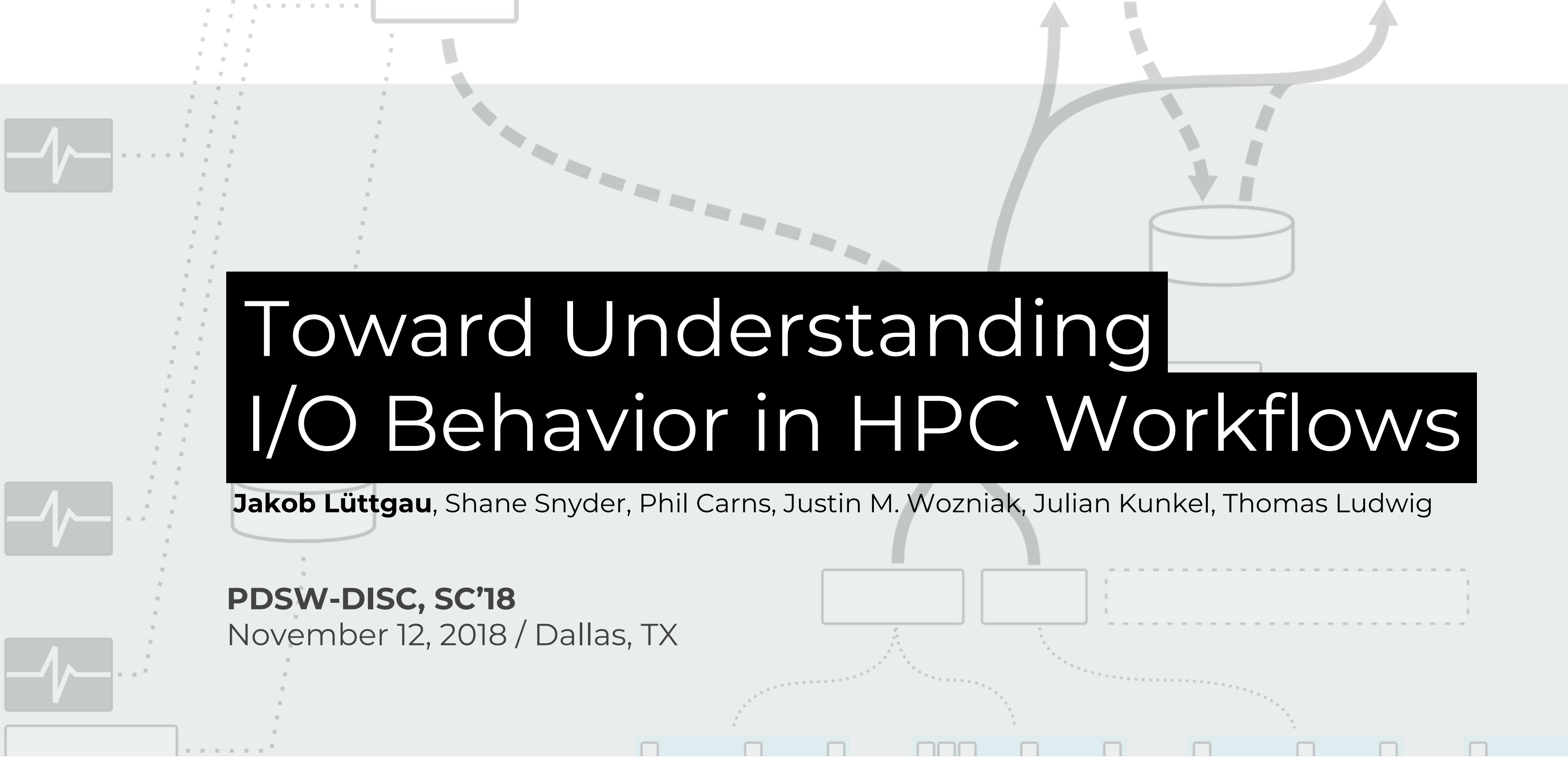


Toward Understanding I/O Behavior in HPC Workflows

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

PDSW-DISC, SC'18
November 12, 2018 / Dallas, TX

DKRZ
DEUTSCHES
KLIMARECHENZENTRUM

Argonne
NATIONAL LABORATORY

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

UHH
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Overview

Motivation

Workflows & I/O Monitoring

Architecture

Demo

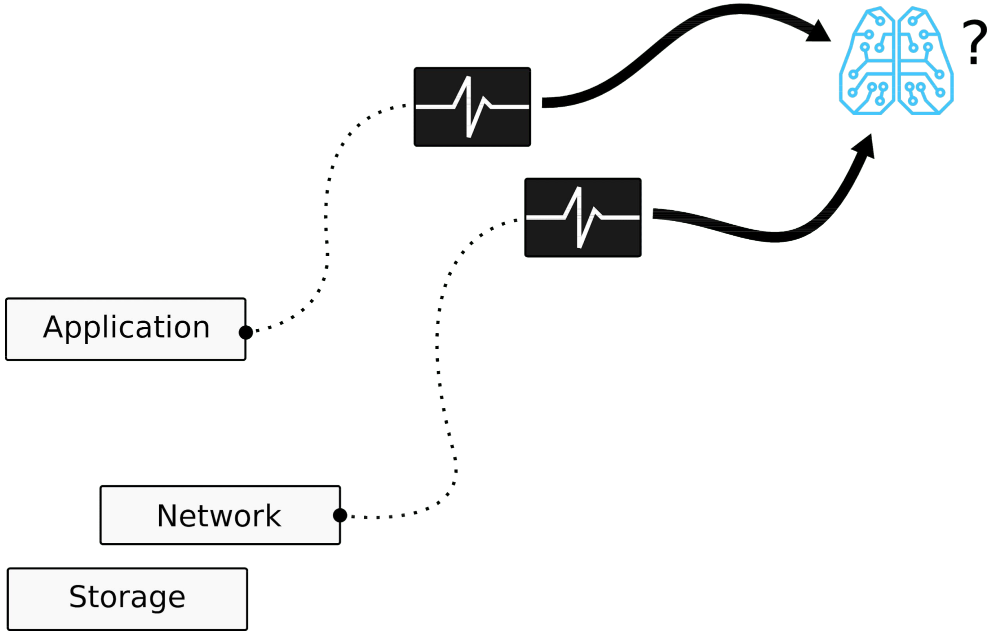
Outlook & Summary

Trying to add a missing link so we can move closer to realizing smarter systems…

Require **new interfaces** to preserve information about **structure of data**.

How to anticipate **user intentions** and **I/O behavior** of applications ?

Require **tools to observe and record** system activity as a basis to gain insight



Workflows
a HPC Storage
Perspective?

Workflows offer …

… anticipatable future activity

… implicit intent to be discovered

… explicit intent description

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.

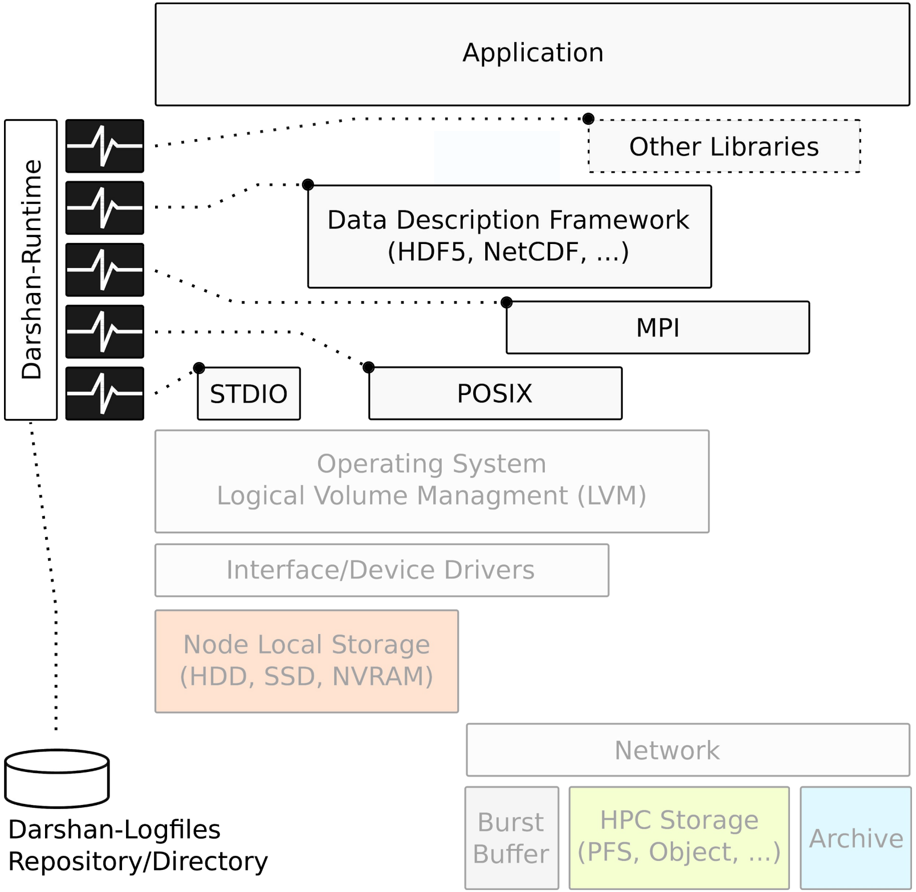
Closer to a programming language.

