Department of Computer Science

Toward Next Generation Interfaces for Exploiting Workflows

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

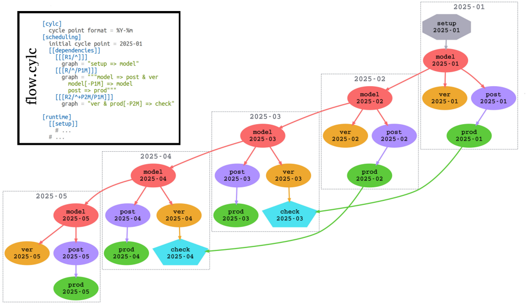
Julian M. Kunkel, thanks to: Luciana Pedro, Bryan Lawrence, Glenn Greed, David Matthews, Hua Huang

2020-04-23

Climate/Weather Workflows

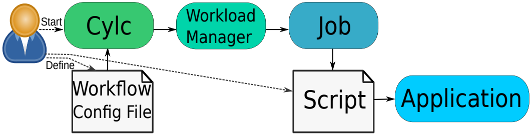
University of Reading

- A workflow consists of many steps
  - Repeated for simulation time
  - E.g., weather for 14 days
- Scientists use **Cylc** to handle such **cycling** workflows
- Cylc workflow specifies
  - Tasks with commands
  - Environment variables
  - Dependencies



Workflow Execution

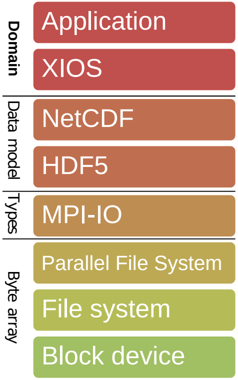
1. Cylc analyzes workflow
   - Creates a job script for each task
   - Submits to workload manager
2. Wflow manager allocates resources
   - Starts a job with env. vars
3. Job script runs applications
   - File names set by
     - env. var
     - command
   - May depend on cycle



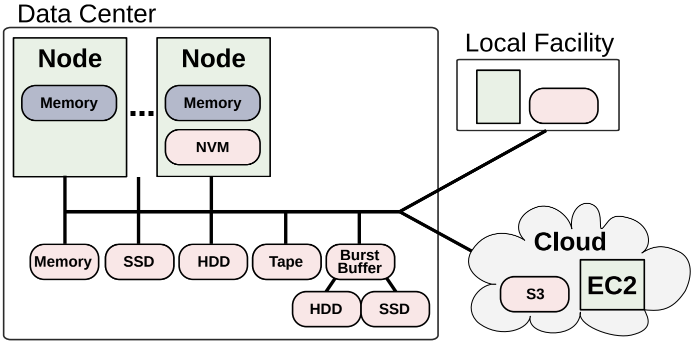
- The data dependency between tasks is currently stored implicitly

Environments of Applications in HPC

University of Reading



I/O path for an MPI-parallel application. HDF5 can be replaced with ESDM.



Example of an heterogeneous HPC landscape

Outline

Data Center Perspective: Utilization of HPC Resources

University of Reading

Projects run in Data Centers

- Proposals may 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 actual usage and 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

Projects executed in Cern/LHC and other big experiments

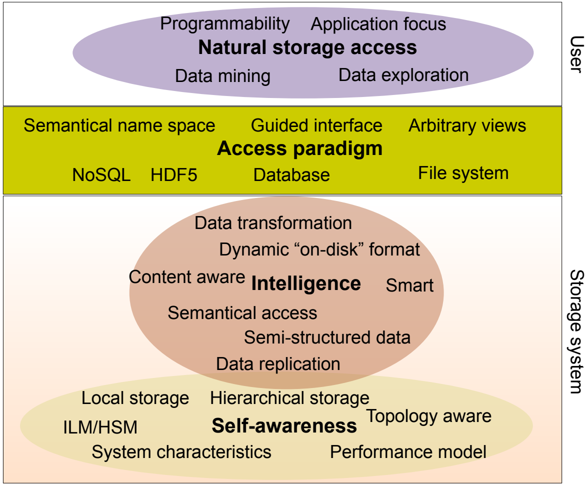
- A detailed planning of activities is performed
- Experiments are proposed with detailed plans (time, resource utilization)

