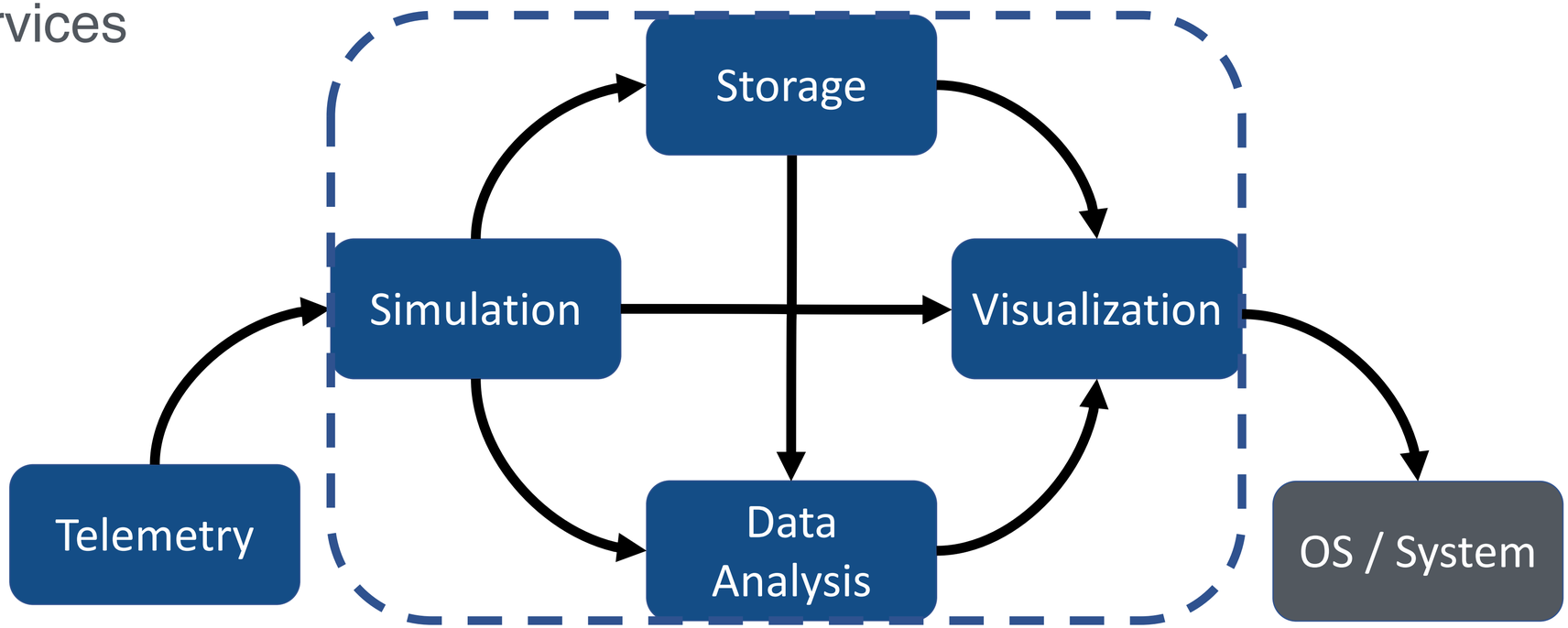
Birds of a Feather: Enabling Data Services for HPC

November 14, 2018

Jerome Soumagne (The HDF Group)
Matthieu Dorier, Philip Carns, Robert Ross (Argonne National Laboratory)
Johann Lombardi (Intel Corporation)
Chad Woods, Kevin Huck (University of Oregon)
Philip Davis, Manish Parashar (Rutgers University)

Introduction

- **Data services become essential to HPC workflow and productivity**
  - Storage services
  - Data analysis and Visualization services
  - Telemetry services
  - etc



Objective

- **Several frameworks already exist and/or are being developed**
  - DAOS / DeltaFS / UnifyCR / Dataspaces / ParaView / Visit / SOS / Faodel

- **Any HPC data service must face similar challenges**
  - Communication between applications
  - Resilience and fault tolerance
  - Deployment
  - Security

- **How can we help service developers/integrators and share knowledge?**

Questions

- **How to deploy data services in HPC?**
  - Scheduler integration
  - Wire-up and bootstrapping
  - Negotiation of resources with end-user application
- **How to handle resiliency?**
  - How can the application recover from a service fault?
- **How is security provided?**
  - Does it matter or just single user?
  - What kind of security do we need?
- **Extras**
  - How to handle communication?
  - How to earn trust from application users?

