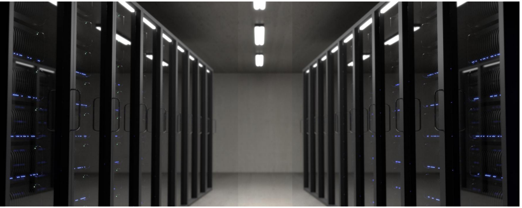
GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

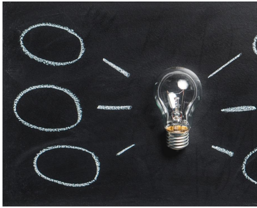
Institute for Computer Science / GWDG

HPS
https://hps.vi4io.org

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

BoF: Data-Centric Computing for the Next Generation

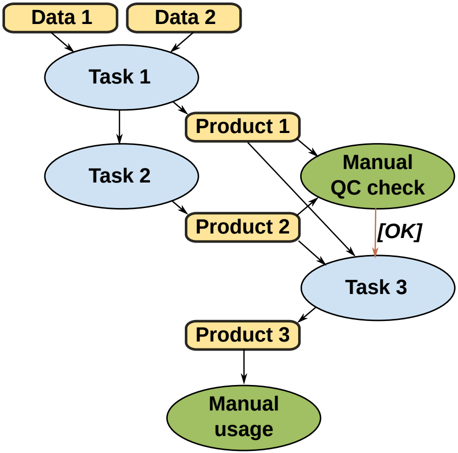




Outline

Workflows

- Consider workflow from 0 to insight
  - Needs/produces data
  - Uses tasks
    - HPC and big data tools
    - Manual analysis
  - May need months to complete
  - Manual tasks are unpredictable
  - What are users interested in?
- Not well described in HPC
  - Mostly hardcoded in scripts
- Can we technically exploit workflow knowledge?
  - Abstract locality data/compute
  - Allow system optimization
  - Enforce user policies (e.g., ILM)



Planning HPC Resources

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

- Proposals include: Time needed, CPU (GPU) hours, storage space
- After resources are granted scientists (basically) do what they want
  - Some limitations, e.g., quota, compute limit
  - But workflows and observable I/O 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

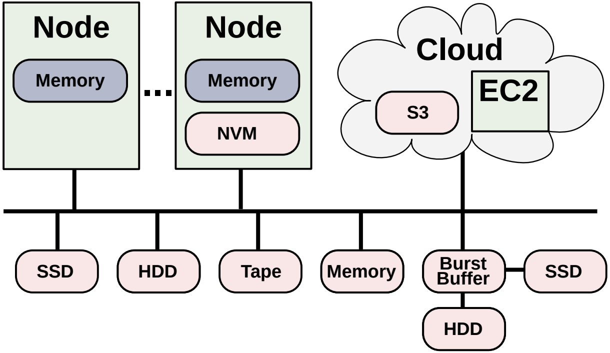
Critical Discussion

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?

Future Systems: Coexistence of Storage/File Systems



- We shall be able to use all compute/storage technologies concurrently
  - Without explicit migration etc. put data where it fits, compute where sensible
  - Administrators just add a new technology (e.g., hybrid) 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
  - Determine the best option to run
  - Estimate costs, energy consumption
- 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
- 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)
- Scientists need all 1000 y for detailed analysis!

A typical workflow execution

- Run the 1000 y simulation split into jobs
  - 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

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 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
- Months 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
  - Should aim for domain/platform independence and heterogenous landscapes

Scenario: Data Organization

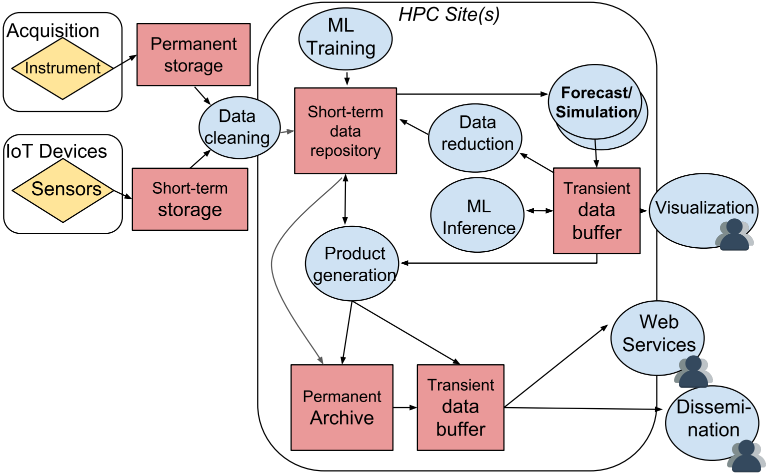
High-Level questions addressed by (scientific) users

- 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

Goal: Semantic Namespace

- Provide features of data repositories to explore data
- User-defined properties but provide means to validate schemas
- Similar to MP3 library ...

Example: Smarter Climate/Weather Workflows in 2020+



- 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

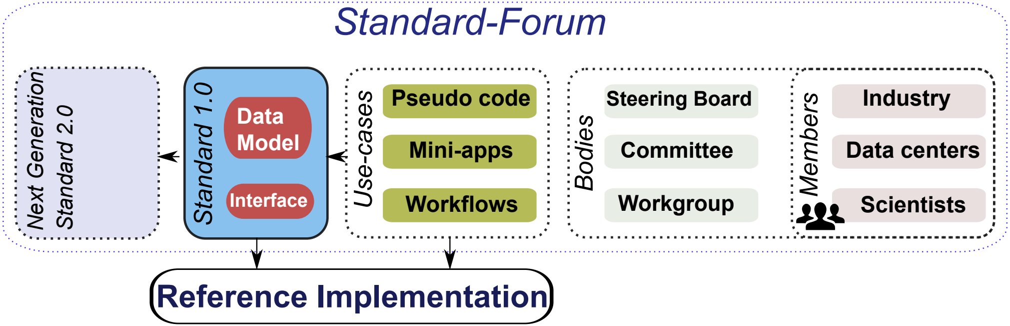
Outline

A Potential Approach in the Community: Following MPI

- **Standardization** of a high-level *data model & interface* & workflow spec
  - 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

Towards a new data centric compute/IO stack considering:

- Smart hardware and software components
- Storage and compute are covered together
  - **Liquid Computing**: Running workflow fragments 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

Summary

- 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 gen- APIs?

Participate defining NG interfaces

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

Discussion

Appendix

Outline

Example: A Software Stack for NWP/Climate

- 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

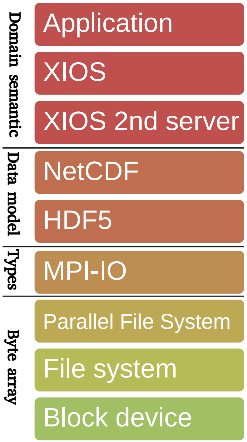


Figure: Typical I/O stack

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

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

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