Holistic I/O Monitoring for HPC

Tracking at the Application/Library Layer

Total Knowledge of I/O in Data Centers

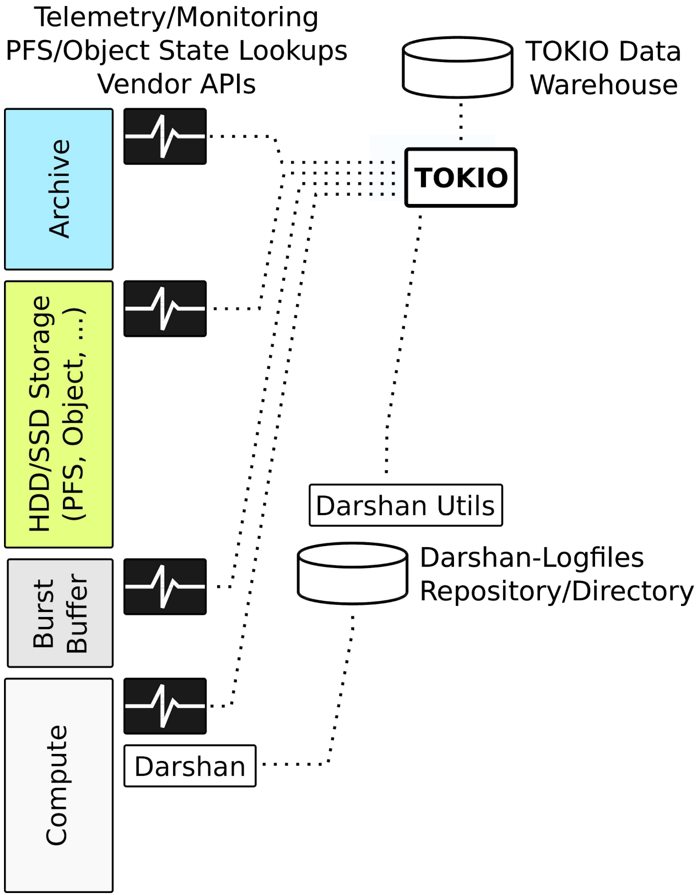
Darshan: Instrumentation at Library/Application Layer



```
$ 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/

Toward Understanding Workflow I/O

Combine workflow descriptions with monitoring information from Darshan/TOKIO, etc.

Benefits:

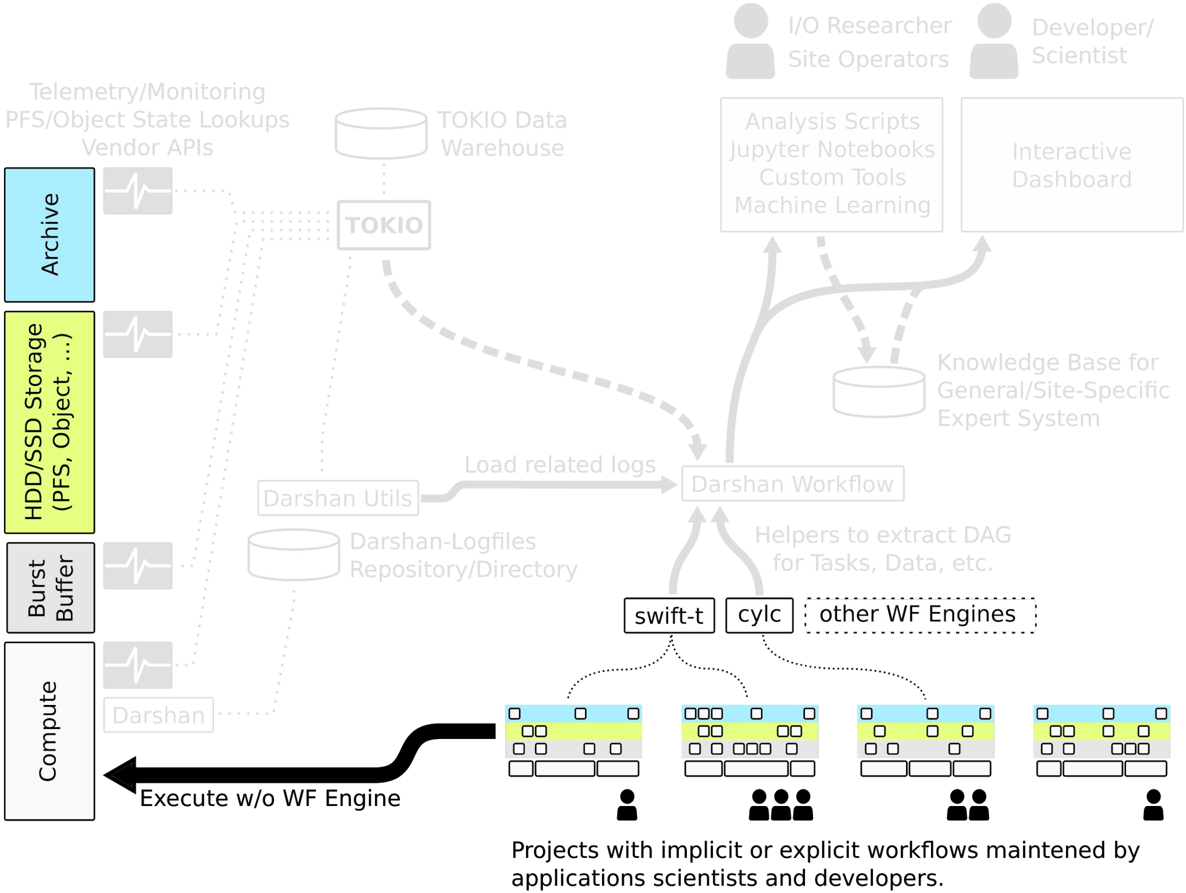
- Insight useful for operating decisions and system design
- Communication with users, relatable to their scientific process
- Source of information for smarter systems

Requirements:

- Support multiple workflow engines as communities use different tools across difference sites

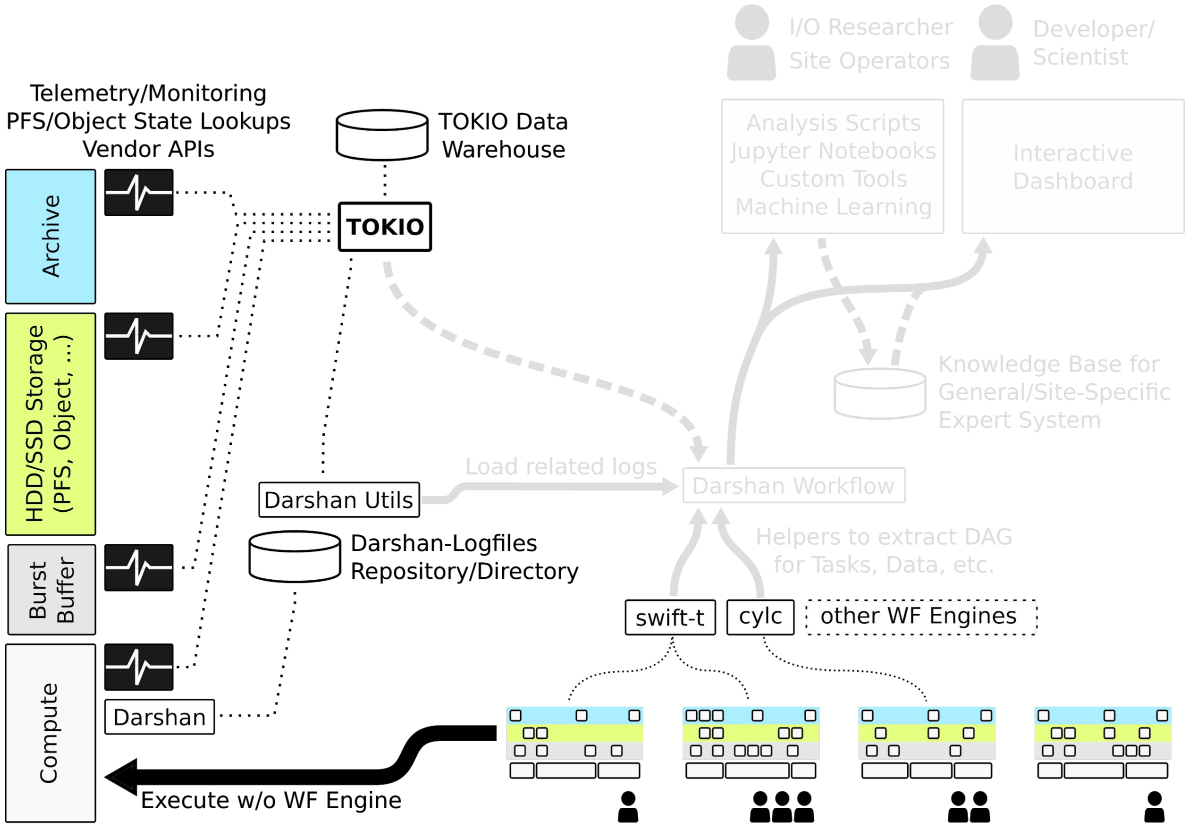
- Explore convenient toolchain for researchers and operators
- User facing component to communicate advice

Architecture for Augmenting I/O in Workflows



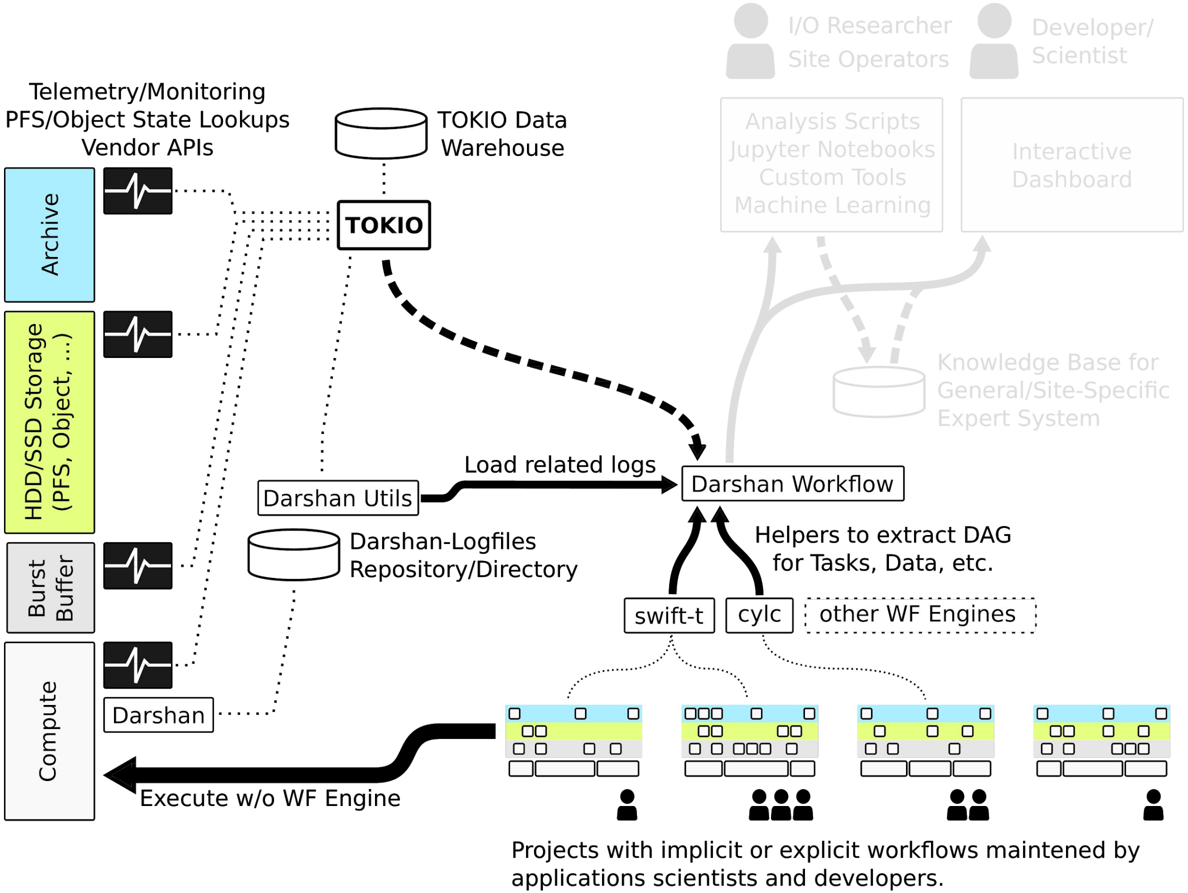
Projects with implicit or explicit workflows maintened by applications scientists and developers.

Architecture for Augmenting I/O in Workflows

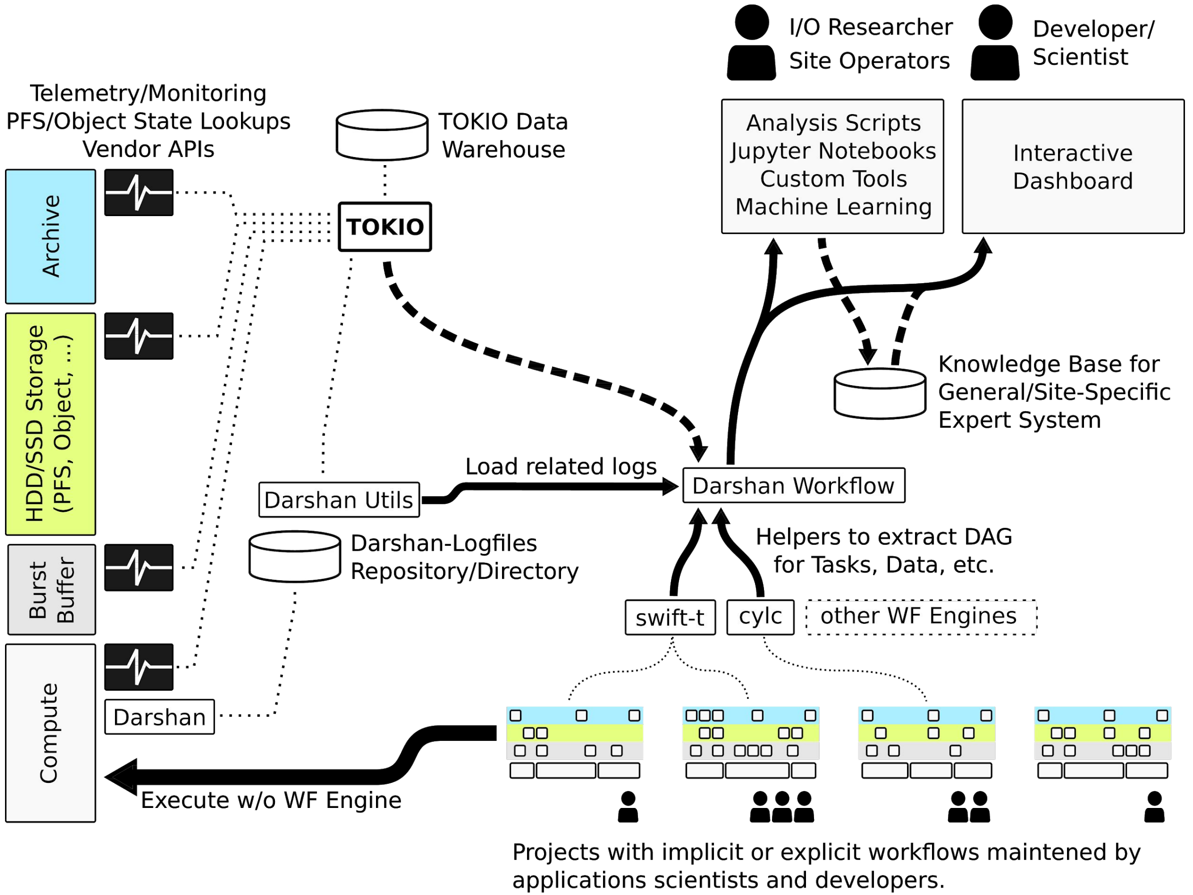


Projects with implicit or explicit workflows maintened by applications scientists and developers.

Architecture for Augmenting I/O in Workflows



Architecture for Augmenting I/O in Workflows



Projects with implicit or explicit workflows maintened by applications scientists and developers.

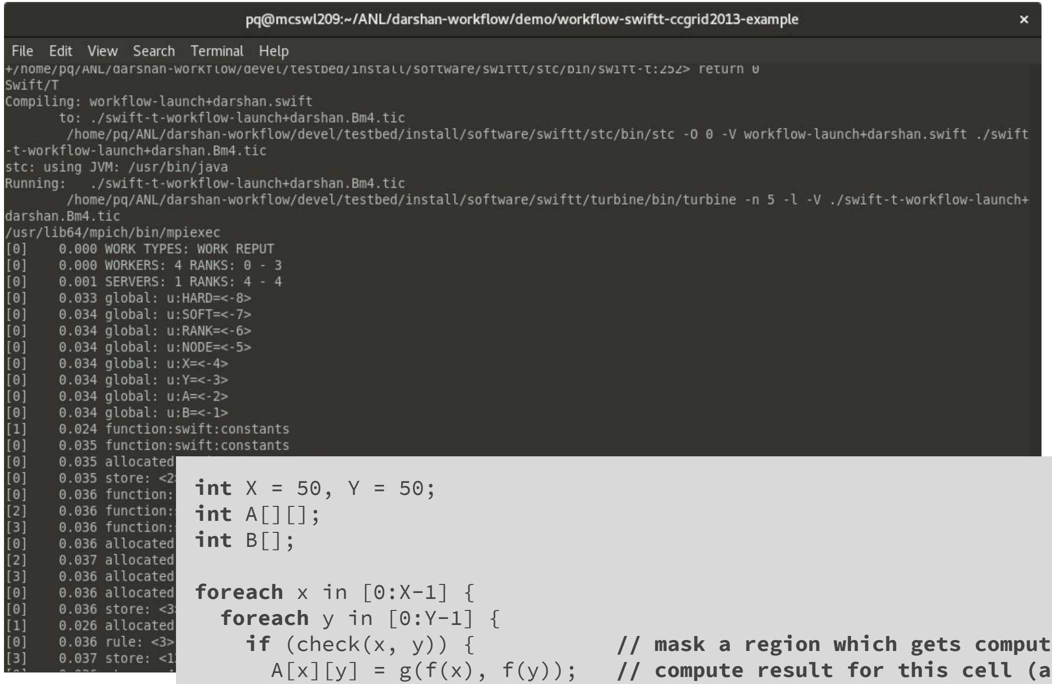
Case Study
& Demonstration

Example Workflow

Research Perspective

User Perspective

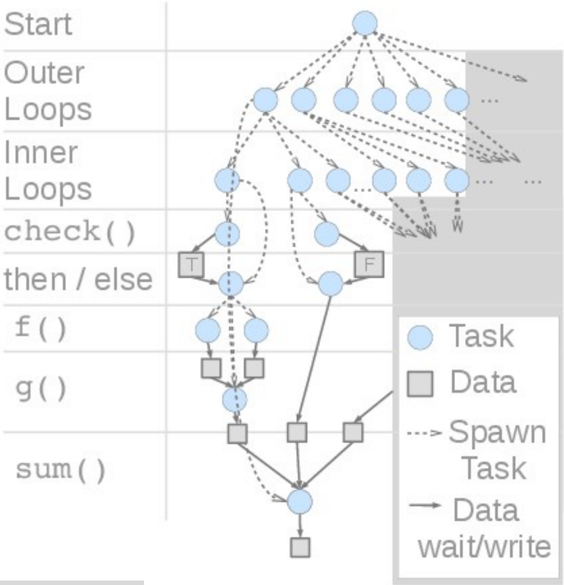
swift


pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-swiftt-ccgrid2013-example
File Edit View Search Terminal Help

