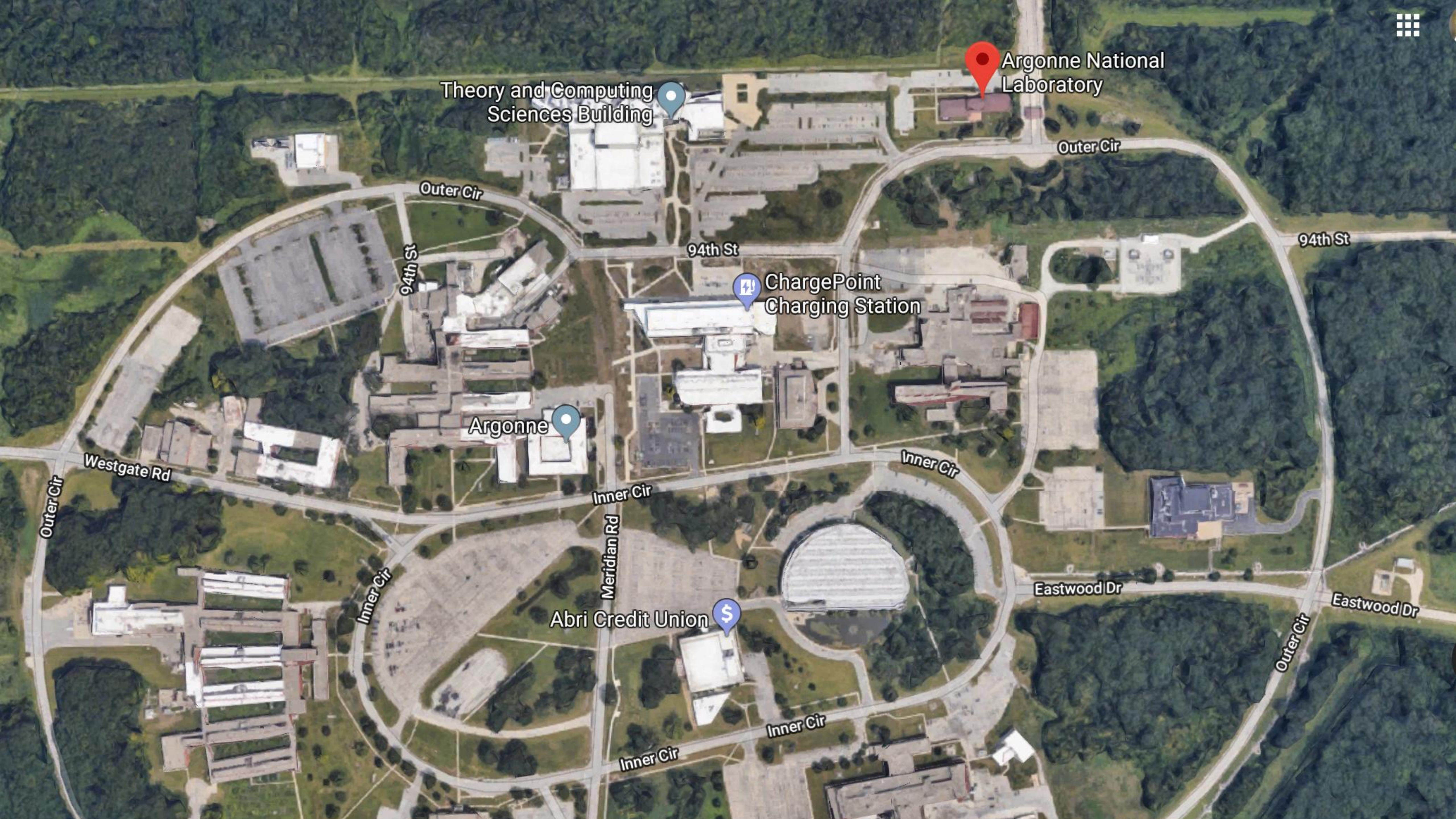
Welcome to the first Mochi Boot Camp!


Argonne National Laboratory
Theory and Computing Sciences Building
Outer Cir
94th St
ChargePoint Charging Station
Argonne
Westgate Rd
Inner Cir
Meridian Rd
Abri Credit Union
Eastwood Dr

Time for Introductions!

