Department of Computer Science

# BoF: Data-Centric I/O – ISC HPC

**Limitless** Storage
**Limitless** Possibilities

https://hps.vi4io.org

Julian M. Kunkel, Jay Lofstead,
Jean-Thomas Acquaviva

BoF: Data-Centric I/O – ISC HPC

2019-06-18

# BoF: Data-Centric I/O

## Agenda

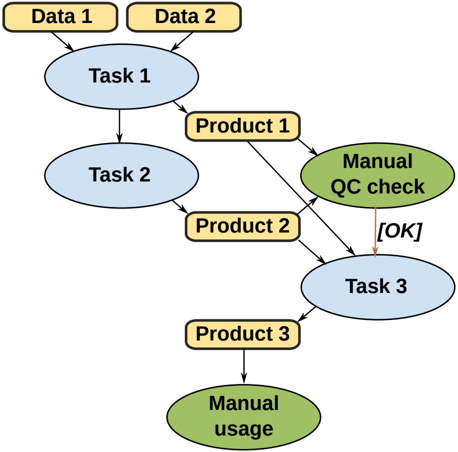
- High-Level Workflows – Potential for Innovation? (10 min)
- Peeking at the current IO stack (2 min)
- Changing Your Archive From a Black Hole to a Gold Mine (10 min)
- Approaches to Programming Extremely Heterogenous Memory Systems (10 min)
- The goldilocks node: getting the RAM just right (5 min)
- NGI initiative: toward a bridge in the semantic gap (10 min)
- The community can make the difference (5 min)
- Discussion

Some parts of the presentations are on purpose a bit provocing.

# Outline

# Workflows

- Consider workflow from 0 to insight
  - Needs/produces data
  - Uses tasks
    - Parallel apps?
    - Big data tools?
    - Manual analysis
  - May need month to complete
  - Manual tasks are unpredictable
  - What are users interested in?
- Not well described in HPC
  - Mostly hardcoded in scripts
- Can we exploit workflows?
  - Does it matter where data is?
  - Vendors simulations?
  - Enforce ILM as needed by users



# Planning HPC Resources

University of Reading

## Planning for Cern/LHC and other big experiments

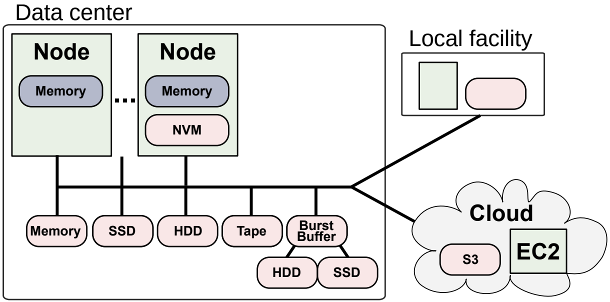
- A detailed planning of activities is performed
- Experiments are proposed with plans (time, resource utilization)

## Planning for Data Centers

- May include: Time needed, CPU (GPU) hours, storage space
- After resources are granted scientists do what they want
  - Some limitations, e.g., quota, compute limit
  - But access patterns?
  - The system is not aware what possibly could happen
  - The data center does not know suffiently what users do
- Additionally: Execution uses often tools with 40year old concepts

# Future Systems: Coexistence of Storage/File Systems

University of Reading




- We shall be able to use all compute/storage technologies concurrently
  - Without explicit migration etc. put data where it fits
  - Administrators just add a new technology (e.g., SSD pool) and users benefit

# Planning HPC Resources: An Alternative Universe

- Scientists deliver
  - detailed but abstract workflow orchestration
  - containers with all software
  - data management plan with data lifecycle
  - time constraints and budget
- Data centers and vendors
  - Simulate the execution before workflow is executed
  - Estimate costs, energy consumption
  - Determine if it is the best option to run
- Systems
  - Utilize the information to orchestrate I/O
  - Make decisions about data location and placement:
    - Trade compute vs. storage and energy/costs vs. runtime
  - Ensure proper execution
- Provocing: Big data is ahead in such an agenda!

# Scenario: Large Simulation

- Assume large scale simulation, timeseries (e.g., 1000 y climate)
- Assume manual data analysis needed (but time consuming)
- We need all 1000 y for detailed analysis!

## A typical workflow execution

- Run simulation for 1000 y
  - Store various data on (online) storage
  - Keep checkpoints to allow reruns
  - Maybe backup data in archive
- Explore data to identify how to analyze data
- At some point: Run the analysis on all data
- Problem: Occupied storage capacity

# Alternative Workflows Done by Scientists

University of Reading

## Recomputation

- Run simulation
  - Store checkpoints
  - Store only selected data (wrt. resolution, section, time)
- Explore data
  - Run recomputation to create needed data (e.g., last year)
- At some point: run analysis across all data needed
- This is a manual process, must consider
  - Runtime parameters
  - System configuration/available resources
  - We are trading compute cycles vs. storage
  - It would be great if a system would consider costs and does this automatically...