Format

- **Series of Talks**
  - Mochi and Mercury – Matthieu Dorier (Argonne National Laboratory) and Jerome Soumagne (The HDF Group)
  - DAOS – Johann Lombardi (Intel Corporation)
  - SOS – Chad Wood (University of Oregon)
  - Dataspaces – Philip Davis and Manish Parashar (Rutgers University)

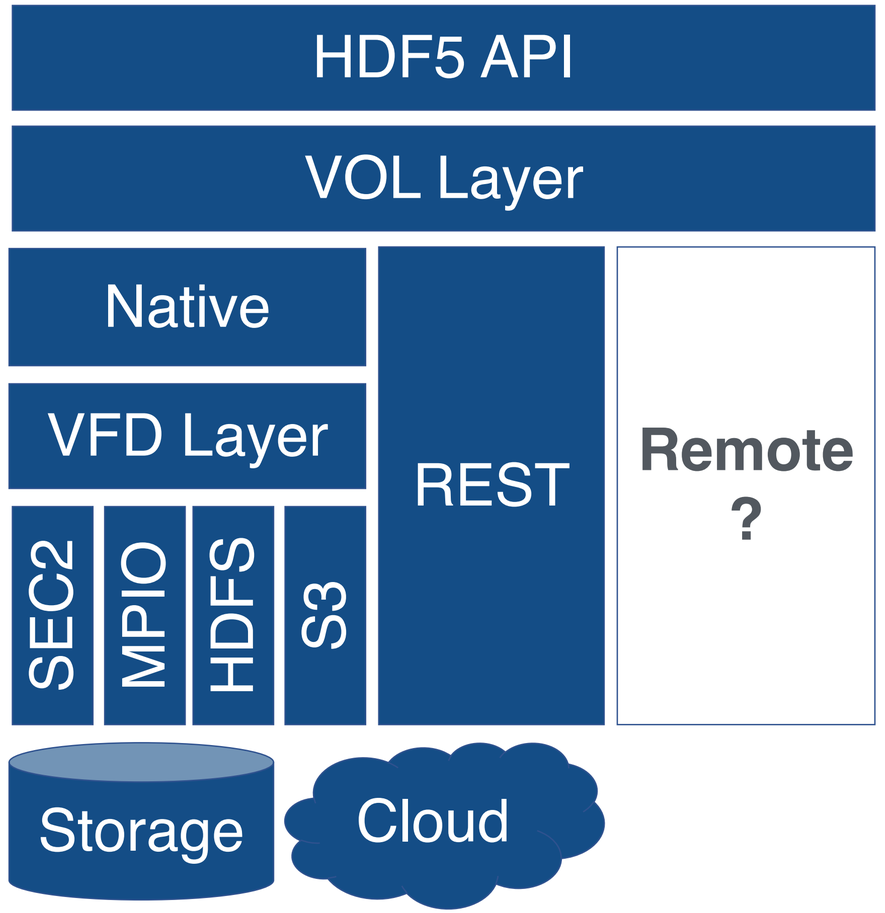
- **Discussion and Q&A**

- **Material will be posted online**
  - http://mercury-hpc.github.io/news/2018/10/17/data-services-bof.html

From Mercury to Mochi

Original Motivation – HDF5

- **Widely used by HPC community**
- **Provide access to storage systems**
  - VFD (through MPI-IO and ROMIO)
  - VOL
- **Access HDF5 files remotely or on separate HPC nodes?**
  - Execute HDF5 calls on remote nodes
  - Remote procedure call



