A Peek into Workflow I/O

Jakob Lüttgau, Shane Snyder, Phil Carns, Justin M. Wozniak, Julian Kunkel, Thomas Ludwig

BoF Analyzing Parallel I/O, SC'18
November 13, 2018 / Dallas, TX

DKRZ
DEUTSCHES KLIMARECHENZENTRUM

Argonne
NATIONAL LABORATORY

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

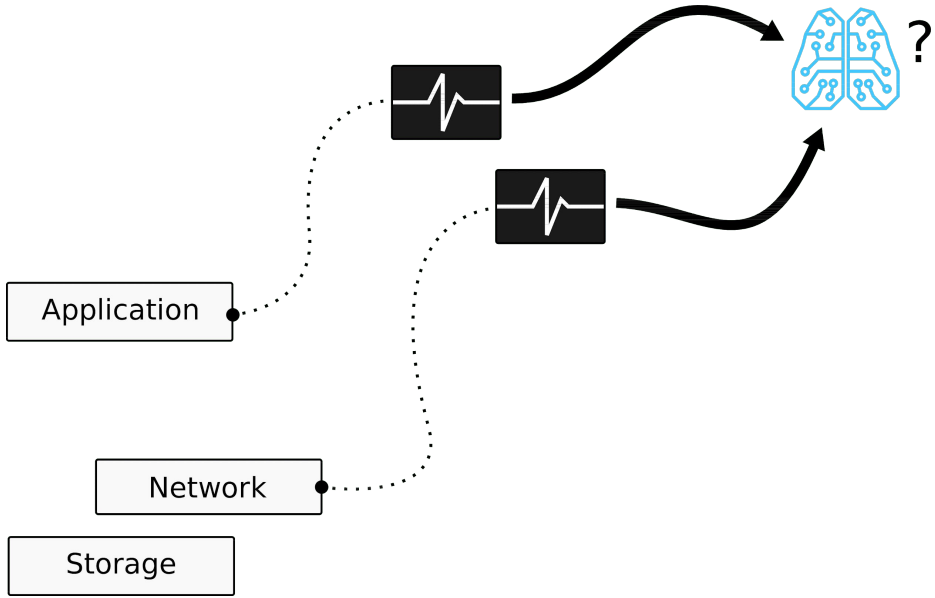
Workflows and HPC

Workflows offer …

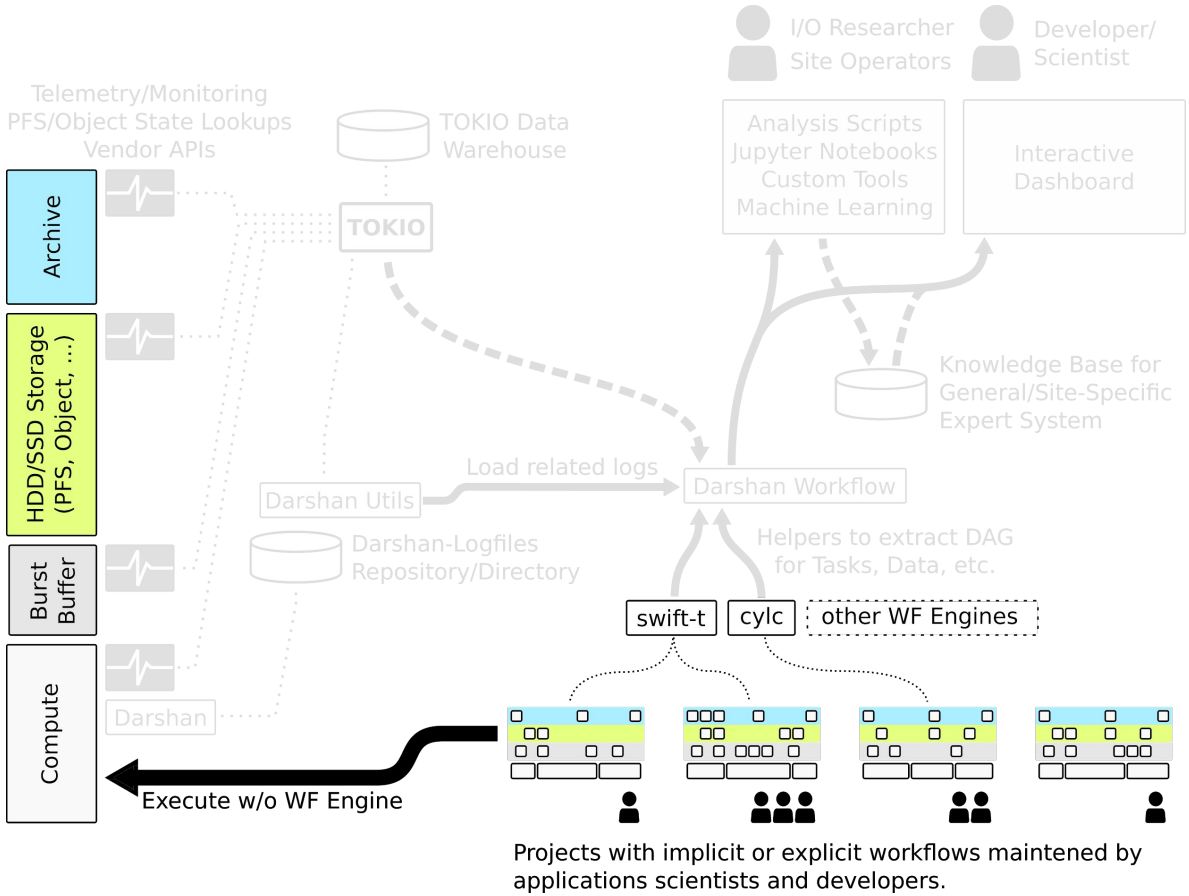
- … anticipatable future activity
- … implicit intent to be discovered
- … explicit intent description

How can we better understand and react to I/O behavior of workflows in HPC?

