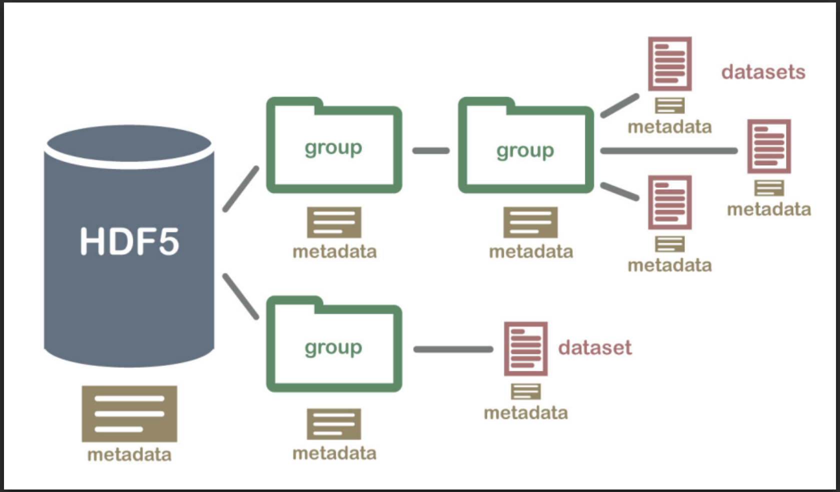
Different needs





Mobject

From microservices
to object store



The HDF Group

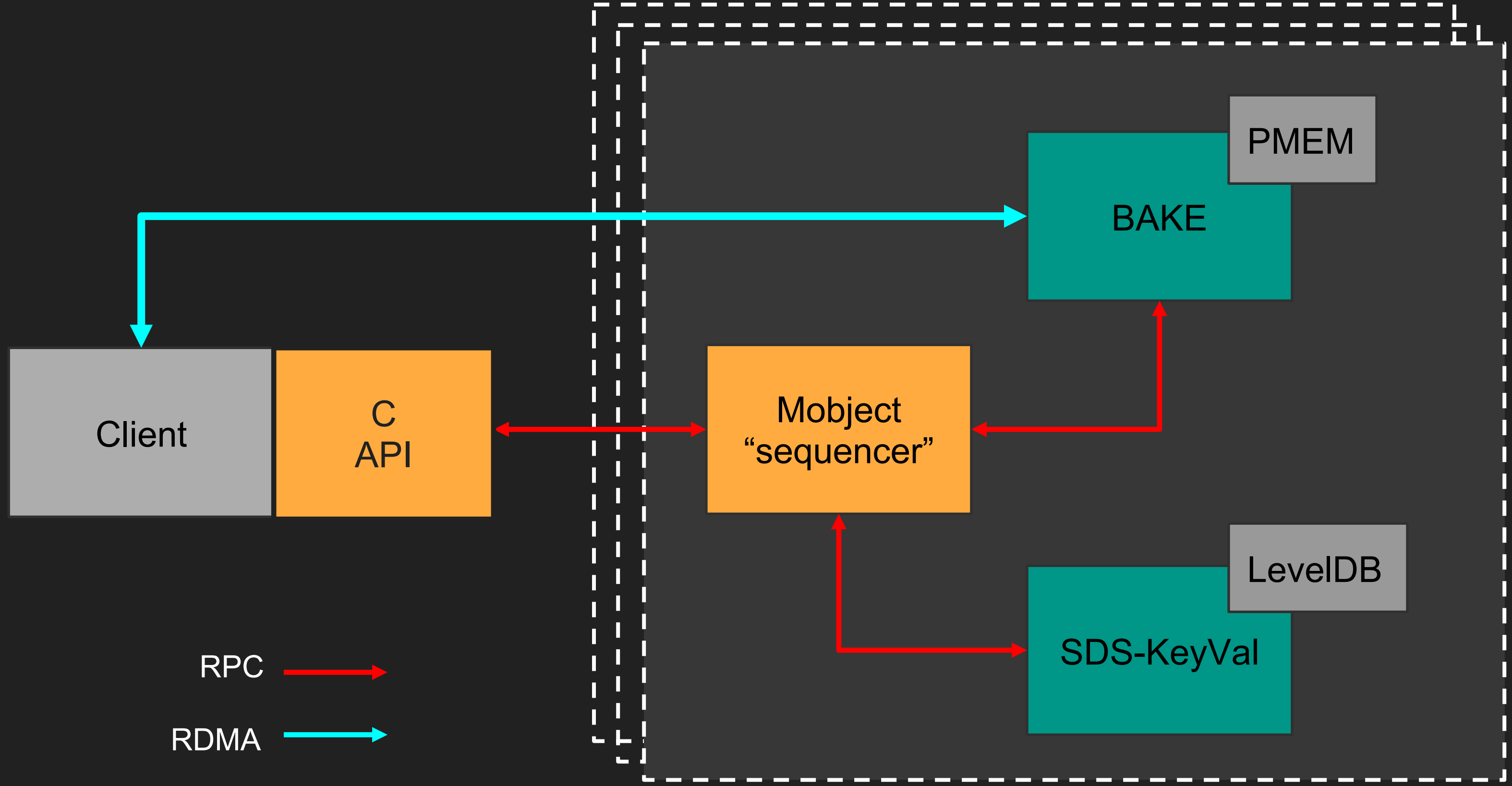
Mobject: from microservices to object store