RPC for HPC

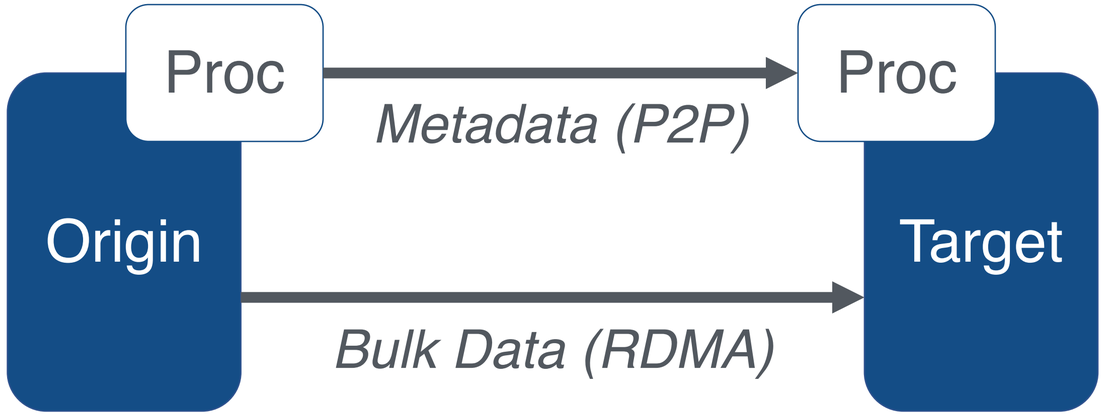
- **RPC (Remote Procedure Call)**
- **Widely used technique to create web services (e.g., gRPC, etc)**
- **Problem: web services frameworks are not designed for HPC**
  - Typically built around TCP/UDP protocols
  - Do not handle large data transfers efficiently
  - Can potentially introduce a lot of jitter (extra threads / memory used / etc)
  - Must be able to run in userspace

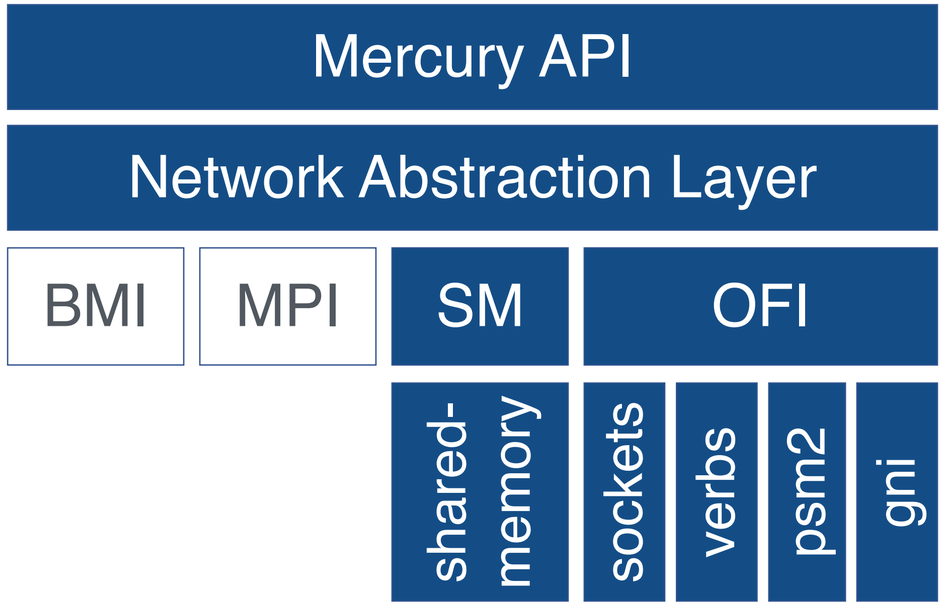
- **Initially developed as part of Exascale FastForward effort w/Intel**
- **Continued through ASCR Mochi project**
- **Mercury (http://mercury-hpc.github.io) / 1.0.0 release yesterday!**

Mercury RPC

- **Data service building block**
  - Origin and Target definitions
  - Input / output arguments = metadata
  - Large data arguments = bulk data

- **Network abstraction layer**
  - Intranode (SM) / internode (OFI)
  - Non-blocking callback-based model





Mercury RPC

- **Deployment**
  - Portability
  - Lightweight / small dependencies
  - Not tied to any threading model
  - Does not provide any notion of group or collectives
  - Require out-of-band mechanism to gather peer information

- **Resiliency**
  - Not resilient on its own but provides building blocks like cancelation

- **Security**
  - Can pass authorization keys down to fabric layer (job granularity)