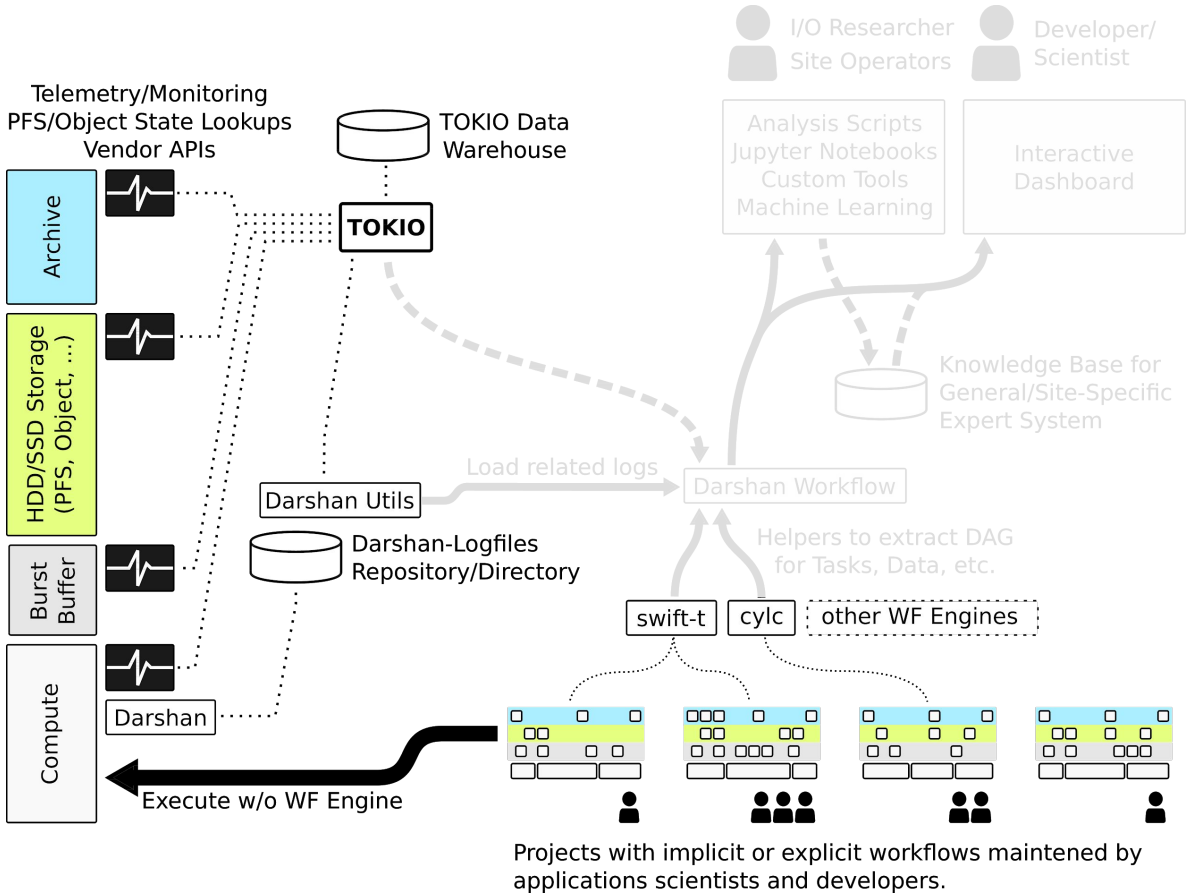
- Explore convenient toolchain for researchers and site operators.
- Considerations for user facing components to improve communicating advice.