# Another Alternative Workflows

## Provided by more intelligent storage and better workflows

- Run simulation
  - Store checkpoints on node-local storage
    - Redundancy: from time to time restart from another node
  - Store selected data on online storage (e.g., 1% of volume)
    - Also store high-resolution data sample (e.g., 1% of volume)
  - Store high-resolution data directly on tape
- Explore data on snapshot
- Month later: schedule analysis of data needed
  - The system retrieves data from tape
  - Performs the scheduled operations on streams while data is pulled in
  - Informs user about analysis progress
- Some people do this manually or use some tools to achieve similarly
  - Aim for domain & platform independence and heterogenous HPC landscapes

# Scenario: Data Organization

University of Reading

## Goal: Semantic Namespace

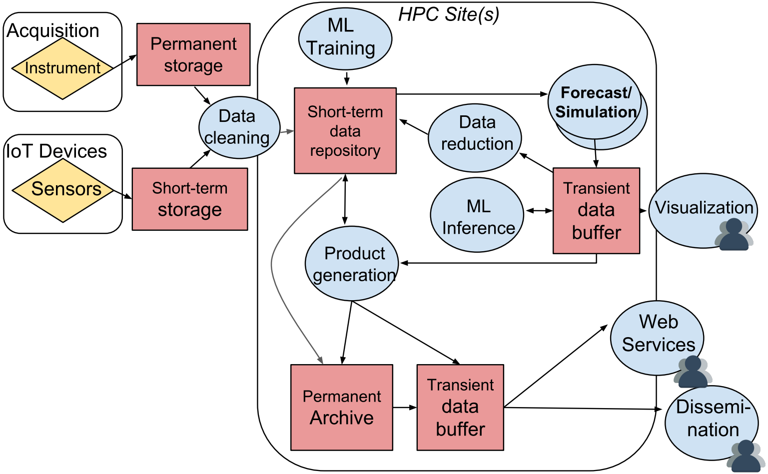
- Provide features of data repositories (e.g., MARS) to explore data
- User-defined properties but provide means to validate schemas
- Similar to MP3 library ...

## High-Level questions addressed by them

- What experiments did I run yesterday?
- Show me the data of experiment X, with parameters Z...
- Cleanup unneeded temporary stuff from experiment X
- Compare the mean temperature of one model for one experiment across model versions

# Smarter Climate/Weather Workflows in 2020+

University of Reading




- IoT (and mobile devices)
  - Additional data provider
  - Improves short-term weather prediction
- Machine learning support
  - Localize known patterns
  - Interactive use Visual analytics
- Data reduction
  - Output is triggered by events (ML)
  - Compress data of ensembles

# Outline

# Example: A Software Stack for NWP/Climate

University of Reading

- Domain semantics
  - XIOS writes independent variables to one file each
  - 2nd servers for performance reasons
- Why user side servers besides data model
  - Performant mappings to files are limited
    - Map data semantics to one "file"
    - File formats are notorious inefficient
  - Domain metadata is treated like normal data
    - Need for higher-level databases
  - Interfaces focus on variables but lack features
    - Workflows
    - Information life cycle management

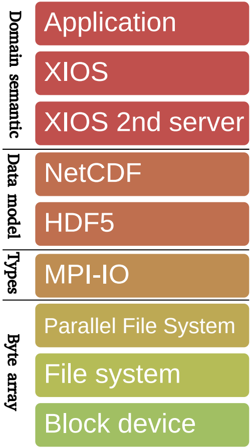



Figure: Typical I/O stack

# Critical Discussion

University of Reading

## Questions from the storage users' perspective

- Why do I have to organize the file format?
  - It's like taking care of the memory layout of C-structs
- Why do I have to convert data between storage paradigms?
  - Big data solutions typically do not require this step!
- Why must I provide system-specific performance hints?
  - It's like telling the compiler to unroll a loop exactly 4 times
- Why is a file system not offering the consistency model I need?
  - My application knows the required level of synchronization

## Being a user, I would rather code an application?

# Challenges Faced by HPC I/O

- Difficulty to analyze behavior and understand performance
  - Unclear access patterns (users, sites)
- Coexistence of access paradigms in workflows
  - File (POSIX, ADIOS, HDF5), SQL, NoSQL
- Semantical information is lost through layers
  - Suboptimal performance, lost opportunities
  - All data treated identically (up to the user)
- Re-implementation of features across stack
  - Unpredictable interactions
  - Wasted resources
- Restricted (performance) portability
  - Optimizing each layer for each system?
  - Users lack technological knowledge for tweaking
- Utilizing the future storage landscapes
  - No performance awareness, manual tuning and mapping to storage needed