The Mochi Landscape

Mochi

http://www.mcs.anl.gov/research/projects/mochi/

Vision

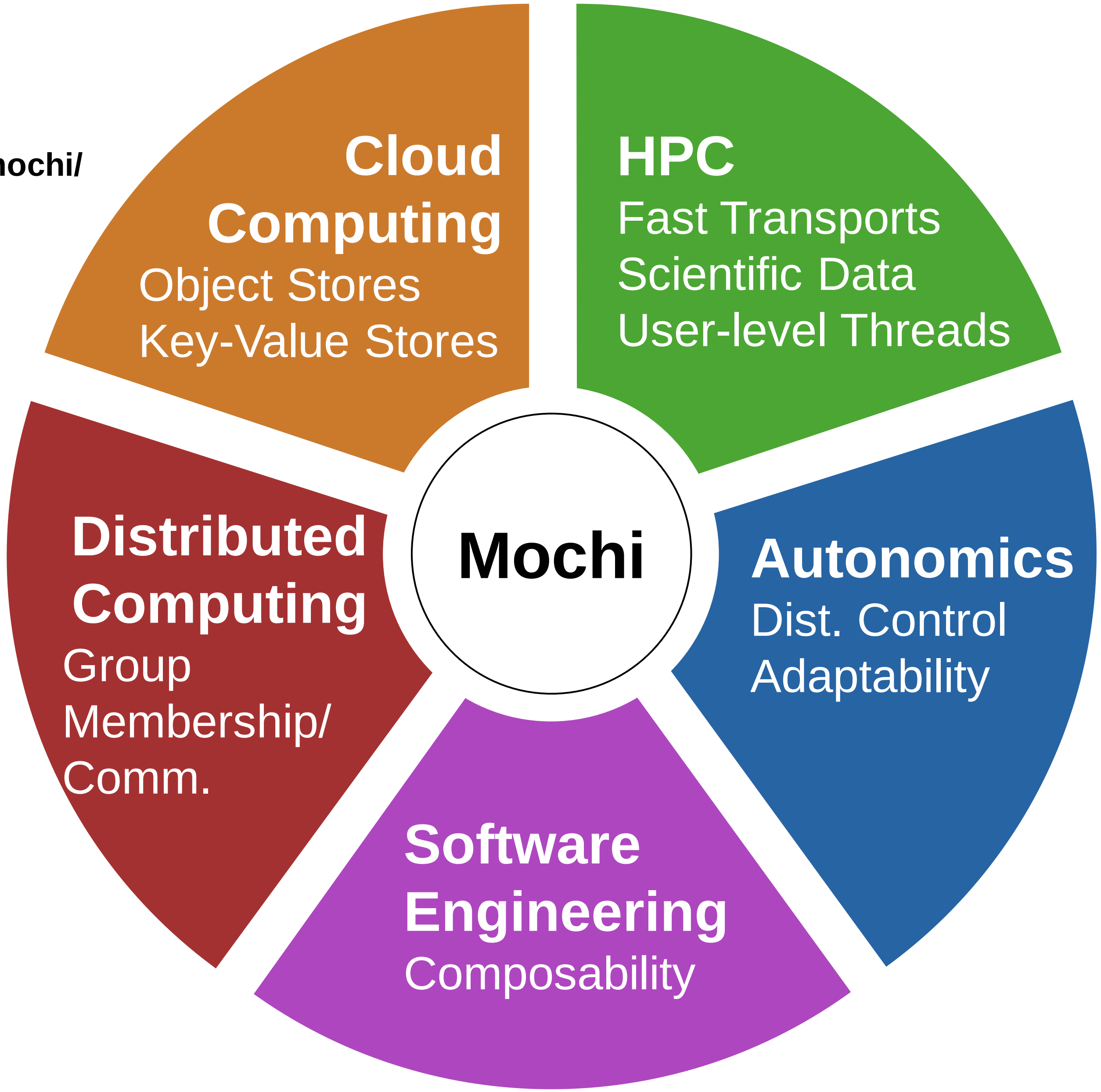
Lowering the barriers to distributed services in computational science.

Approach

- Familiar models (key/value, object, file)
- Easy to build, adapt, and deploy
- Lightweight, user-space components
- Modern hardware support

Impact

- Better, more capable services for specific use cases on high-end platforms
- Significant code reuse
- Ecosystem for service development



Mochi: What are we trying to accomplish?

We're trying to transform HPC data services from a monoculture to an ecosystem.