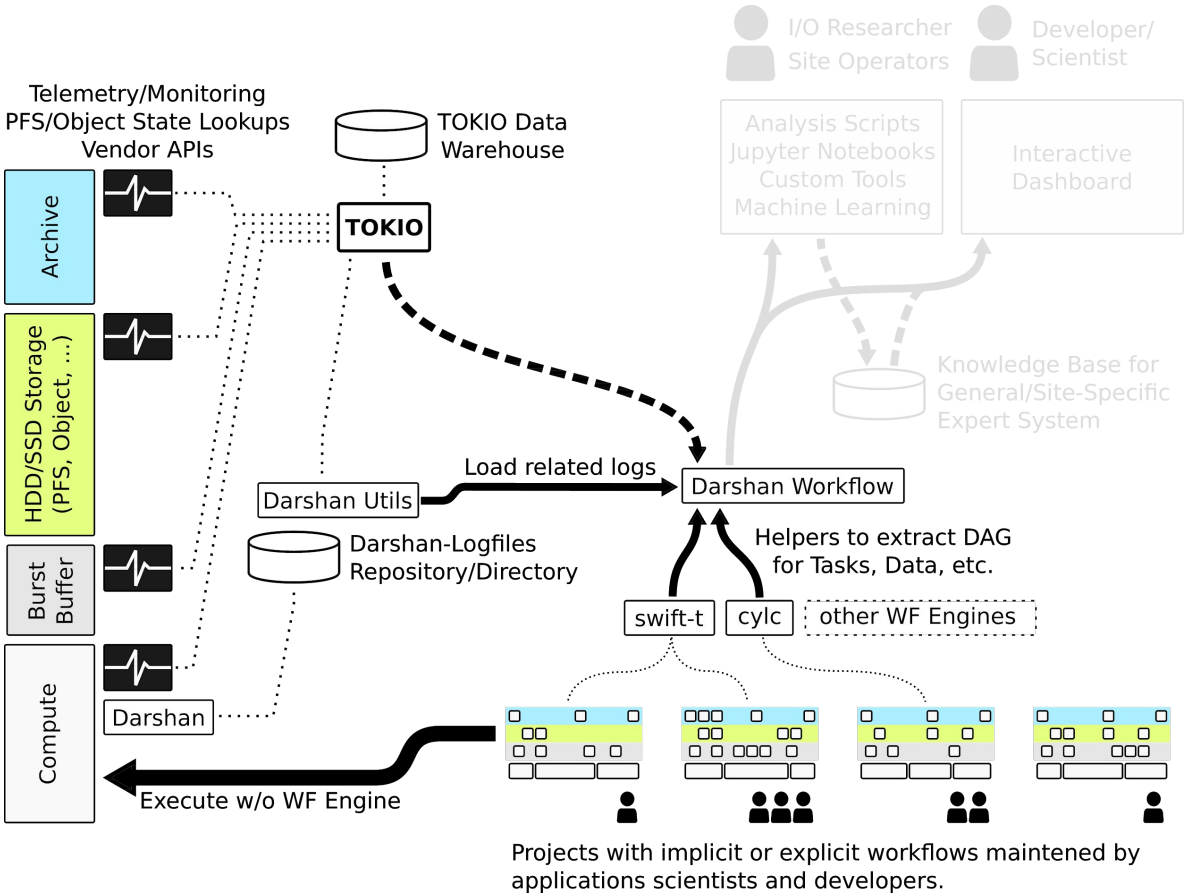
Architecture for Augmenting I/O in Workflows



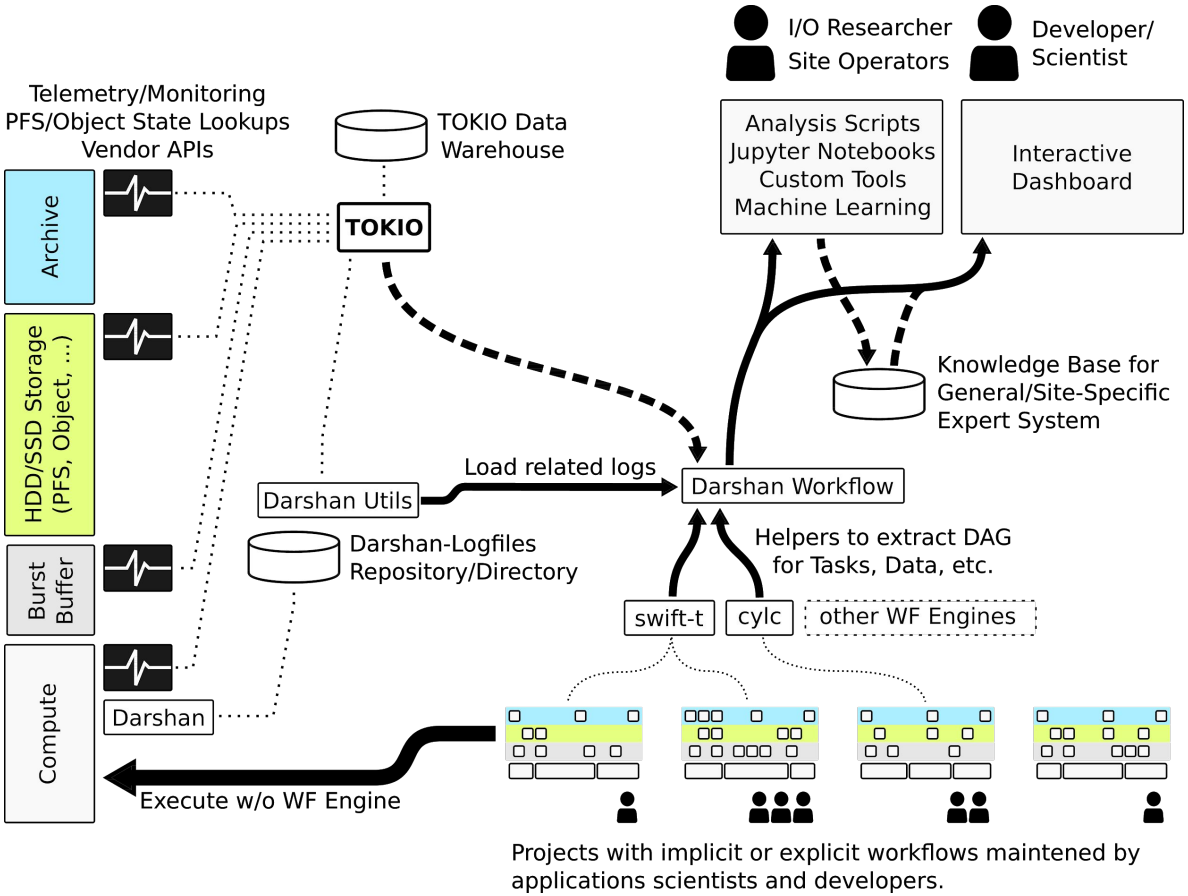
Architecture for Augmenting I/O in Workflows