```
+/home/pq/ANL/darshan-workflow/devel/testbed/install/software/swiftt/stc/bin/swift-t:252> return 0
Swift/T
Compiling: workflow-launch+darshan.swift
       to: ./swift-t-workflow-launch+darshan.Bm4.tic
        /home/pq/ANL/darshan-workflow/devel/testbed/install/software/swiftt/stc/bin/stc -O 0 -V workflow-launch+darshan.swift ./swift
-t-workflow-launch+darshan.Bm4.tic
stc: using JVM: /usr/bin/java
Running:   ./swift-t-workflow-launch+darshan.Bm4.tic
        /home/pq/ANL/darshan-workflow/devel/testbed/install/software/swiftt/turbine/bin/turbine -n 5 -l -V ./swift-t-workflow-launch+
darshan.Bm4.tic
/usr/lib64/mpich/bin/mpiexec
[0]    0.000 WORK TYPES: WORK REPUT
[0]    0.000 WORKERS: 4 RANKS: 0 - 3
[0]    0.001 SERVERS: 1 RANKS: 4 - 4
[0]    0.033 global: u:HARD=<-8>
[0]    0.034 global: u:SOFT=<-7>
[0]    0.034 global: u:RANK=<-6>
[0]    0.034 global: u:NODE=<-5>
[0]    0.034 global: u:X=<-4>
[0]    0.034 global: u:Y=<-3>
[0]    0.034 global: u:A=<-2>
[0]    0.034 global: u:B=<-1>
[1]    0.024 function:swift:constants
[0]    0.035 function:swift:constants
[0]    0.035 allocated
[0]    0.035 store: <2
[0]    0.036 function:
[2]    0.036 function:
[3]    0.036 function:
[0]    0.036 allocated
[2]    0.037 allocated
[3]    0.036 allocated
[0]    0.036 allocated
[0]    0.036 store: <3
[1]    0.026 allocated
[0]    0.036 rule: <3>
[3]    0.037 store: <1
```

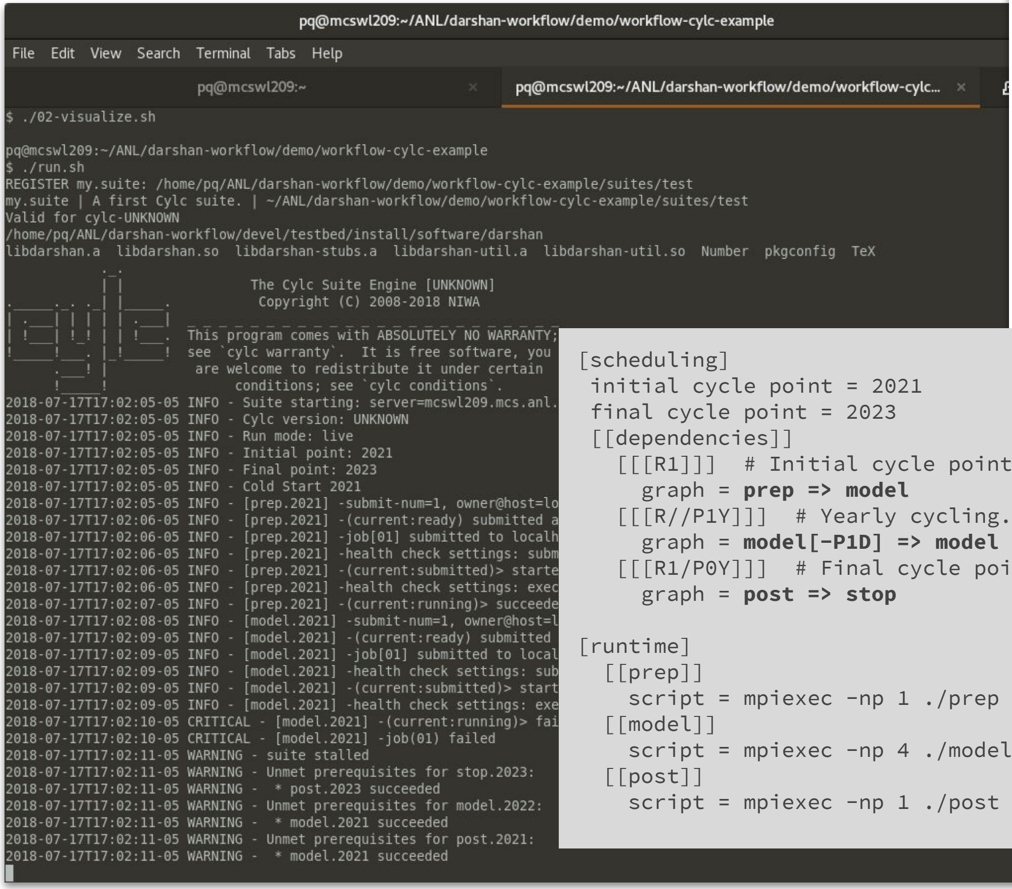
```
int X = 50, Y = 50;
int A[][];
int B[];