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 AND computation
  - Make decisions about data location and placement:
    - Trade compute vs. storage and energy/costs vs. runtime
  - Ensure proper execution
- Provoking: Big data technology is ahead of HPC in such an agenda

Personal Vision: Towards Intelligent Storage Systems and Interfaces

University of Reading



- Abstract data interfaces
- Enhanced data management
- Integrated compute/storage
- Flexible views on data
- Smart hardware/storage
  - Self-aware systems
  - AI optimized placement
  - Bring-your-own-behavior
- Cross sites and cloud

Vision: Exploit Workflow Knowledge

University of Reading

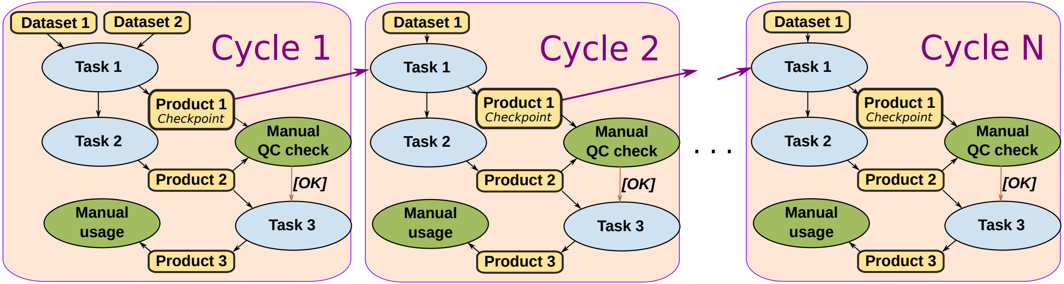
- Enhance workflow description (for climate/weather) with IO characteristics
  - Needed input
  - Generated output and its characteristics
  - Information Lifecycle (data life)
    - How long to keep data, type of data...
  - ⇒ Explicit input/output definition (dependencies) instead of implicit
- Smarter IO scheduling
  - Considering the hardware/software environment
  - Data placement: Transfer, migration, staging, replication, allocation
  - Data reduction: data compression and data recomputation

⇒ Providing a separation of concern

- Scientist declares workflow including IO
- System maps workflow to hardware using expert knowledge and ML

Extended Workflow Description

University of Reading



- Enhance workflow description with IO characteristics
  - Input required
  - Output generated and its characteristics

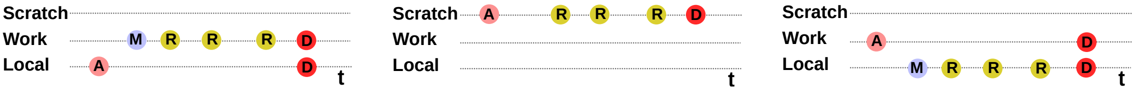
Smarter IO Scheduling: Advantage for Data Placement

University of Reading

Scenario

- Consider three file systems: local, scratch, and work
  - Local is a compute-node local storage system
- Data can be stored on any of these storage systems
- Scheduler to optimize data placement throughout life cycle to hardware
- Optimally: scheduler to optimize computation of data-driven workflow too

Alternative life cycles for mapping a dataset (Selection)



Local and work file systems | Scratch file system only | Local and work file systems

Allocation, **M**igration, **R**eading, and **D**eleting

Outline

University of Reading

Earth-System Data Middleware

University of Reading

- Part of the ESiWACE Center of Excellence in H2020
  - Centre of Excellence in Simulation of Weather and Climate in Europe

  https://www.esiwace.eu
- Integrated as NetCDF backend

ESDM provides a transitional approach towards a vision for I/O addressing

- Scalable data management practice
- The inhomogeneous storage stack
- Suboptimal performance and performance portability
- Data conversion/merging

Optimal IO in an Application with ESDM

University of Reading

Example Application IO

- Compute timestep (N times)
- IO timestep: Start IO after computation (asynchronously)
  - This requires one additional memory read (even remotely)