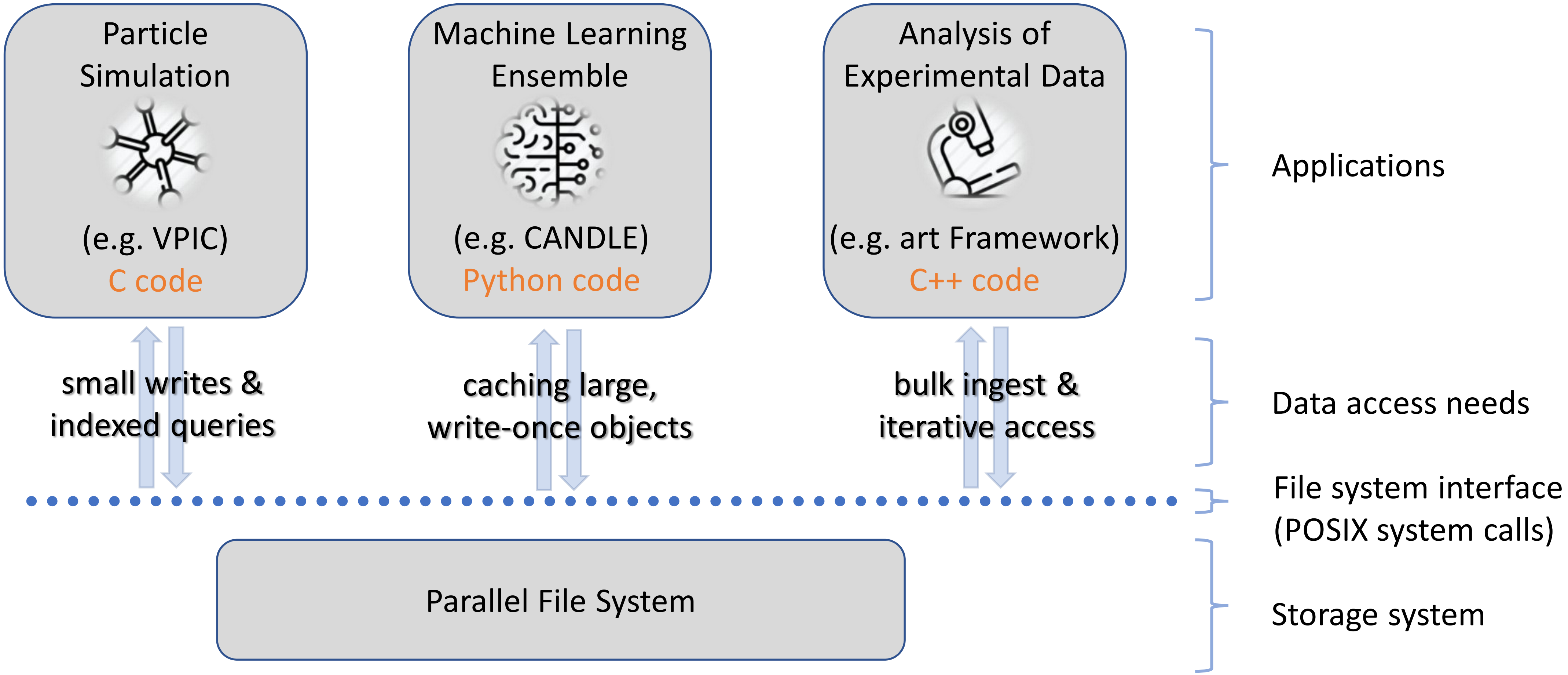
- Redefining how teams design and develop distributed services for use in HPC systems.
- Providing a portable "programming model" for these services.
- Providing a set of core building blocks.
- Demonstrating the methodology and tools with DOE science use cases.

We're trying to foster a community of service developers.

- Developing a set of training materials that will help others employ the tools.
- Making all these building blocks available to the larger community.

How is this traditionally done in HPC?