foreach x in [0:X-1] {
  foreach y in [0:Y-1] {
    if (check(x, y)) {              // mask a region which gets computed
      A[x][y] = g(f(x), f(y));     // compute result for this cell (a physics process)
    } else {
      A[x][y] = 0;                 // default for skipped cells
    }
  }
  B[x] = sum(A[x]);                 // compute some aggregate metric
}
```



http://swift-lang.org

cylc

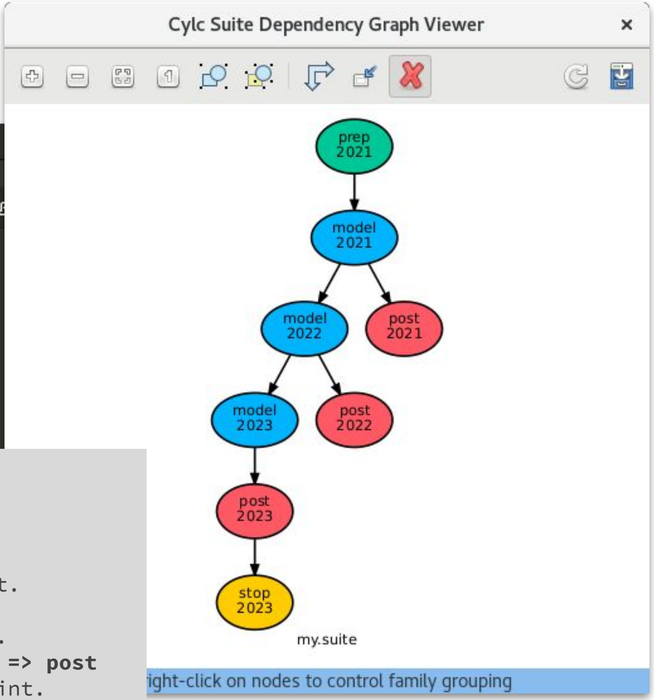

pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc-example
File Edit View Search Terminal Tabs Help
pq@mcswl209:~
pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc...

```
$ ./02-visualize.sh

pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc-example
$ ./run.sh
REGISTER my.suite: /home/pq/ANL/darshan-workflow/demo/workflow-cylc-example/suites/test
my.suite | A first Cylc suite. | ~/ANL/darshan-workflow/demo/workflow-cylc-example/suites/test
Valid for cylc-UNKNOWN
/home/pq/ANL/darshan-workflow/devel/testbed/install/software/darshan
libdarshan.a  libdarshan.so  libdarshan-stubs.a  libdarshan-util.a  libdarshan-util.so  Number  pkgconfig  TeX
            ._.
            | |                 The Cylc Suite Engine [UNKNOWN]
._____._. ._| |_____.           Copyright (C) 2008-2018 NIWA
| .___| | | | | .___|  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
| !___| !_! | | !___.  This program comes with ABSOLUTELY NO WARRANTY;
!_____!___. |_!_____!  see `cylc warranty`.  It is free software, you
      .___! |           are welcome to redistribute it under certain
      !_____!                conditions; see `cylc conditions`.
2018-07-17T17:02:05-05 INFO - Suite starting: server=mcswl209.mcs.anl.
2018-07-17T17:02:05-05 INFO - Cylc version: UNKNOWN
2018-07-17T17:02:05-05 INFO - Run mode: live
2018-07-17T17:02:05-05 INFO - Initial point: 2021
2018-07-17T17:02:05-05 INFO - Final point: 2023
2018-07-17T17:02:05-05 INFO - Cold Start 2021
2018-07-17T17:02:05-05 INFO - [prep.2021] -submit-num=1, owner@host=lo
2018-07-17T17:02:06-05 INFO - [prep.2021] -(current:ready) submitted a
2018-07-17T17:02:06-05 INFO - [prep.2021] -job[01] submitted to localh
2018-07-17T17:02:06-05 INFO - [prep.2021] -health check settings: subm
2018-07-17T17:02:06-05 INFO - [prep.2021] -(current:submitted)> starte
2018-07-17T17:02:06-05 INFO - [prep.2021] -health check settings: exec
2018-07-17T17:02:07-05 INFO - [prep.2021] -(current:running)> succeede
2018-07-17T17:02:08-05 INFO - [model.2021] -submit-num=1, owner@host=l
2018-07-17T17:02:09-05 INFO - [model.2021] -(current:ready) submitted
2018-07-17T17:02:09-05 INFO - [model.2021] -job[01] submitted to local
2018-07-17T17:02:09-05 INFO - [model.2021] -health check settings: sub
2018-07-17T17:02:09-05 INFO - [model.2021] -(current:submitted)> start
2018-07-17T17:02:09-05 INFO - [model.2021] -health check settings: exe
2018-07-17T17:02:10-05 CRITICAL - [model.2021] -(current:running)> fai
2018-07-17T17:02:10-05 CRITICAL - [model.2021] -job(01) failed
2018-07-17T17:02:11-05 WARNING - suite stalled
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for stop.2023:
2018-07-17T17:02:11-05 WARNING -  * post.2023 succeeded
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for model.2022:
2018-07-17T17:02:11-05 WARNING -  * model.2021 succeeded
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for post.2021:
2018-07-17T17:02:11-05 WARNING -  * model.2021 succeeded
```