Application IO with ESDM Streaming

- Compute timestep (N-1 times)
- IO timestep mix compute and IO
  - Append data to a buffer once it is computed
  - Perform transformations in-flight
  - Execute IO once buffer is sufficiently big

Example from a Shallow Water Model

University of Reading

- Stores data column-wise in memory
- Keep existing separation of compute phase and IO phase for now[1]

Existing NetCDF code for IO phase

```
size_t start[] = {0, 0};
size_t count[] = {nY, 1};
for(unsigned int col = 0; col < nX; col++) {
  start[1] = col; //select col (dim "x")
  nc_put_vara_float(dataFile, i_ncVariable, start, count,
    &i_matrix[col+boundarySize[0]][boundarySize[2]]);
}
```

[1] DSLs will help to separate those phases

ESDM Code for the Application

University of Reading

```c
int64_t offset[] = {(int64_t) timeStep, offsetY, offsetX};
int64_t size[] = {1, (int64_t) nY, (int64_t) nX};

esdm_status ret;
esdm_write_request_t ew;
ret = esdm_write_req_start(& ew, dset, size, offset);
checkRet(ret);
for(int y = 0; y < nY; y++){
  for(int x = 0; x < nX; x++){
    esdm_write_req_pack_float(ew,
      i_matrix[x + boundarySize[0]][boundarySize[2] + y]);
    // note that this may trigger an actual IO as well and postprocessing
  }
}
ret = esdm_write_req_commit(& ew);
```

Outline

Design Overview for Workflow Extensions

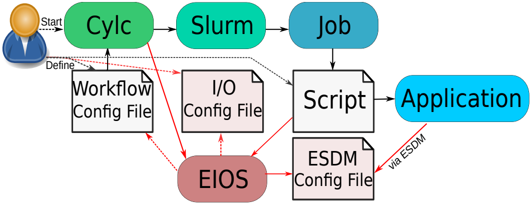
University of Reading

Relevant components

- Configuring system information
- Extending the workflow description
- Providing a smart I/O scheduler (EIOS)

Modified workflow execution

1. Cylc analyzes workflow
   - EIOS provides Slurm variables
2. Wflow manager allocates resources
   - May schedule on nodes of prev. jobs
3. Job script runs applications
   - EIOS generates pseudo filenames encoding scheduling information



Configuring System Information

- Reuse the Earth-System Data Middleware (ESDM) configuration file
  - Contains available storage targets, performance model, further information
  - We will be extending the performance model but how to describe storage best?

```
"backends": [
    {"type": "POSIX", "id": "work1", "target": "/work/lustre01/projectX/",
        "performance-model" : {"latency" : 0.00001, "throughput" : 500000.0},
        "max-threads-per-node" : 8,
        "max-fragment-size" : 104857600,
        "max-global-threads" : 200,
        "accessibility" : "global"
    },
    {"type": "POSIX", "id": "work2", "target": "/work/lustre02/projectX/",
        "performance-model" : {"latency" : 0.00001, "throughput" : 200000.0},
        "max-threads-per-node" : 8,
        "max-fragment-size" : 104857600,
        "max-global-threads" : 200,
        "accessibility" : "global"
    },
    {"type": "POSIX", "id": "tmp", "target": "/tmp/esdm/",
        "performance-model" : {"latency" : 0.00001, "throughput" : 200.0},
        "max-threads-per-node" : 0,
        "max-fragment-size" : 10485760,
        "max-global-threads" : 0,
        "accessibility" : "local"
    }
] ...
```

Extending Workflow Description

- Additional IO workflow file (later to be integrated)
- EIOS knows workflow from Cylc and reads this file

```
[Task 1]
  [[inputs]]
    topography = "/pool/input/app/config/topography.dat"
    checkpoint = "[Task 1].checkpoint$(CYCLE - 1)"
    init       = "/pool/input/app/config/init.dat"
  [[outputs]]
    [[[varA]]] # This is the name of the variable
      pattern = 1 day
      lifetime = 5 years
      type = product
      datatype = float
      size = 100 GB
      precision.absolute_tolerance = 0.1
    [[[checkpoint]]]
      pattern = $(CYCLE)
      lifetime = 7 days
      type = checkpoint
      datatype = float
      dimension = (100,100,100,50)
```