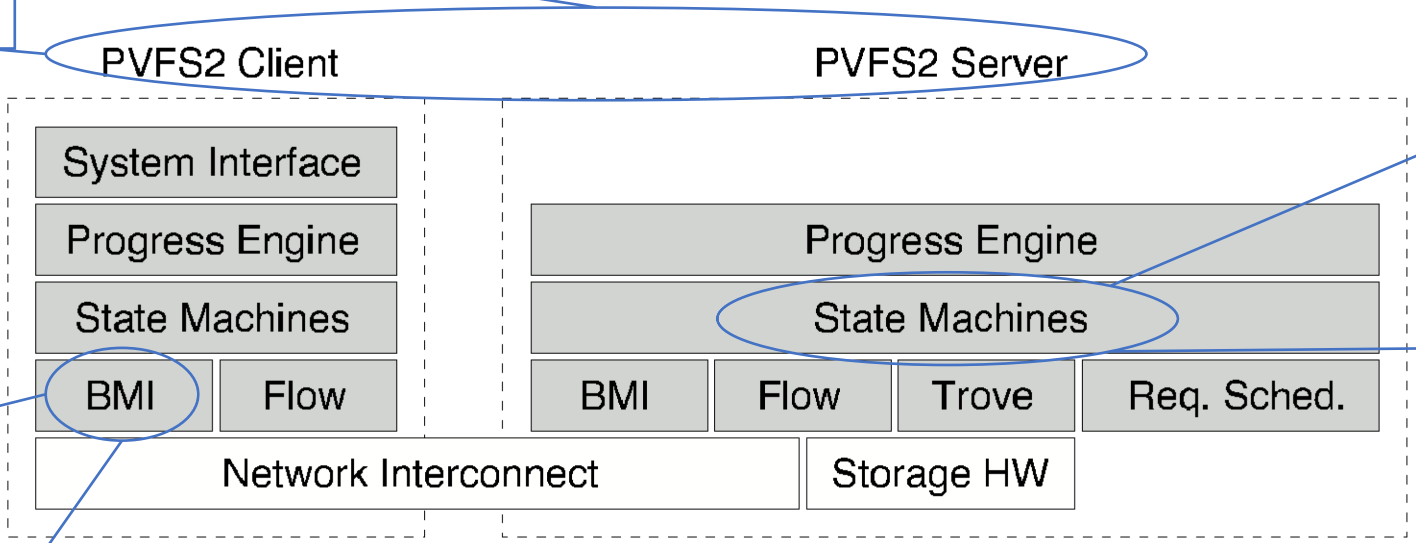
File system monoculture for data (dis)service



All applications use the same "one size fits all" file system interfaces, semantics, and policies for data access.

How is this traditionally done in HPC?
The internals of a parallel file system (PVFS2 example)

1. All applications bend to the same service: single data model (string of bytes), decomposition (blocks), and consistency.



3. Programming is accomplished with proprietary state machine language. Effective but steep learning curve.

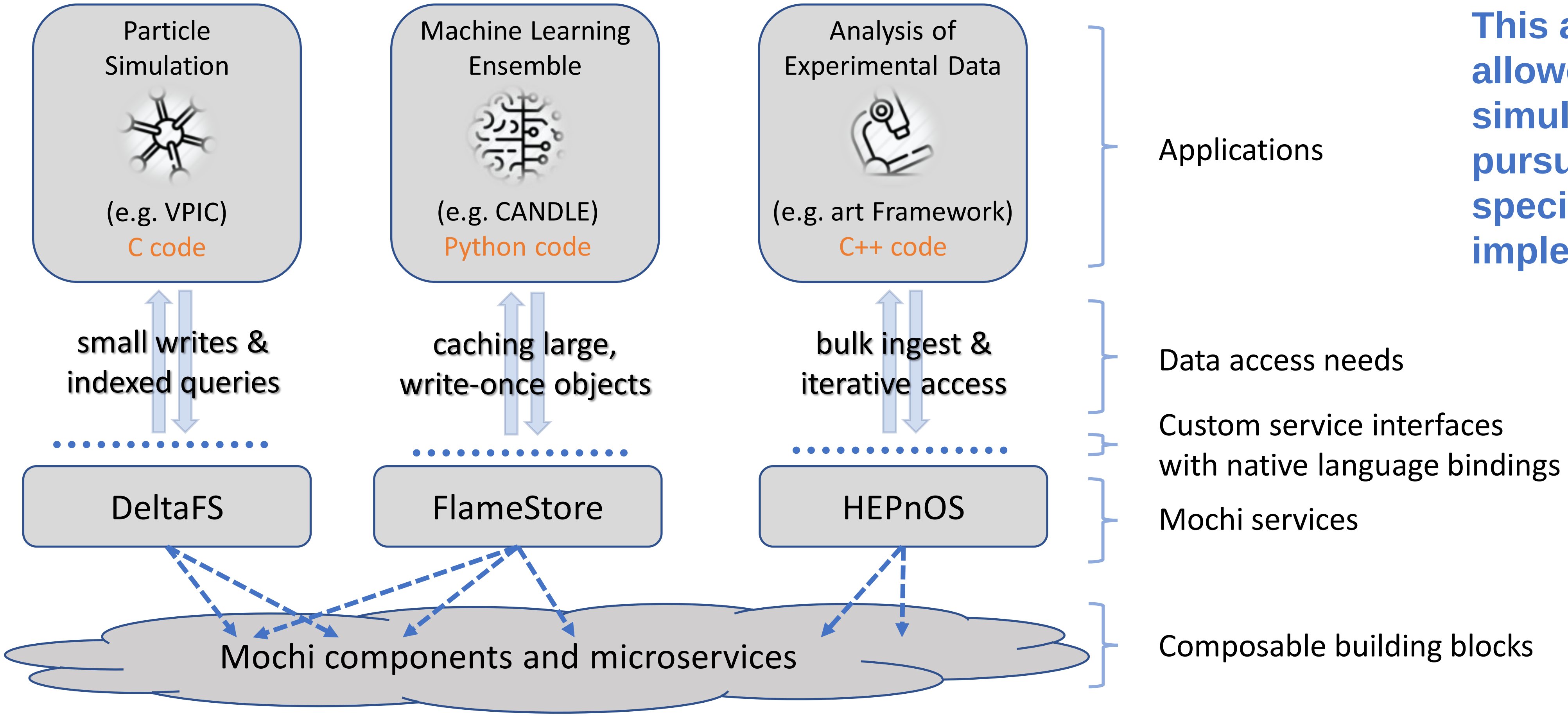
2. Portable networking layer (BMI) is the only component usable separate from PVFS. Service developers build RPC abstraction on top of it.