Architecture for Augmenting I/O in Workflows

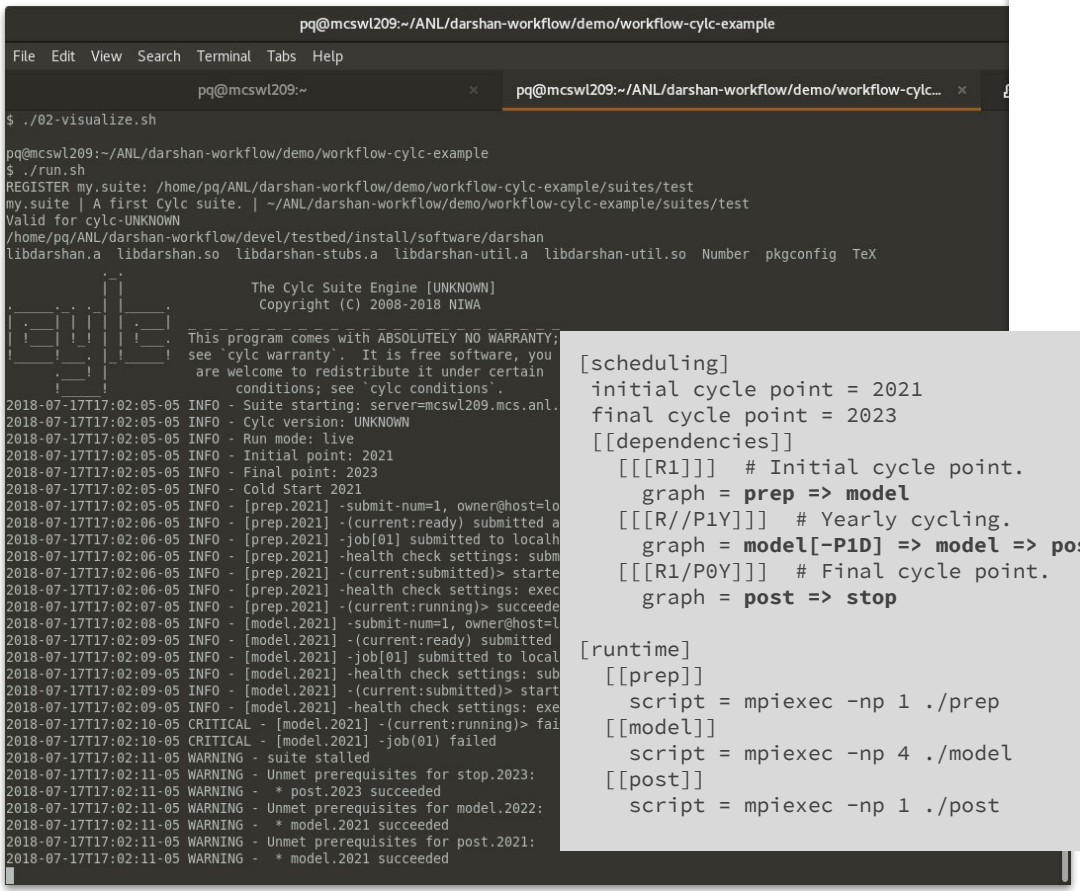


Architecture for Augmenting I/O in Workflows



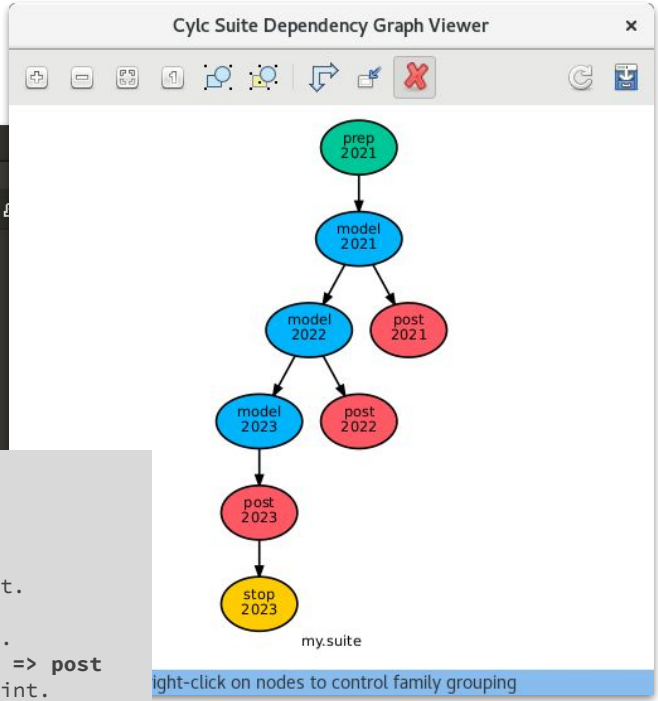
Data-Intensive Exascale Workflow: Climate Modeling

ICON is a climate model used by Researchers at Max-Planck and by the German Weather Service (DWD).
CDO is a pre/post-processing tool (climate operators) for NetCDF files.
ParaView is a popular visualisation toolkit built on top of VTK.




```
[scheduling]
 initial cycle point = 2021
 final cycle point = 2023
 [[dependencies]]
   [[[R1]]]  # Initial cycle point.
     graph = prep => model
   [[[R//P1Y]]]  # Yearly cycling.
     graph = model[-P1D] => model => post
   [[[R1/P0Y]]]  # Final cycle point.
     graph = post => stop

[runtime]
  [[prep]]
    script = mpiexec -np 1 ./prep
  [[model]]
    script = mpiexec -np 4 ./model
  [[post]]
    script = mpiexec -np 1 ./post
```



https://cylc.github.io/cylc/

Augmenting Workflow I/O with Darshan/TOKIO

Scientific discovery increasingly depends on complex workflows consisting of multiple phases and sometimes millions of parallelizeable tasks or pipelines. Typical workflows on HPC systems routinely require the pre-processing, generation by simulation and post-processing of data. Unfortunately, most workflow models focus on the scheduling and allocation of resources for tasks while the impact on storage systems remains a secondary objective.