- Transaction-enabled
- Flat namespace
- RADOS client API

- Components used: MERCURY, ARGOBOTS, MARGO, SDSKV, BAKE SSG

- Extra code: Sequencer, "RADOS-like" API

Mobject: from microservices to object store



HEPnOS

Fast event-store for High Energy Physics experiments

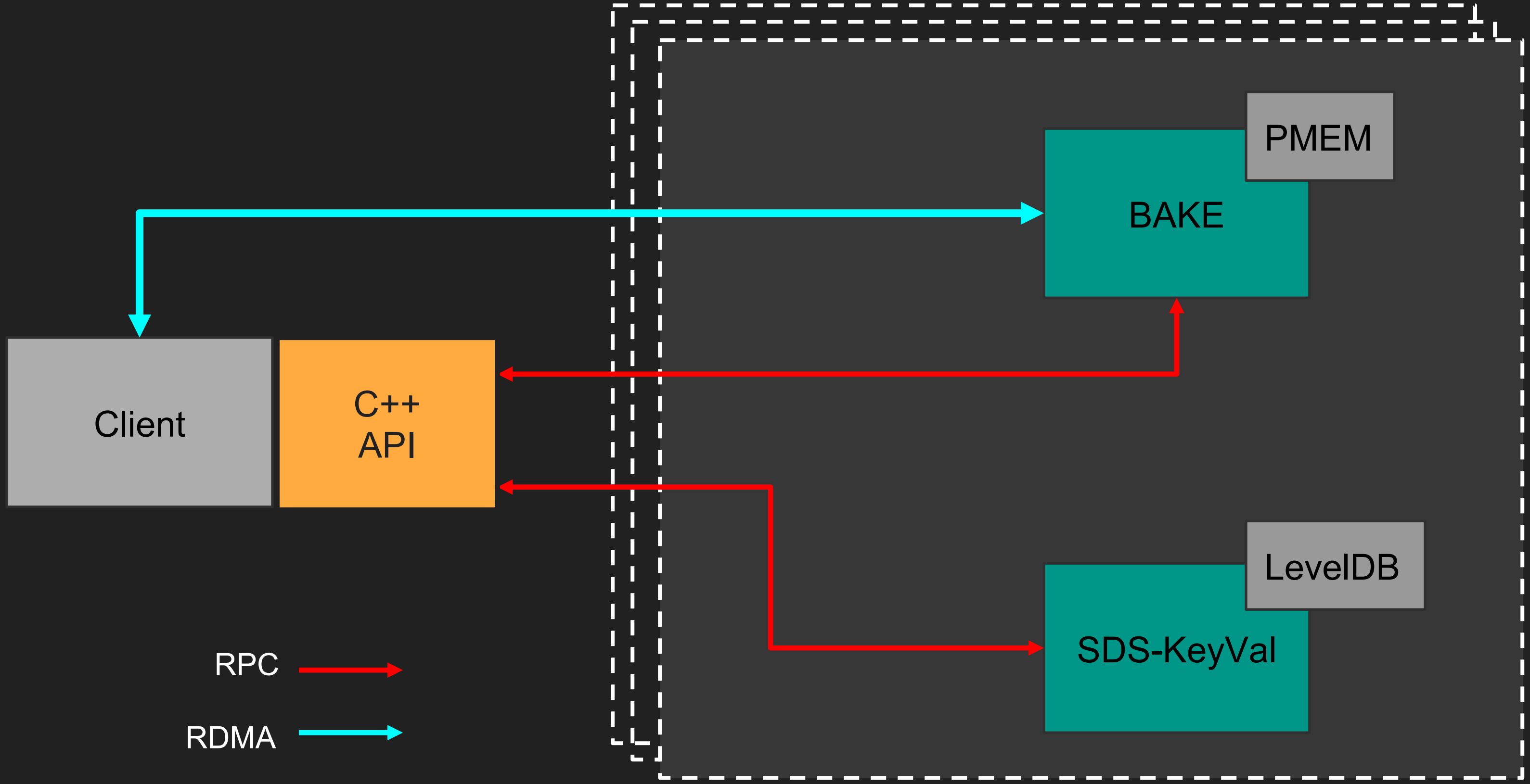
HEPnOS: fast event-store for High-Energy Physics experiments