```
[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
```



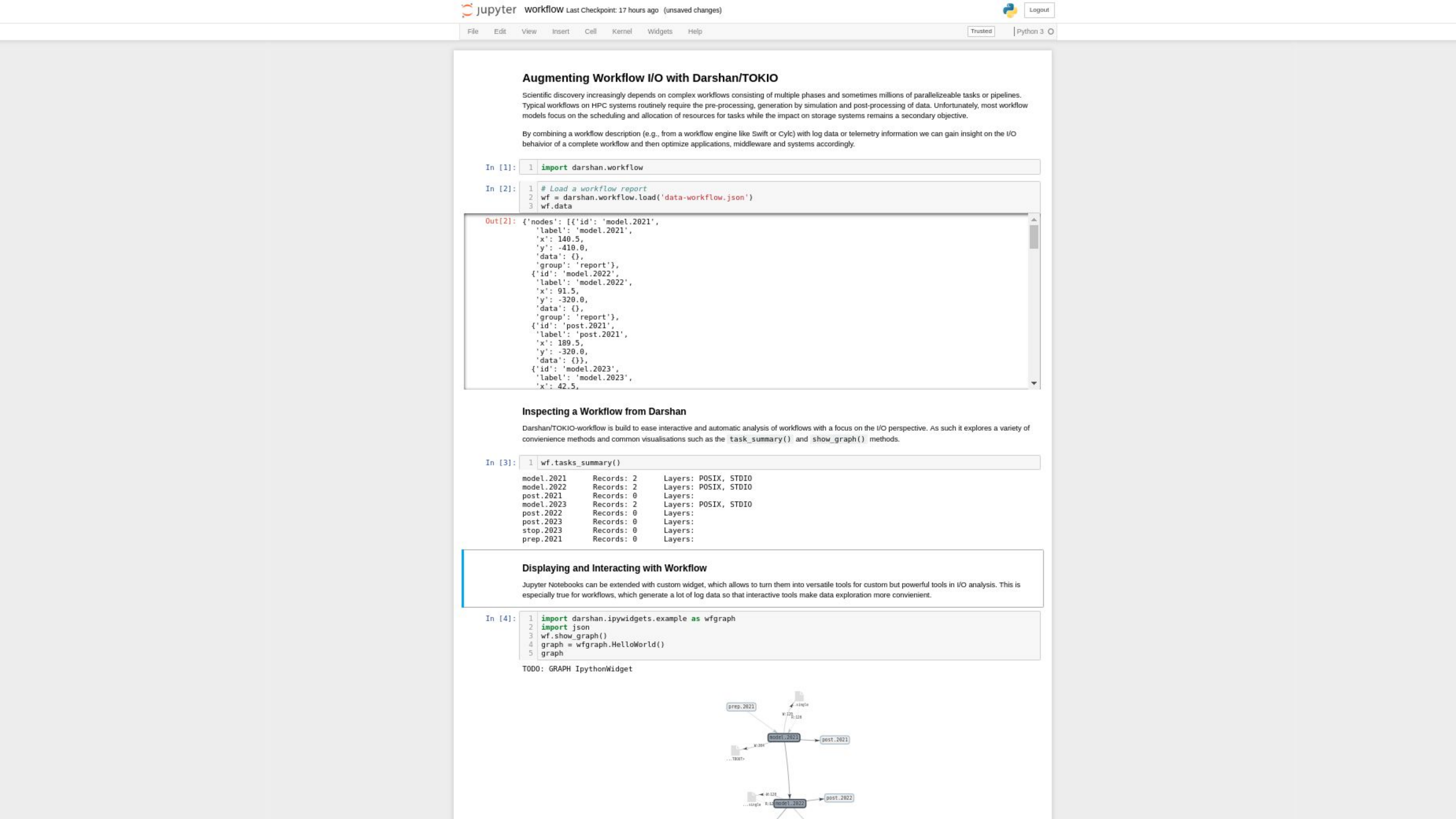
https://cylc.github.io/cylc/

Perspective for **I/O Research** and **Site Operating**?

Interactive Tools/Dashboards to ease navigating **overwhelming amounts of log data**, with “algebra”-like semantics for convenient aggregation of multiple tasks, data objects or pipelines.

Python Library for use in, e.g., **jupyter notebooks**, to draft/prototype/provide **templates** for more sophisticated and **reproducible** analysis.

JavaScript Packages (NPM) for visualisation/tools allowing easy **reuse** in custom tools , jupyter notebooks (widget plugins), and dashboards (e.g., Grafana).



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 convienience methods and common visualisations such as the `task_summary()` and `show_graph()` methods.

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

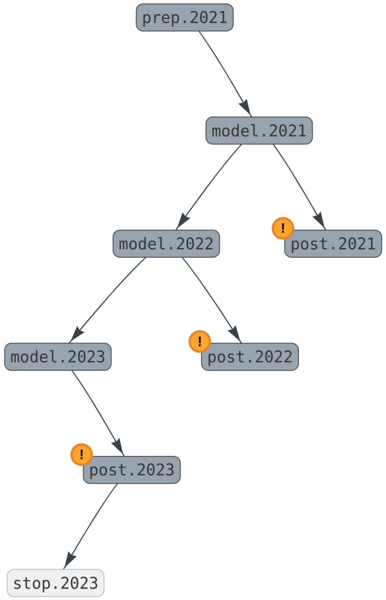
```
model.2021      Records: 2      Layers: POSIX, STDIO
model.2022      Records: 2      Layers: POSIX, STDIO
post.2021       Records: 0      Layers:
model.2023      Records: 2      Layers: POSIX, STDIO
post.2022       Records: 0      Layers:
post.2023       Records: 0      Layers:
stop.2023       Records: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

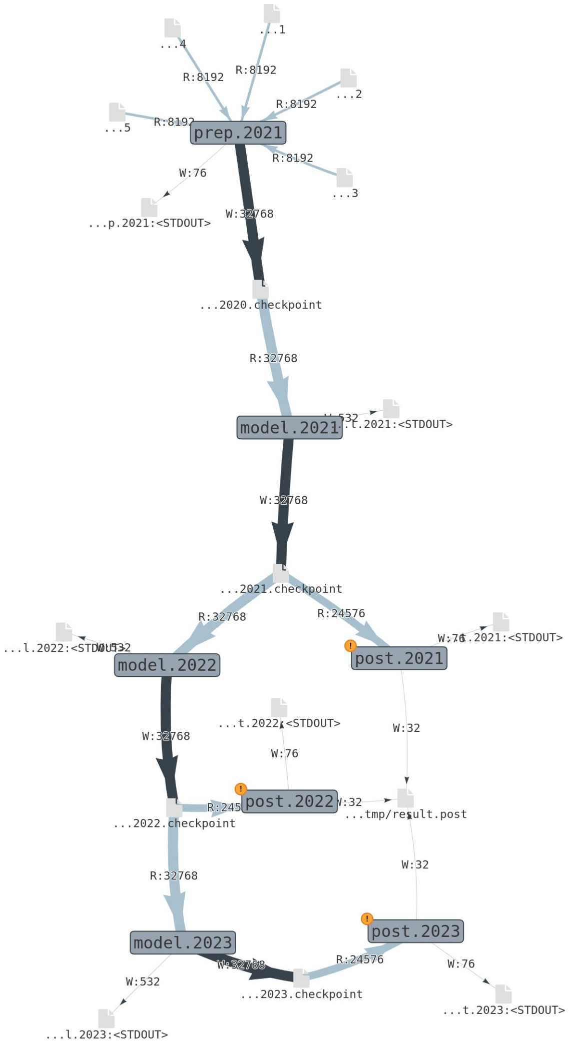
Displaying and Interacting with Workflow

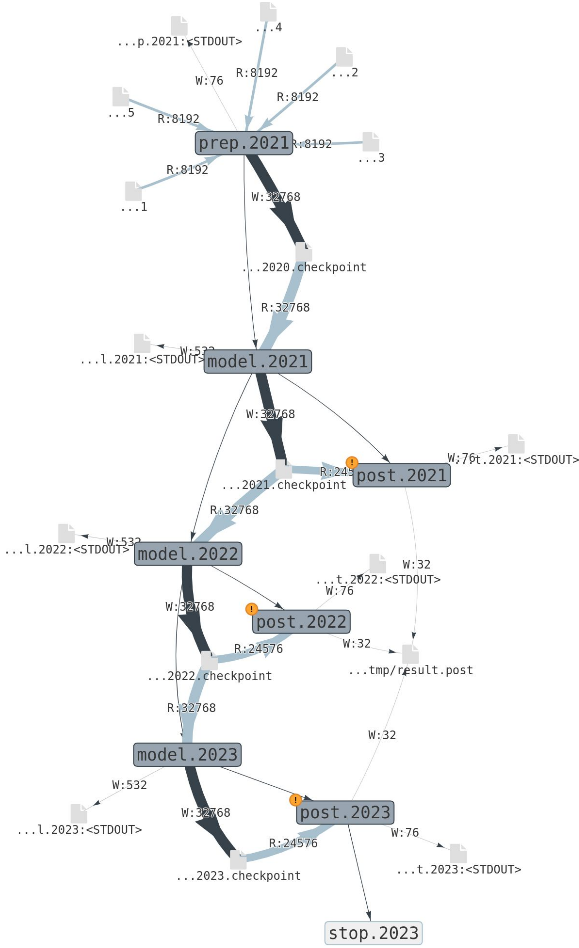
Jupyter Notebooks can be extended with custom widget, which allows to turn them into versatile tools for custom but powerful tools in I/O analysis. This is especially true for workflows, which generate a lot of log data so that interactive tools make data exploration more convienient.

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


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


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

Communication with Scientists/Developers

Maintain affinity to scientists perspective

- Stick to relationship of tasks/pipelines used by scientists/developers
- Use intuitiv presentation of data-flow by extending graph of workflow