(Many) other researchers employ PVFS in their work, but it is difficult to build a *new* service from it.

What’s new in the Mochi approach?

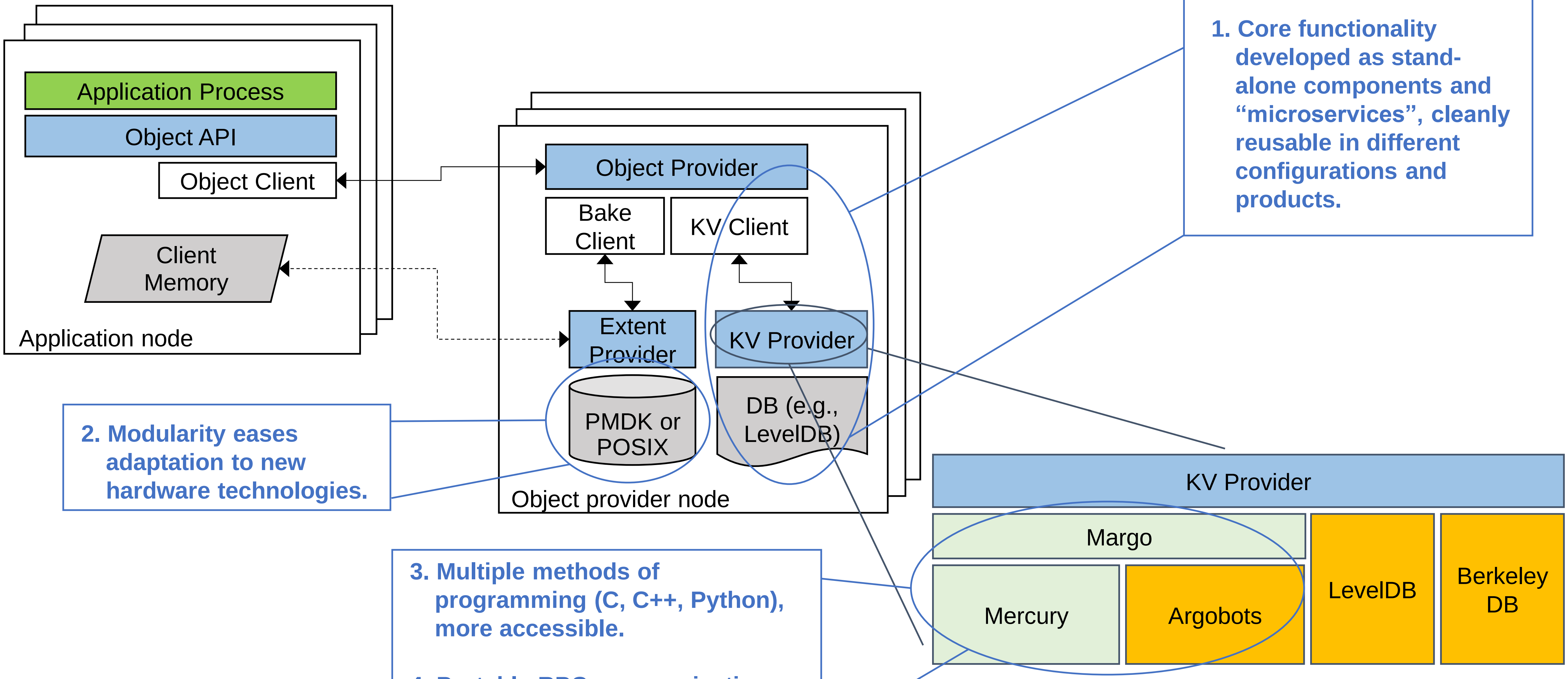
An ecosystem of services co-existing and reusing functionality