- Write-once-read-many
- Hierarchical namespace (datasets, runs, subruns)
- C++ API (serialization of C++ objects)

- Components used: MERCURY, ARGOBOTS, MARGO, SDSKV, BAKE, SSG

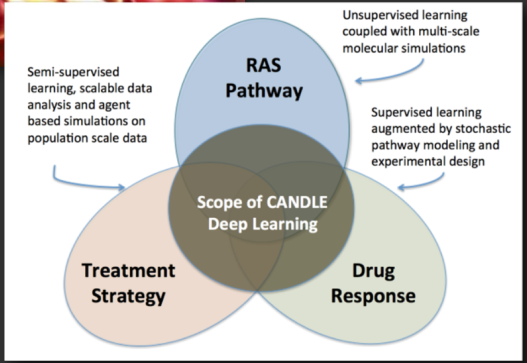
- Extra code: C++ interface

HEPnOS: fast event-store for High-Energy Physics experiments



FlameStore

A transient storage system for deep neural networks


Unsupervised learning coupled with multi-scale molecular simulations
RAS Pathway
Semi-supervised learning, scalable data analysis and agent based simulations on population scale data
Supervised learning augmented by stochastic pathway modeling and experimental design
Scope of CANDLE Deep Learning
Treatment Strategy
Drug Response