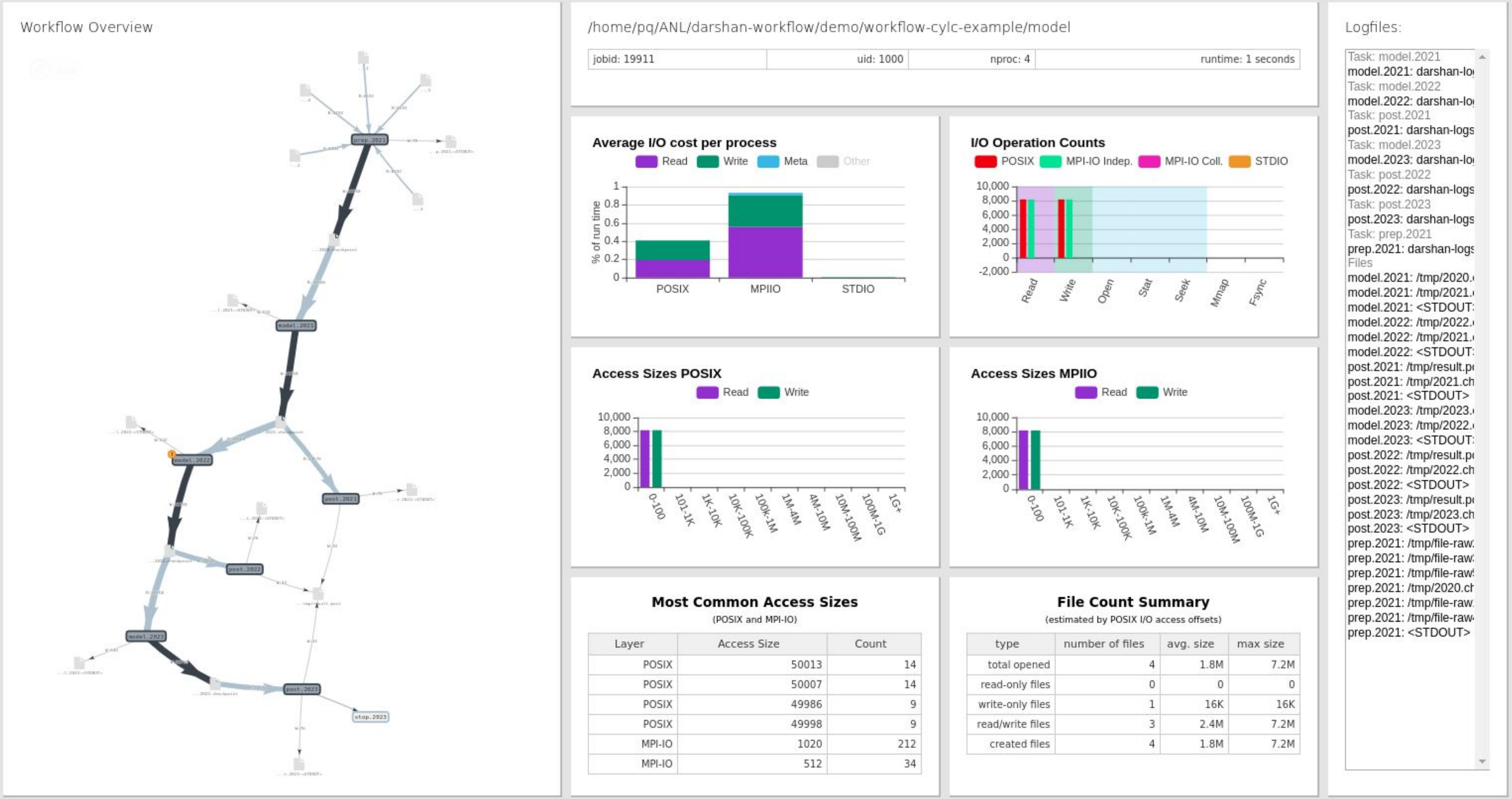
Interactive to manage complexity

- 100s or 1000s of different tasks and files in a workflow
- Possibly, millions of log records per task (HTC, UQ)
- Make it easy to aggregate multiple log records

Integration with expert advice

- Human in the loop
- Automatic advisories with machine learning (mid/long-term)

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



/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

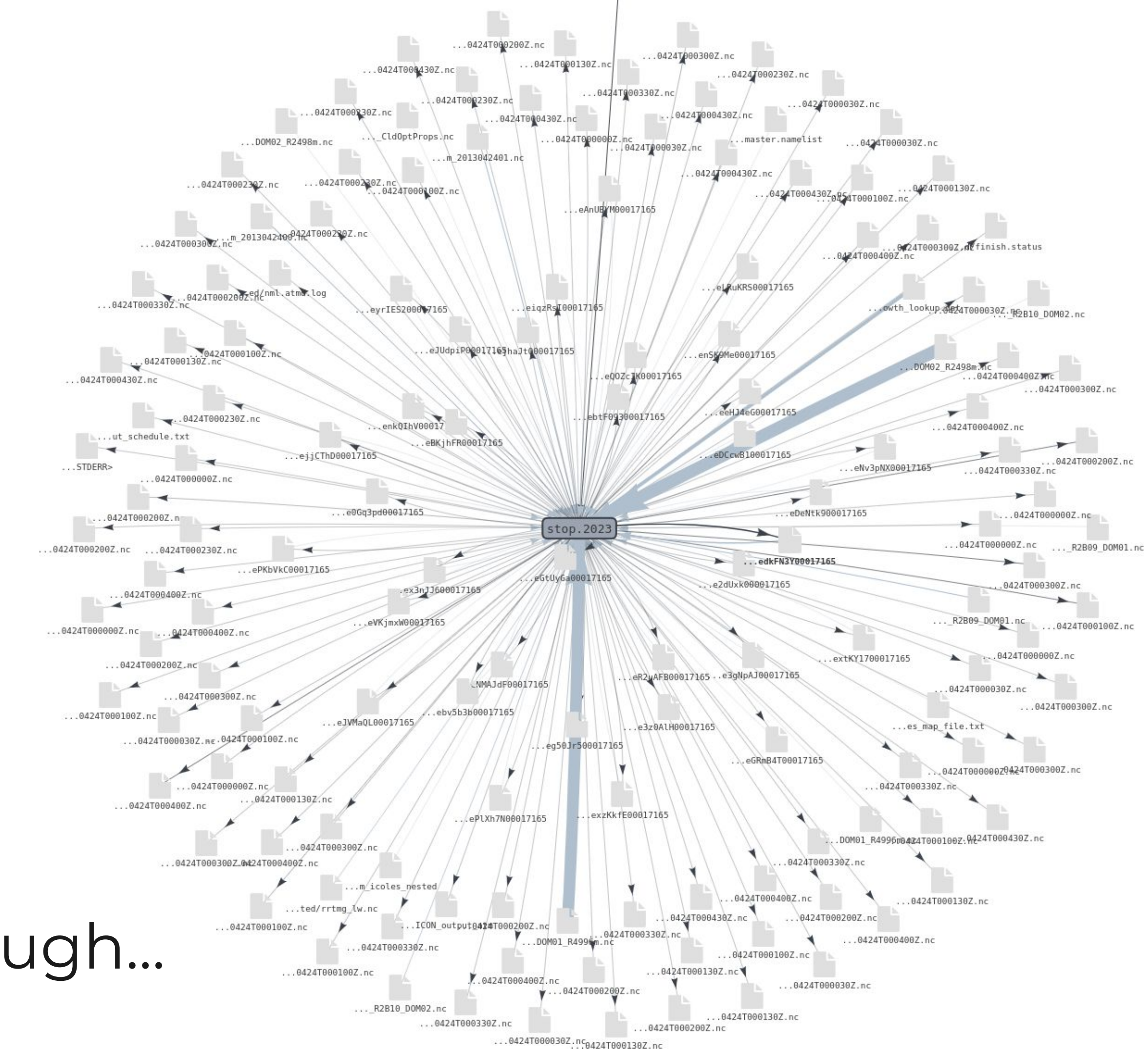
File Count Summary

(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

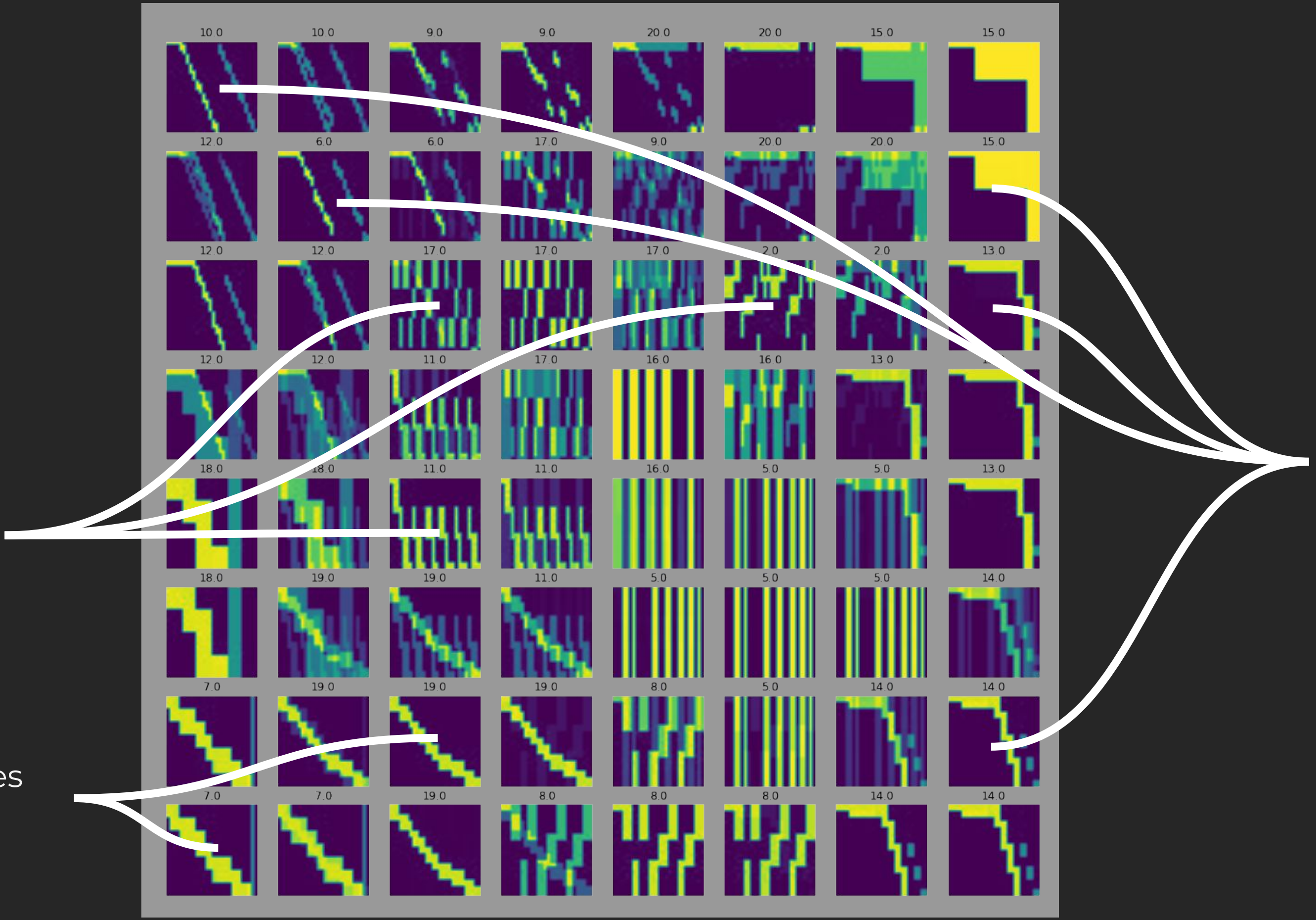
Logfiles:

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>



What a **real task** might look like though…

Analyzing **Access Patterns**



Input Files

Output Files

In this case
diagnostic files
otherwise
not so clear

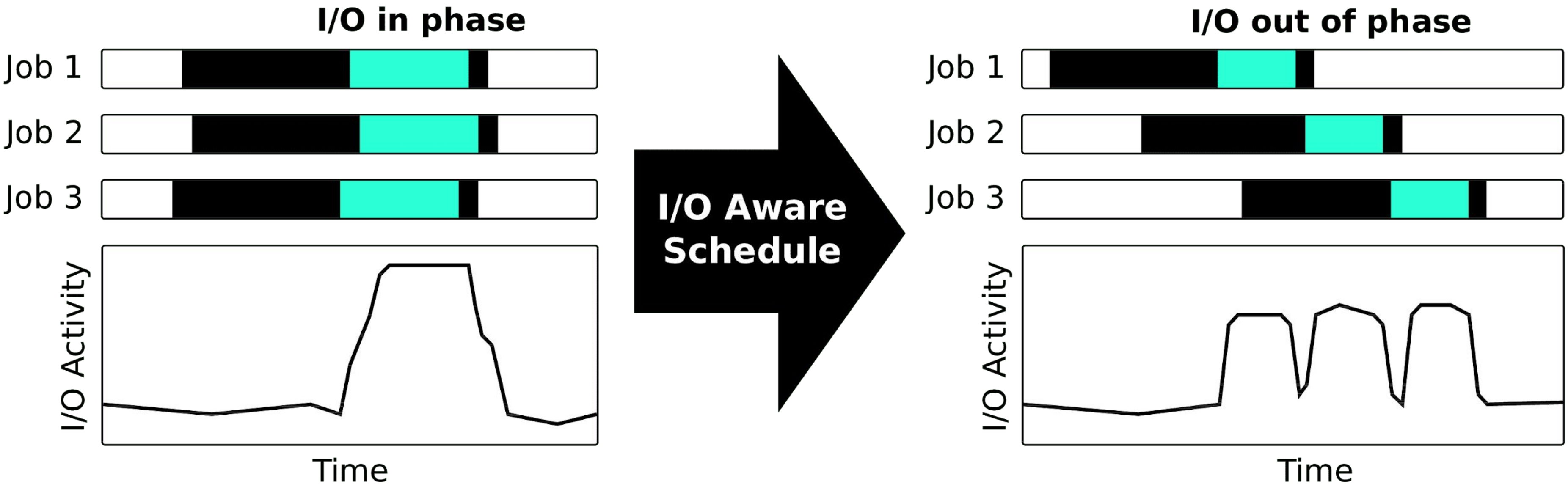
Toward Adaptive I/O Systems

Influence **Job Scheduling** decisions

Support **I/O Middleware**

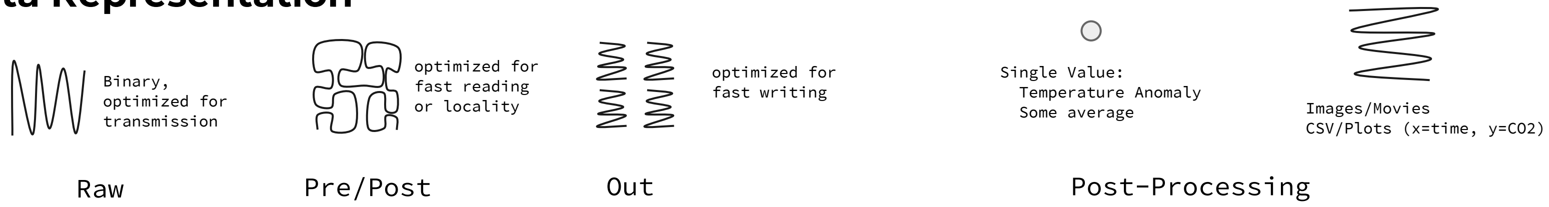
- Data Placement
- Transformations

Use Case 1: I/O-Aware Scheduling for Workflows



Use Case 2: Benefits for I/O Middleware (1/2)

Data Representation



Binary,
optimized for
transmission

Raw

optimized for
fast reading
or locality

Pre/Post

optimized for
fast writing

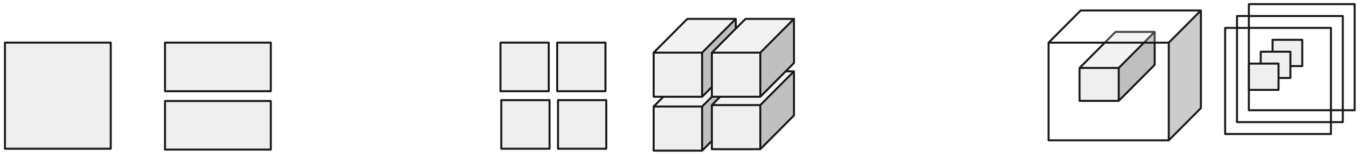
Out

Single Value:
Temperature Anomaly
Some average

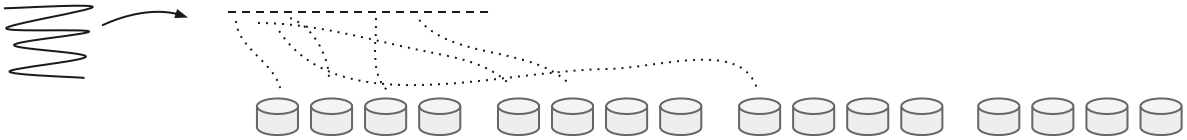
Images/Movies
CSV/Plots (x=time, y=CO2)

Post-Processing

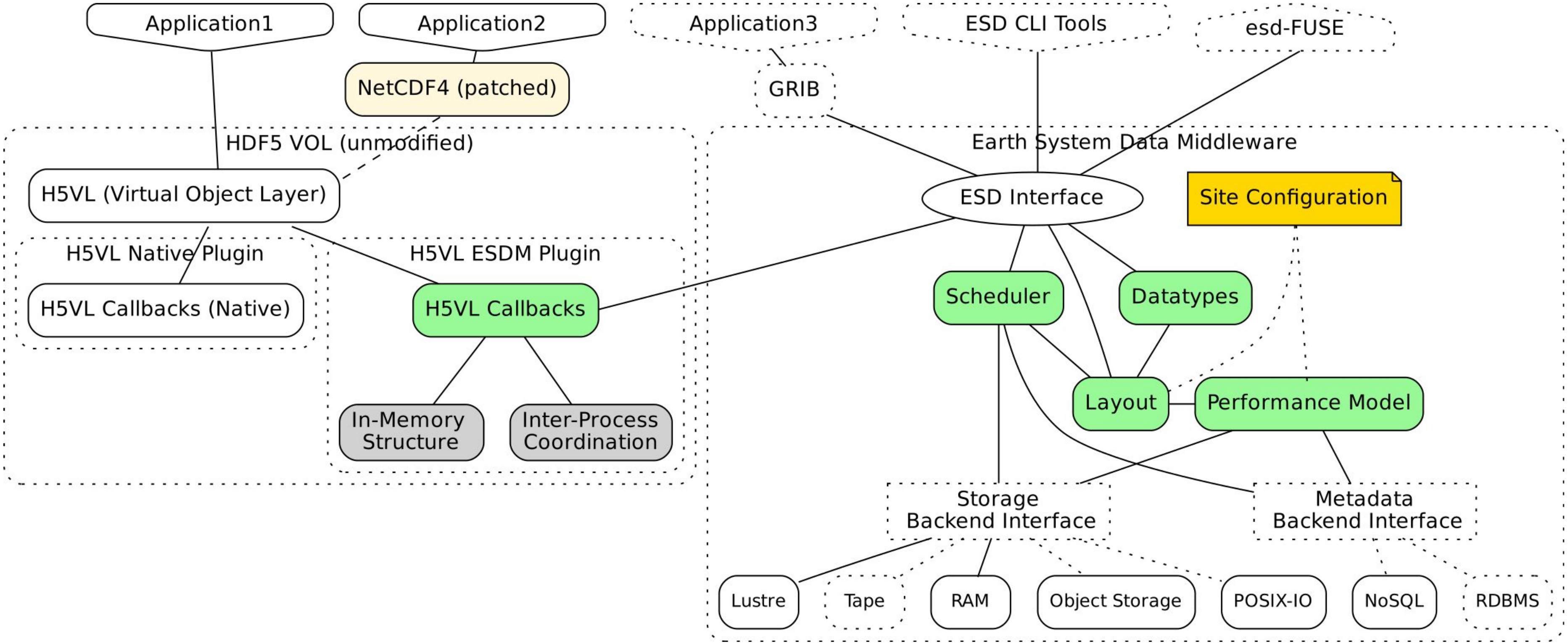
Domain Decomposition



Layout on Storage



Use Case 2: Benefits for I/O Middleware (2/2)



Discussion Summary

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 instrumentation with a subset of runs

Collect traces and logs for a training body.

Thank you!
Questions?

luettgau@dkrz.de

Disclaimer

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favor by the United States Government, the Department of Energy, or the National Energy Technology Laboratory. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government, the Department of Energy, or the National Energy Technology Laboratory, and shall not be used for advertising or product endorsement purposes.

This work was supported by the ESiWACE project, which received funding from the EU Horizon 2020 research and innovation programme under grant agreement No 675191.

The information and views set out in this work are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

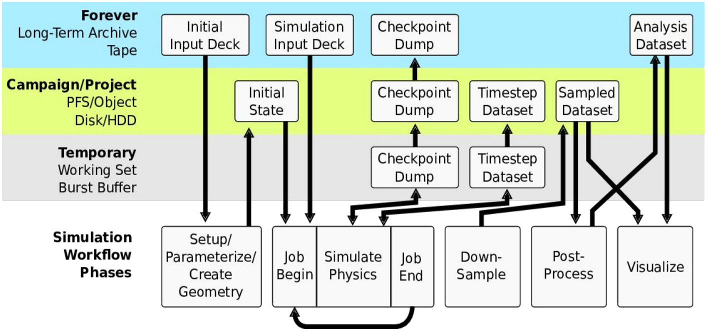
Appendix

Generic HPC Workflows

Example Climate Workflow

Common Scientific Workflows in HPC

What makes a workflow?



Forever
Long-Term Archive
Tape

Campaign/Project
PFS/Object
Disk/HDD

Temporary
Working Set
Burst Buffer

Simulation
Workflow
Phases

Initial Input Deck

Simulation Input Deck

Checkpoint Dump

Analysis Dataset

Initial State

Checkpoint Dump

Timestep Dataset

Sampled Dataset

Checkpoint Dump

Timestep Dataset

Setup/ Parameterize/ Create Geometry

Job Begin

Simulate Physics

Job End

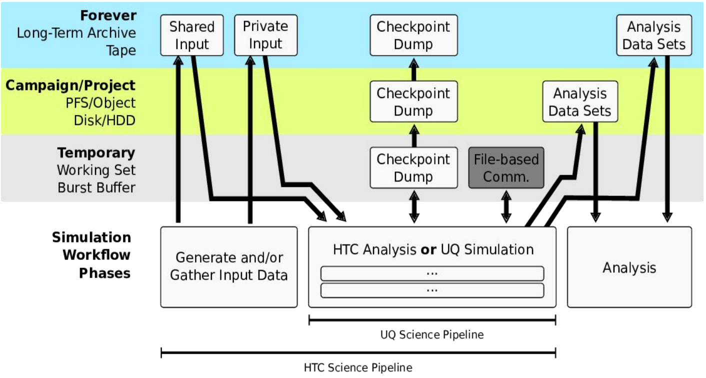
Down-Sample

Post-Process

Visualize

SIM

UQ or HTC



Forever
Long-Term Archive
Tape

Campaign/Project
PFS/Object
Disk/HDD

Temporary
Working Set
Burst Buffer

Simulation
Workflow
Phases

Shared Input

Private Input

Checkpoint Dump

Analysis Data Sets

Checkpoint Dump

Analysis Data Sets

Checkpoint Dump

File-based Comm.