This approach has allowed us to simultaneously pursue multiple specialized service implementations.



Instead of “one size fits all”, Mochi data services present tailored interfaces, semantics, and policies for data access while still leveraging robust building blocks.

What's new in the Mochi approach?



1. Core functionality developed as stand-alone components and "microservices", cleanly reusable in different configurations and products.

2. Modularity eases adaptation to new hardware technologies.

3. Multiple methods of programming (C, C++, Python), more accessible.

4. Portable RPC communication library designed for multi-service environments

Carnegie Mellon University · HDF · Los Alamos National Laboratory EST. 1943 · Argonne National Laboratory

Mochi Composed Services

Fast Event-Store for High-Energy Physics (HEPnOS)

Goals

- Manage physics event data through multiple analysis phases
- Retain data in the system to accelerate analysis

Features

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

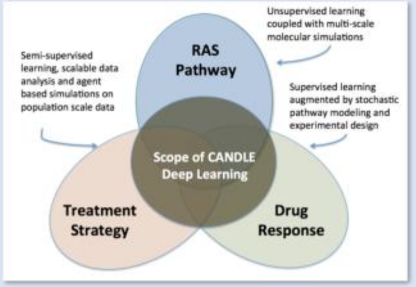
Transient Storage for Deep Neural Networks (FlameStore)

Goals

- Store deep neural network models during a deep learning workflow
- Retain most promising candidate models

Features

- Flat namespace
- Python API (Keras models)



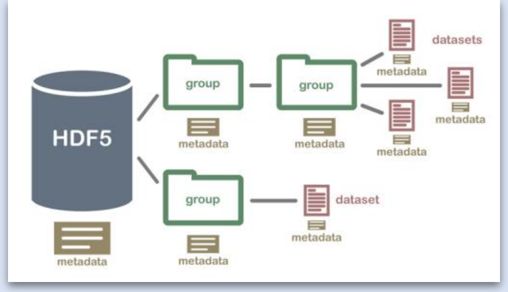
In-System Object Store (Mobject)

Goals

- Provide familiar model as alternative to POSIX

Features

- Concurrent read/write
- Flat namespace
- RADOS client API (subset)



Mochi Bits and Pieces

	Component	Client	Server	Other	External Users
Core					
	Argobots				Intel, LLNL, Mainz
	Mercury				Intel, LBL, LLNL, Mainz
	Margo				Intel, LLNL, Mainz
	Thallium				
	SSG				
	MDCS				
	Nexus				
Microservices					
	SDSKV				
	BAKE				
	POESIE				
Composed Services					
	HEPnOS				FNAL
	FlameStore				
	DeltaFS				
	Mobject				

Mochi: Core Components

A programming environment for distributed services

Mercury

Portable RPC comm.

- Support for shared memory and multiple HPC transports
  - PSM2
  - IB
  - GNI
- RDMA for bulk data movement
- Busy wait or wake on network event

https://mercury-hpc.github.io/

Argobots

Lightweight runtime for concurrent execution.

- Utilize HW and OS constructs for performance
- hwloc-aware
- Custom scheduling and placement

http://www.argobots.org/

Margo

Simple service development.