- What information to integrate is not yet perfectly clear

Smarter I/O Scheduler

University of Reading

- Provides hints for colocating tasks with data
  - Create dummy file name to include schedule (e.g., prefer local storage)
  - ESDM parses the schedule information and enacts it (if possible)
- Optimizing data placement strategy in ESDM/workflow scheduler
  - Utilizing hints for IME to pin data to cache
  - Storing data locally between depending tasks (using modified Slurm)
  - Optimizing initial data allocation (e.g., alternating storage between cycles)

These changes are planned as part of the ESiWACE project

- Relevant for climate/weather applications and achievable now
- Considered to be intermediate and leading towards the vision

Outline

Next Generation Interfaces: Community Approach

Goal a new data-driven compute/storage and NGI

- Workflows and metadata as first-class citizens
- Storage and compute for heterogenous environments
- Smart software (and hardware) instead of manual
- Improving over time (self-aware/learning)
- Standardized interfaces beyond POSIX/Spark/Dask/...

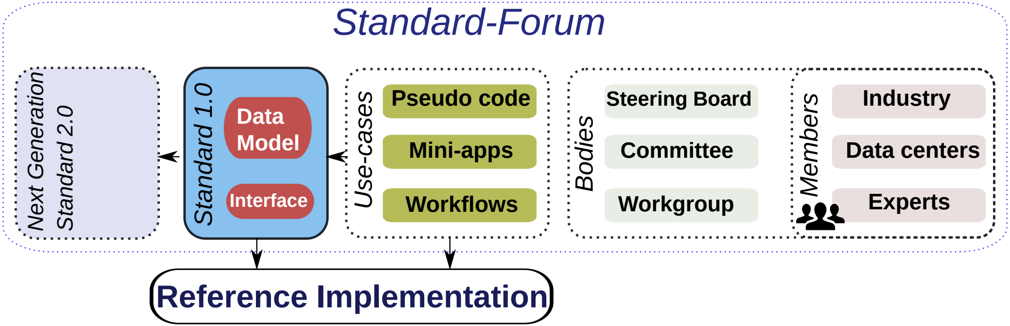
Why do we need a new domain-independent API?

- Other domains have similar issues: Harness RD&E effort across domains
- It is a hard problem approached by countless approaches
- Existing approaches address only a subset of the problems

Community-Driven Development of Data Model & API

University of Reading

- Establishing a Forum (similarly to MPI)
- Model targets High-Performance Computing and data-intensive compute
- Open board: encourage community collaboration



Project Plan

Approach

- Development of a set of whitepapers
  - Limitations of the current state of practice
  - A vision for scientific computing in data centers in 2025+
  - Vision for next generation interfaces
  - Selected use-cases and community approach
- Get more community on board

Status

- Coarse paper exists since Dec/2018
  - Contributions from individuals
  - Splitted the paper into simpler subpapers, release pending
- Experience: Difficult to pull community together
  - Commitment of individuals is
- Good news: BoF for ISC'20 was accepted (ISC will be virtual though...)

Summary and Conclusions

University of Reading

Goals of our vision and design

- Separation of concerns between developer/user and system optimization
  - Scientists enhances workflow descriptions with IO characteristics
  - System exploits workflow specification considering system characteristics
- IO middleware orchestrates computation of post-processing effectively
  - ESDM post-processing part of ESiWACE2 project

Outlook: Opportunities Knowing Workflows

- Performance modelling (simulation or via. recorded behavior)
  - Imagine to include compute model, too
  - Analyse: How long will the workflow run, costs to run it on a given platform?
  - What if analysis: How to change the system / storage to improve performance?
- Data centers may require submission of workflow descriptions for proposals
  - Data center could predict benefit, costs, explore how to run it optimally
  - May hand over to vendors, explore signposting to alternative systems