By combining a workflow description (e.g., from a workflow engine like Swift or Cylc) with log data or telemetry information we can gain insight on the I/O behaivior of a complete workflow and then optimize applications, middleware and systems accordingly.

In [1]:
```
import darshan.workflow
```

In [2]:
```
# Load a workflow report
wf = darshan.workflow.load('data-workflow.json')
wf.data
```

Out[2]:

```
{'nodes': [{'id': 'model.2021',
   'label': 'model.2021',
   'x': 140.5,
   'y': -410.0,
   'data': {},
   'group': 'report'},
  {'id': 'model.2022',
   'label': 'model.2022',
   'x': 91.5,
   'y': -320.0,
   'data': {},
   'group': 'report'},
  {'id': 'post.2021',
   'label': 'post.2021',
   'x': 189.5,
   'y': -320.0,
   'data': {}},
  {'id': 'model.2023',
   'label': 'model.2023',
   'x': 42.5,
```


Inspecting a Workflow from Darshan

Darshan/TOKIO-workflow is build to ease interactive and automatic analysis of workflows with a focus on the I/O perspective. As such it explores a variety of convenience methods and common visualisations such as the `task_summary()` and `show_graph()` methods.

In [3]:
```
wf.tasks_summary()
```

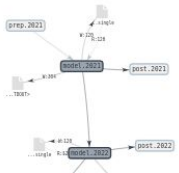
```
model.2021      Records: 2      Layers: POSIX, STDIO
model.2022      Records: 2      Layers: POSIX, STDIO
post.2021       Records: 0      Layers:
model.2023      Records: 2      Layers: POSIX, STDIO
post.2022       Records: 0      Layers:
post.2023       Records: 0      Layers:
stop.2023       Records: 0      Layers:
prep.2021       Records: 0      Layers:
```

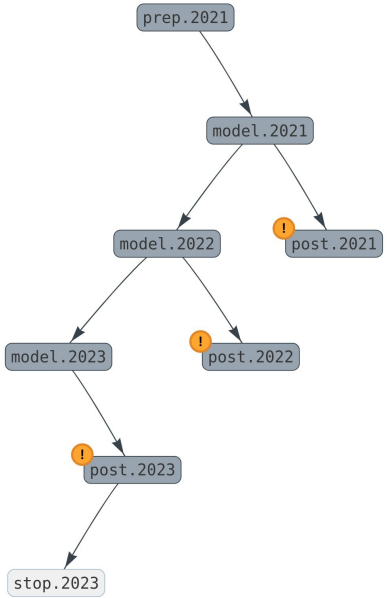
Displaying and Interacting with Workflow

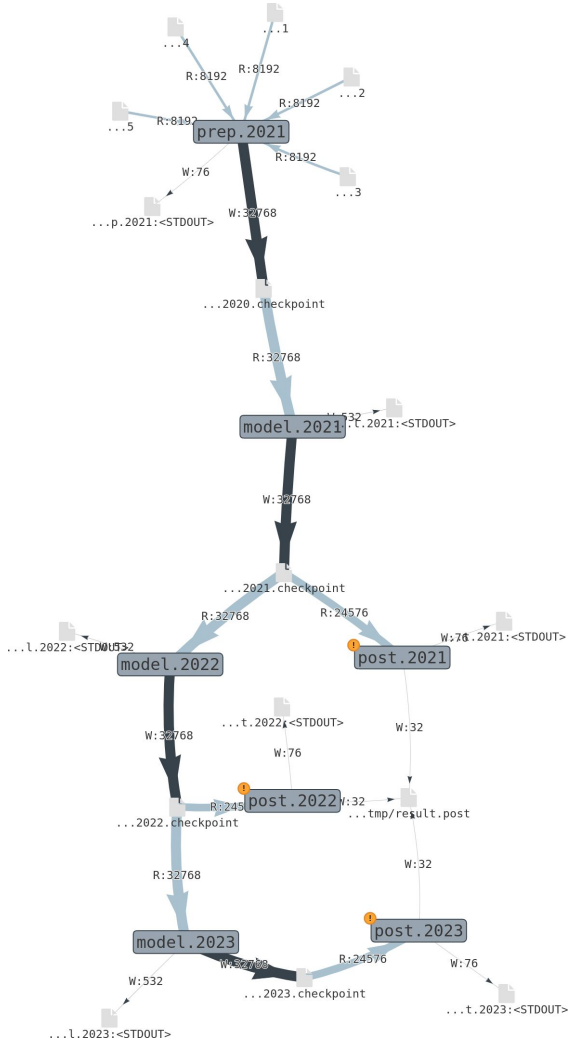
Jupyter Notebooks can be extended with custom widget, which allows to turn them into versatile tools for custom but powerful tools in I/O analysis. This is especially true for workflows, which generate a lot of log data so that interactive tools make data exploration more convienient.