Argonne NATIONAL LABORATORY

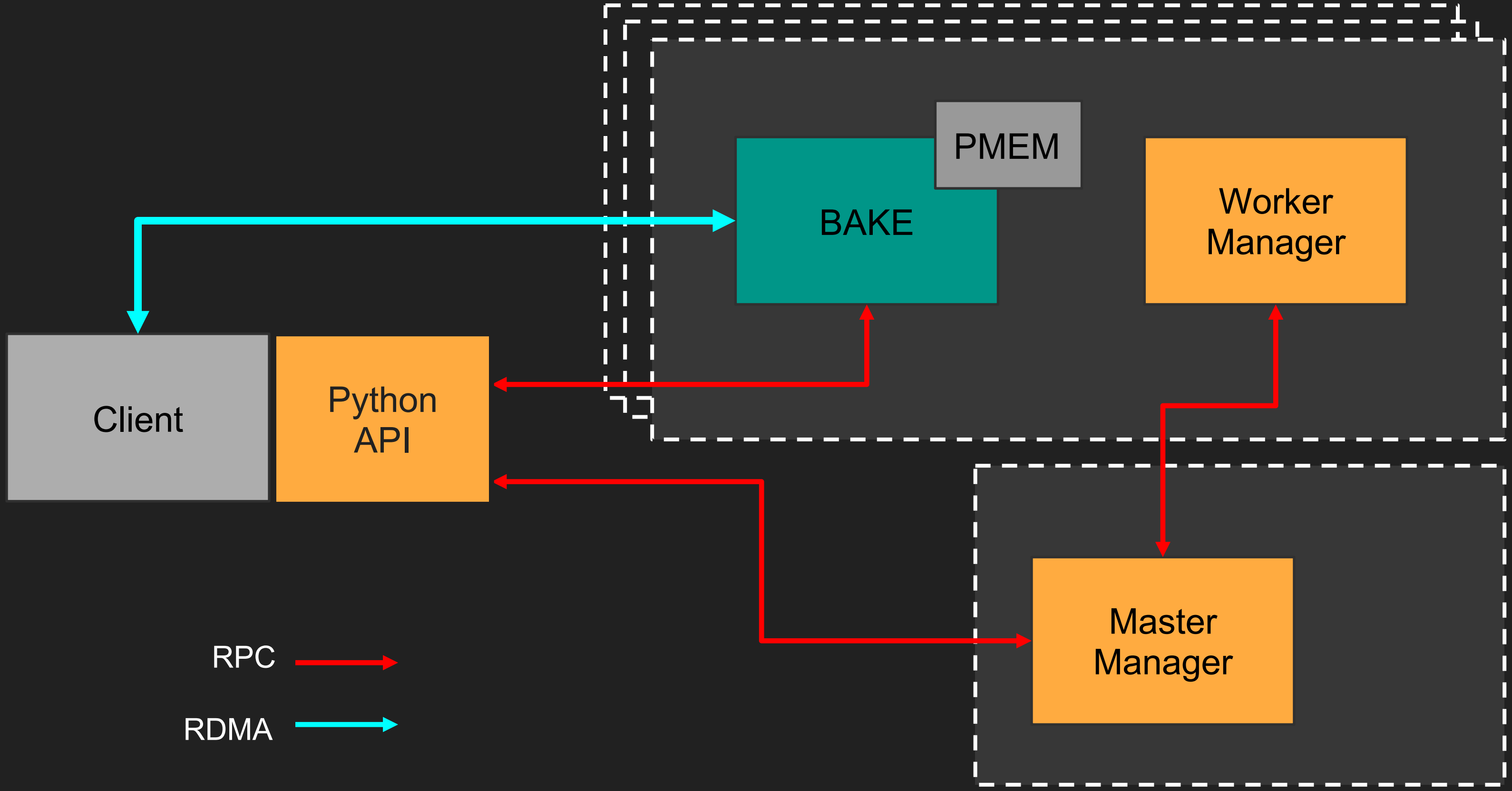
FlameStore: A transient storage system for deep neural networks

- Write-once-read-many
- Flat namespace
- High level of semantics
- Python API (stores Keras models)

- Components used: MERCURY, ARGOBOTS, MARGO, BAKE, POESIE, and their Python wrappers

- Extra code: Python API, master and worker managers

FlameStore: A transient storage system for deep neural networks



What we plan to study next

- Deployment and Sharding
  - single vs multiple Key/Value component(s)
  - collocated vs remote components
  - various object sharding policies
- Elasticity/malleability
  - Deploying and shutting down components at run time
  - Migrating components