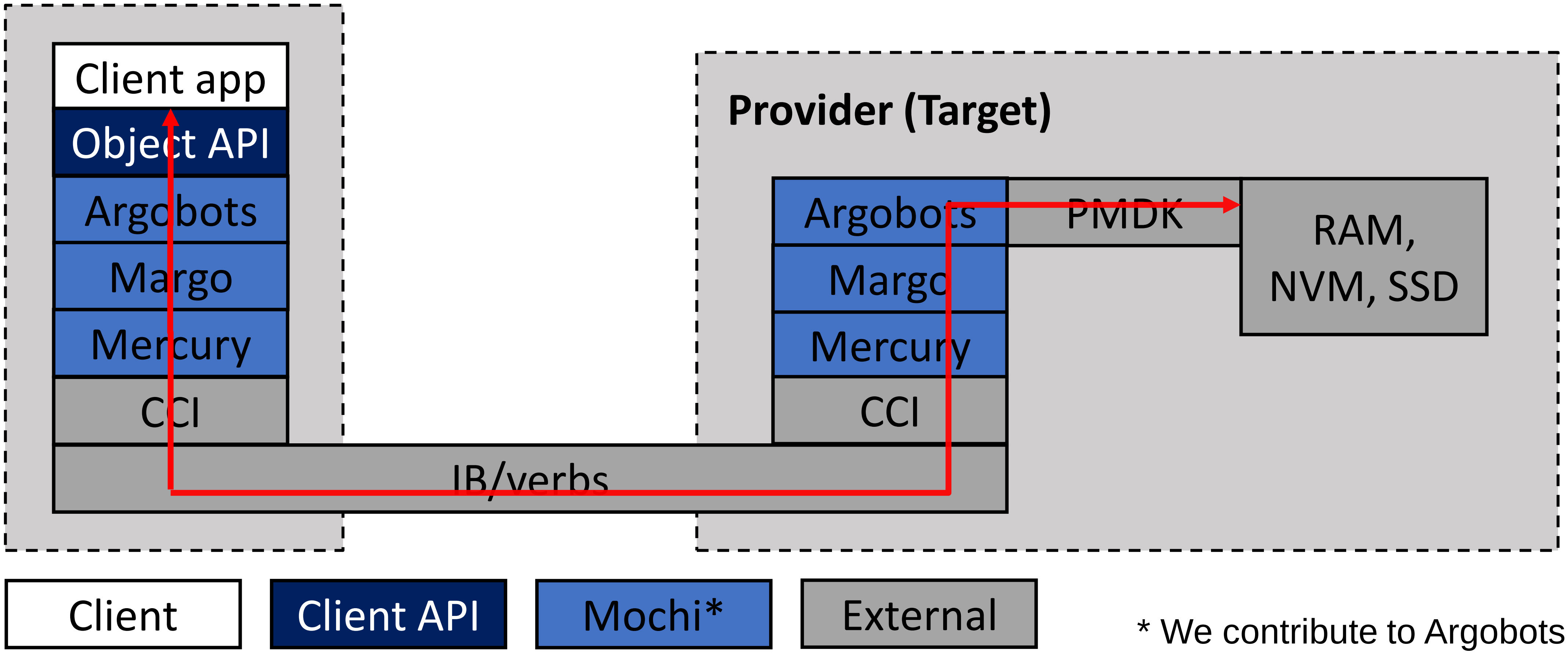
- Multi-threaded model
- Lightweight threads created to handle RPCs
- Express Mercury operations as blocking functions
- Uses Argobots to manage concurrency

https://xgitlab.cels.anl.gov/sds/margo

… and Thallium, if writing in C++!

Microservice Example: BAKE

A Composed Service for Remotely Accessing Objects



* We contribute to Argobots, but it's primarily supported by P. Balaji's team.

P. Carns et al. "Enabling NVM for Data-Intensive Scientific Services." INFLOW 2016, November 2016.

Agenda

Today:

9:00 – 10:00	**Welcome and Introductions**
10:00 – 10:45	**Mochi Landscape**
10:45 – 11:00	**Break**
11:00 – 12:00	**Session 1: Margo and Thallium**
12:00 – 1:15	**Lunch**
1:15 – 2:30	**Session 2: Hands-on: Spack and your first Mochi program**
2:45 – 3:00	**Break**
3:00 – 4:30	**Session 3: SSG and Group Membership**
4:30 – 5:00	**Q&A, Planning**

Wednesday: Components, planning *your* service design, start hacking
Thursday: Performance tuning, porting your service to your target system