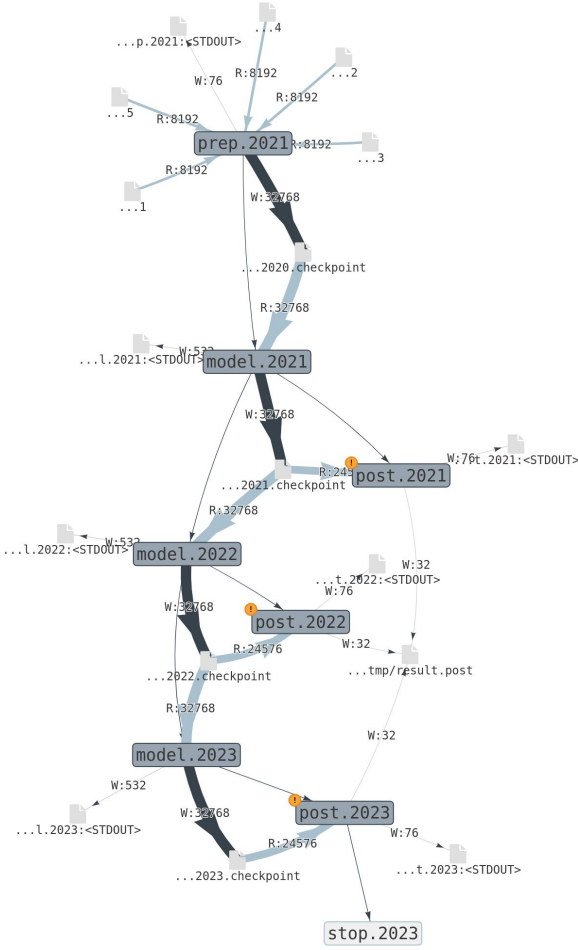
In [4]:
```
import darshan.ipywidgets.example as wfgraph
import json
wf.show_graph()
graph = wfgraph.HelloWorld()
graph
```

```
TODO: GRAPH IpythonWidget
```




prep.2021
model.2021
model.2022
post.2021
model.2023
post.2022
post.2023
stop.2023


...4
...1
R:8192
R:8192
...2
R:8192
...5
R:8192
prep.2021
R:8192
...3
W:76
...p.2021:<STDOUT>
W:32768
...2020.checkpoint
R:32768
model.2021
...t.2021:<STDOUT>
W:32768
...2021.checkpoint
R:32768
R:24576
...l.2022:<STDOUT>
model.2022
post.2021
...t.2021:<STDOUT>
...t.2022:<STDOUT>
W:32768
W:32
W:76
post.2022
...2022.checkpoint
...tmp/result.post
R:32768
W:32
model.2023
post.2023
W:532
R:24576
W:76
...2023.checkpoint
...l.2023:<STDOUT>
...t.2023:<STDOUT>

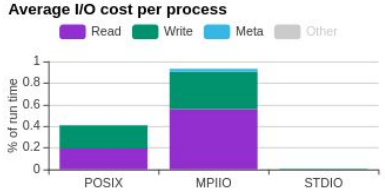

...p.2021:<STDOUT>
...4
W:76
R:8192
...2
R:8192
...5
R:8192
prep.2021
R:8192
...3
R:8192
...1
W:32768
...2020.checkpoint
R:32768
...l.2021:<STDOUT>
model.2021
W:32768
post.2021
...t.2021:<STDOUT>
...2021.checkpoint
R:32768
...l.2022:<STDOUT>
model.2022
...t.2022:<STDOUT>
W:32
W:76
W:32768
post.2022
R:24576
W:32
...2022.checkpoint
...tmp/result.post
R:32768
W:32
model.2023
W:532
post.2023
...l.2023:<STDOUT>
W:32768
W:76
R:24576
...2023.checkpoint
...t.2023:<STDOUT>
stop.2023

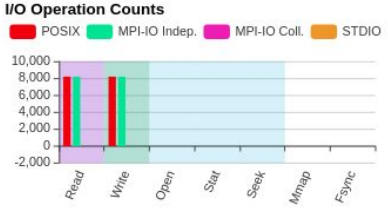
http://my.datacenter/workflow-io?worfklow_id=314159

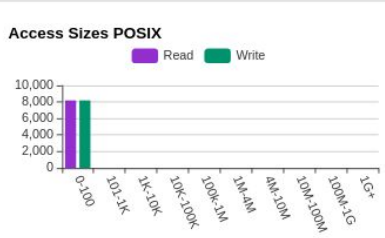
Workflow Overview

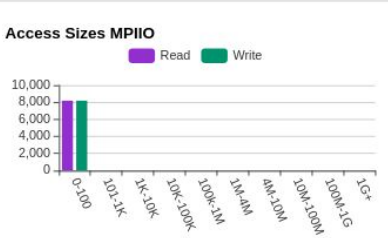
/home/pq/ANL/darshan-workflow/demo/workflow-cylc-example/model

jobid: 19911	uid: 1000	nproc: 4	runtime: 1 seconds









Most Common Access Sizes

(POSIX and MPI-IO)

Layer	Access Size	Count
POSIX	50013	14
POSIX	50007	14
POSIX	49986	9
POSIX	49998	9
MPI-IO	1020	212
MPI-IO	512	34