# Alternative Software Stack

University of Reading

## Some examples of the zoo of alternatives

- High-level abstractions: Dataclay, Dataspaces, Mochi
- Data models: ADIOS, HDF5, NetCDF, VTK
- Standard API across file formats: Silo, VTK, CDI, HDF5
- Data management tools: iRODS
- Low-level libraries: SIONlib, PLFS
- Storage interfaces: MPI-IO, POSIX, vendor-specific (e.g., CLOVIS), S3, DAOS
- Big-data: HDFS, Spark, Flink, MongoDB, Cassandra
- Projects: EXAHDF, Maestro (FET Proactive)
- Data flow processing: Flink, DeepStream
- Research: Countless new prototypes in that domain every year

# Standardization Attempts

University of Reading

## Promising

- Container storage interface (community driven / involves companies)
- Cloud Data Management Interface (SNIA driven)
- pmem.io (good candidate for persistent memory programming)
- HDF5 (towards a de-facto standard interfaces)

## How about HPC?

- MPI-IO (partially successful)
- Exascale10/EOFS (failed)
- *Various* earlier attempts that failed to make the difference

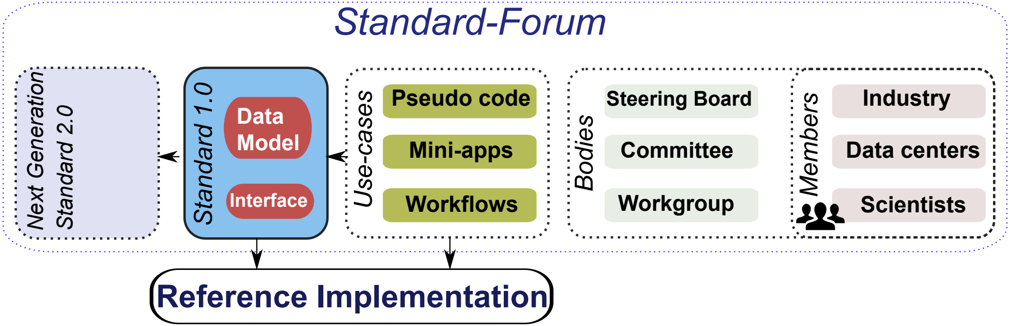
# Outline

# A Potential Approach in the Community: Following MPI

- The **standardization** of a high-level *data model & interface*
  - Targeting data intensive and HPC workloads
  - Lifting semantic access to a new level
  - To have a future: must be beneficial for Big Data + Desktop, too
- Development of a reference implementation of a **smart runtime system**
  - Implementing key features
- Demonstration of benefits on socially relevant data-intense apps

# Development of the Data Model and API

- Establishing a Forum similarly to MPI
- Define data model for HPC
- Open board: encourage community collaboration



# Next Generation Interfaces

University of Reading

## Towards a new data centric compute/IO stack considering:

- Smart hardware and software components
- Storage and compute are covered together
  - **LiquidComputing**: Running pieces on storage, compute, IoT, network, PC
- User metadata and workflows as first-class citizens
- Improving over time (self-learning, hardware upgrades)

## Why do we need a new domain/funding independent API?

- Many domains have similar issues; projects are competitive
- It is a hard problem approached by countless approaches
- Harness RD&E effort across domains

# One Year NGI in Retrospective



- One year ago, we started NGI as informal collaboration
- Fair interest in individuals from
  - Institutions (UCAR, Sandia, Argonne, ...)
  - Vendors (NVIDIA, (Mellanox), Kove, DDN, ...)
- Issue: converting committments into actions (without funding)
- Started to work on white-papers (Minipapers)
  - Use-cases, visions, APIs, coordination
  - Everyone is welcome to contribute
- We will publish the first ones before SC

# Summary

University of Reading

- The separation of concerns in the existing storage stack is suboptimal
- There is a huge potential for the next-generation interface
- Can the community work together to define vision and next generation APIs?

## Participate defining NG interfaces

- Join the mailing list / Slack
- Visit: `https://ngi.vi4io.org`

# Appendix

# Outline

# A Pragmatic View

- Take existing data model like VTK (or NetCDF) as baseline
- With a hint of:
  - Scientific metadata handling
  - Workflow and processing interface
  - Information lifecycle management
  - Hardware model interface (hardware provides its own performance models)
- First prototype utilizes existing software stack
  - Like Cylc for workflows
  - Like MongoDB for metadata
  - Like a parallel file system (or object storage)
- Work on:
  - Scheduler for performant mapping of data/compute to storage/compute
  - A FUSE client for flexible data mappings on semantic metadata
  - Importer/Exporter tools for standard file formats
- Add magic (knowledge of experts developing APIs)
- Next prototype: move on with true implementation

# Next-Generation HPC IO API Key Features

- High-level data model for HPC
  - Storage understands data structures vs. byte array
  - Relaxed consistency
- Semantic namespace and storage-aware data formats
  - Organize based on domain-specific metadata (instead of file system)
  - Support domain-specific operations and addressing schemes
- Integrated processing capabilities
  - Offload data-intensive compute to storage system
  - Managed data-driven workflows supporting events and services
  - Scheduler maps compute and I/O to hardware
- Enhanced data management features
  - Information life-cycle management (and value of data)
  - Embedded performance analysis
  - Resilience, import/export, ...