File Count Summary

(estimated by POSIX I/O access offsets)

type	number of files	avg. size	max size
total opened	4	1.8M	7.2M
read-only files	0	0	0
write-only files	1	16K	16K
read/write files	3	2.4M	7.2M
created files	4	1.8M	7.2M

Logfiles:

Task: model.2021
model.2021: darshan-lo
Task: model.2022
model.2022: darshan-lo
Task: post.2021
post.2021: darshan-logs
Task: model.2023
model.2023: darshan-lo
Task: post.2022
post.2022: darshan-logs
Task: post.2023
post.2023: darshan-logs
Task: prep.2021
prep.2021: darshan-logs
Files
model.2021: /tmp/2020.
model.2021: /tmp/2021.
model.2021: <STDOUT:
model.2022: /tmp/2022.
model.2022: /tmp/2021.
model.2022: <STDOUT:
post.2021: /tmp/result.p
post.2021: /tmp/2021.ch
post.2021: <STDOUT>
model.2023: /tmp/2023.
model.2023: /tmp/2022.
model.2023: <STDOUT:
post.2022: /tmp/result.p
post.2022: /tmp/2022.ch
post.2022: <STDOUT>
post.2023: /tmp/result.p
post.2023: /tmp/2023.ch
post.2023: <STDOUT>
prep.2021: /tmp/file-raw.
prep.2021: /tmp/file-raw.
prep.2021: /tmp/file-raw
prep.2021: /tmp/2020.ch
prep.2021: /tmp/file-raw.
prep.2021: /tmp/file-raw
prep.2021: <STDOUT>

Lessons Learned

Requirements for Workflow Engines

Expose Context / DAGs of Workflows
Data/(File) Notions
Reflection in Execution Runtime?

Requirements for Monitoring Solutions

Pick up context to allow associations
Support User-Specific Metadata with record
API to interact with monitoring toolkit
Allow counters per MPI Communicator

Requirements for Application Developers

Make intent explicit: use Libs/DSL (e.g. HDF5)
Enable Darshan with at least a subset of runs

Donate traces and logs for a training body.

Thank you!
Questions?

luettgau@dkrz.de

Thank you!
Questions?

luettgau@dkrz.de

Workflow Engines: Swift, Cylc, Tigres, etc.

Cylc, Swift-k, Fireworks

Job centric, with tasks and data targets. Tasks are distributed and possibly run on **remote systems**. Data products might be moved between sites.

Usually, a coarse granular dependency graph.

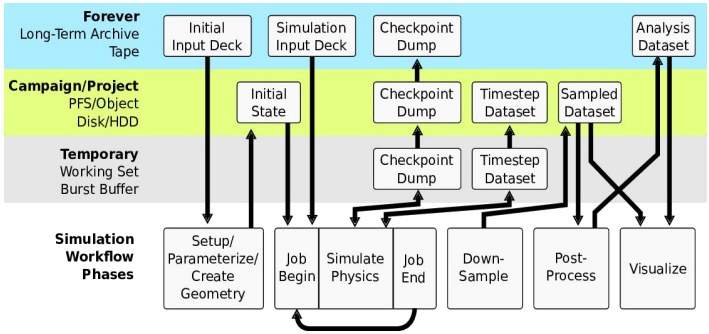
Swift-t, Tigres, Spark/RDD Lineage, QDO

A large **integrated** (MPI) application with many different tasks within the application. With **exascale** in mind and also closer to **in situ** enabled workflows.

Closer to a programming language.

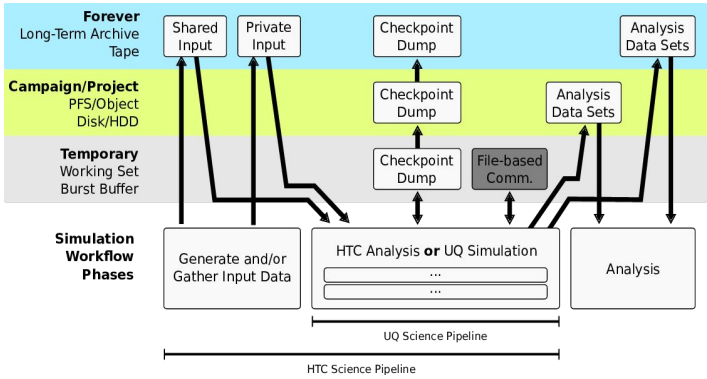
Common Scientific Workflows in HPC

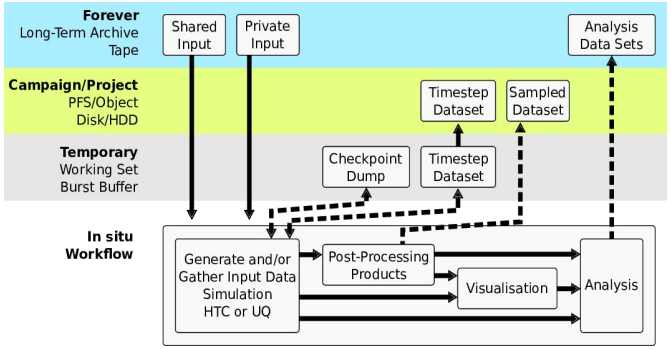
What makes a workflow?



SIM

UQ or HTC





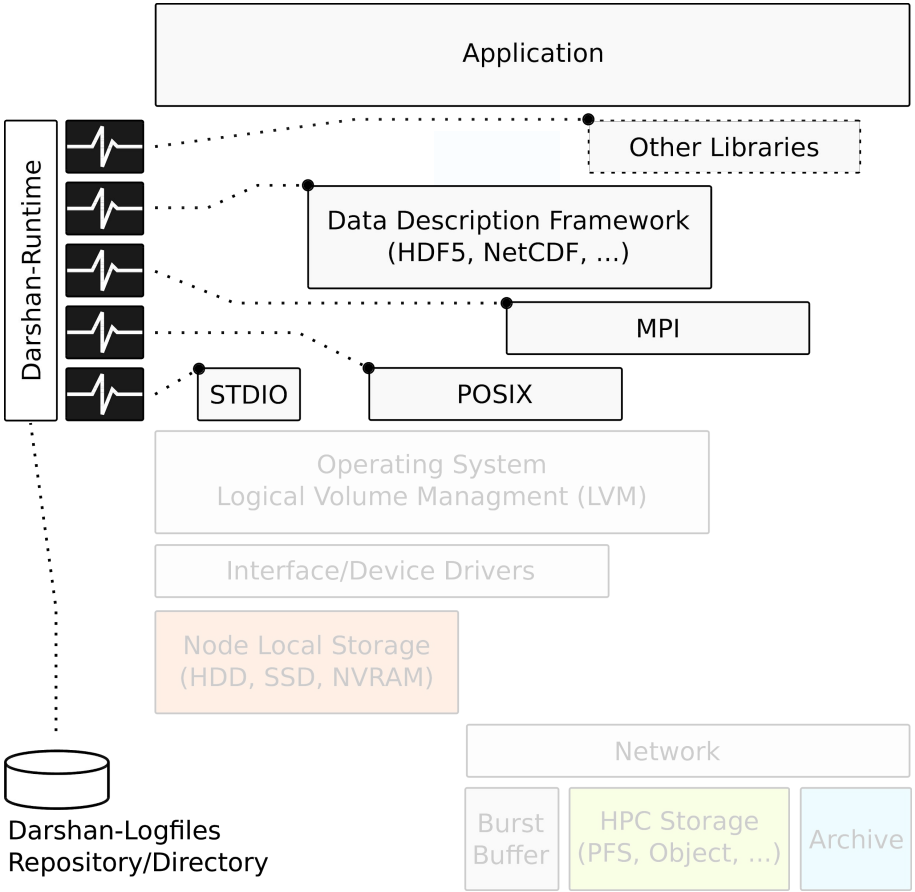
in situ

SIM and HTC/UQ are derived figures from [1]. For outlook on workflows refer to [2].

[1] LANL, NERSC, and SNL, “APEX Workflows.”, Whitepaper, Mar. 2016
Online: https://www.nersc.gov/assets/apex-workflows-v2.pdf

[2] E. Deelman *et al.*, “The future of scientific workflows,” *The International Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 159–175, Jan. 2018.

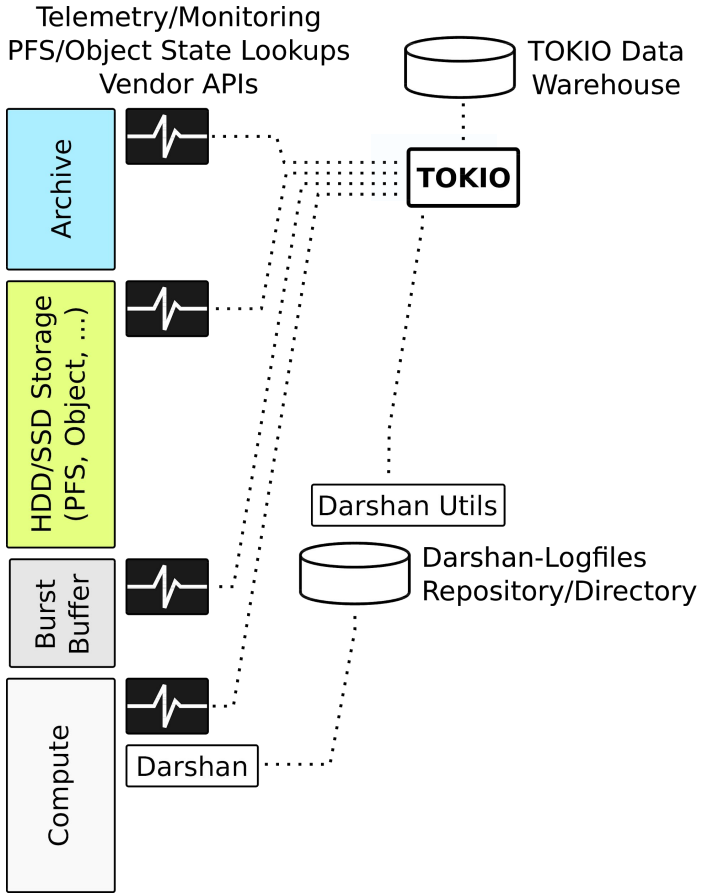
Darshan: Instrumentation at Library/Application Layer




```
$ export LD_PRELOAD=libdarshan.so
$ mpiexec -np 4 ./hellompi
```

[Darshan] HPC I/O Characterization Tool - https://www.mcs.anl.gov/research/projects/darshan/

TOKIO: Total Knowledge of Input/Output



Comprehensive capture of I/O activity

Support different storage services in data center

May require privileged access in many cases

[TOKIO] http://www.nersc.gov/research-and-development/tokio/