Generate and/or Gather Input Data

HTC Analysis **or** UQ Simulation

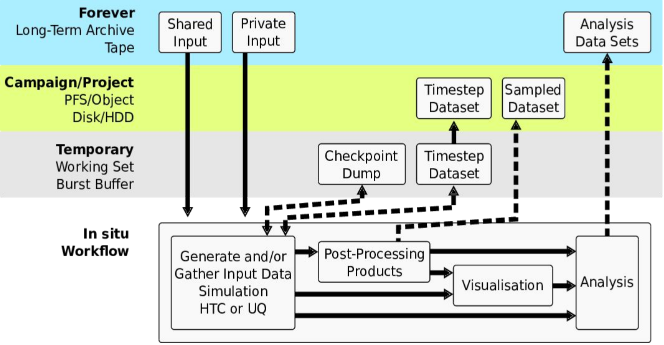
...

...

Analysis

UQ Science Pipeline

HTC Science Pipeline



Forever
Long-Term Archive
Tape

Campaign/Project
PFS/Object
Disk/HDD

Temporary
Working Set
Burst Buffer

In situ
Workflow

Shared Input

Private Input

Analysis Data Sets

Timestep Dataset

Sampled Dataset

Checkpoint Dump

Timestep Dataset

Generate and/or Gather Input Data Simulation HTC or UQ

Post-Processing Products

Visualisation

Analysis

in situ

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

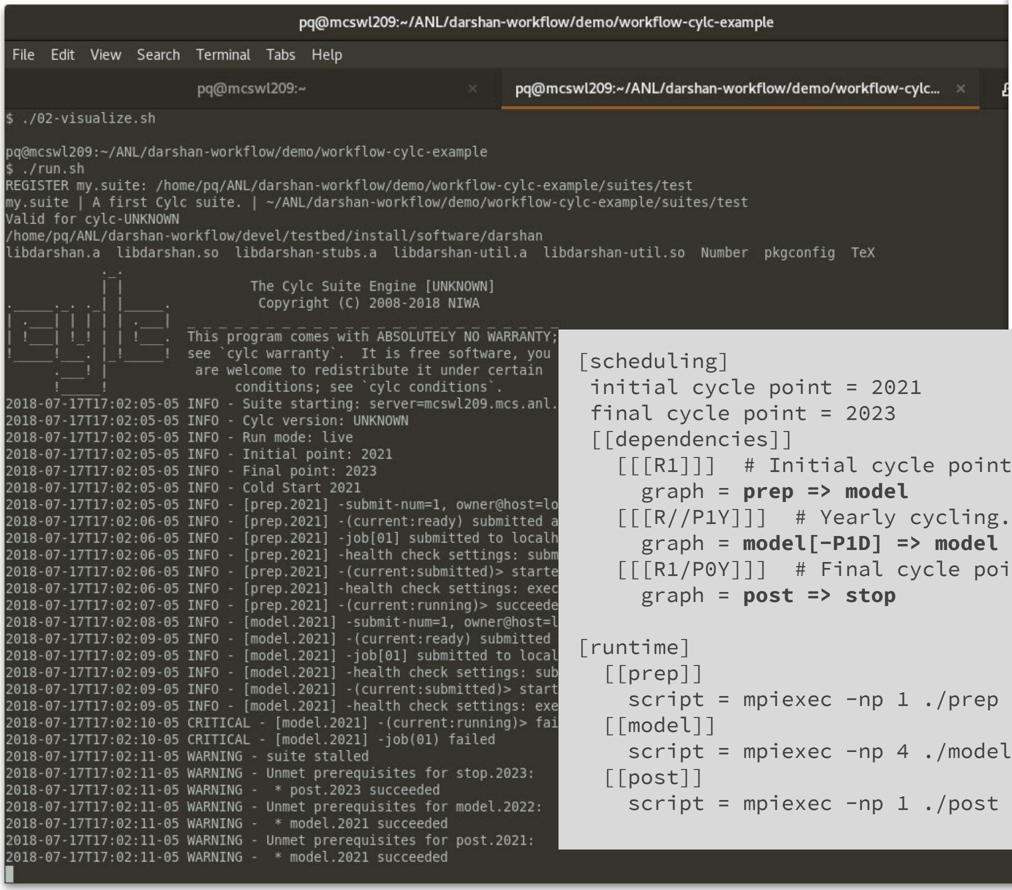
[1] LANL, NERSC, and SNL, “APEX Workflows.”, Whitepaper, Mar. 2016
Online: https://www.nersc.gov/assets/apex-workflows-v2.pdf

[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.

Data-Intensive Exascale Workflow: Climate Modeling

ICON LEM (DOM1)
1.25 km per Cell
4 Million Cells per Level

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.

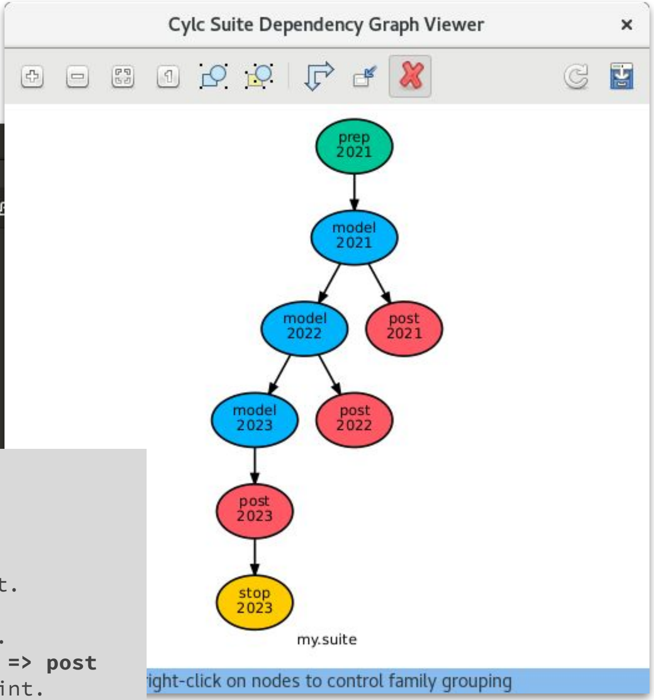

pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc-example
File Edit View Search Terminal Tabs Help
pq@mcswl209:~
pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc...

```
$ ./02-visualize.sh

pq@mcswl209:~/ANL/darshan-workflow/demo/workflow-cylc-example
$ ./run.sh
REGISTER my.suite: /home/pq/ANL/darshan-workflow/demo/workflow-cylc-example/suites/test
my.suite | A first Cylc suite. | ~/ANL/darshan-workflow/demo/workflow-cylc-example/suites/test
Valid for cylc-UNKNOWN
/home/pq/ANL/darshan-workflow/devel/testbed/install/software/darshan
libdarshan.a  libdarshan.so  libdarshan-stubs.a  libdarshan-util.a  libdarshan-util.so  Number  pkgconfig  TeX
            ._.
            | |                 The Cylc Suite Engine [UNKNOWN]
._____._. ._| |_____.            Copyright (C) 2008-2018 NIWA
| .___| | | | | .___|  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
| !___| !_! | | !___.  This program comes with ABSOLUTELY NO WARRANTY;
!_____!___. |_!_____!  see `cylc warranty`.  It is free software, you
      .___! |           are welcome to redistribute it under certain
      !_____!                conditions; see `cylc conditions`.
2018-07-17T17:02:05-05 INFO - Suite starting: server=mcswl209.mcs.anl.
2018-07-17T17:02:05-05 INFO - Cylc version: UNKNOWN
2018-07-17T17:02:05-05 INFO - Run mode: live
2018-07-17T17:02:05-05 INFO - Initial point: 2021
2018-07-17T17:02:05-05 INFO - Final point: 2023
2018-07-17T17:02:05-05 INFO - Cold Start 2021
2018-07-17T17:02:05-05 INFO - [prep.2021] -submit-num=1, owner@host=lo
2018-07-17T17:02:06-05 INFO - [prep.2021] -(current:ready) submitted a
2018-07-17T17:02:06-05 INFO - [prep.2021] -job[01] submitted to localh
2018-07-17T17:02:06-05 INFO - [prep.2021] -health check settings: subm
2018-07-17T17:02:06-05 INFO - [prep.2021] -(current:submitted)> starte
2018-07-17T17:02:06-05 INFO - [prep.2021] -health check settings: exec
2018-07-17T17:02:07-05 INFO - [prep.2021] -(current:running)> succeede
2018-07-17T17:02:08-05 INFO - [model.2021] -submit-num=1, owner@host=l
2018-07-17T17:02:09-05 INFO - [model.2021] -(current:ready) submitted
2018-07-17T17:02:09-05 INFO - [model.2021] -job[01] submitted to local
2018-07-17T17:02:09-05 INFO - [model.2021] -health check settings: sub
2018-07-17T17:02:09-05 INFO - [model.2021] -(current:submitted)> start
2018-07-17T17:02:09-05 INFO - [model.2021] -health check settings: exe
2018-07-17T17:02:10-05 CRITICAL - [model.2021] -(current:running)> fai
2018-07-17T17:02:10-05 CRITICAL - [model.2021] -job(01) failed
2018-07-17T17:02:11-05 WARNING - suite stalled
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for stop.2023:
2018-07-17T17:02:11-05 WARNING -  * post.2023 succeeded
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for model.2022:
2018-07-17T17:02:11-05 WARNING -  * model.2021 succeeded
2018-07-17T17:02:11-05 WARNING - Unmet prerequisites for post.2021:
2018-07-17T17:02:11-05 WARNING -  * model.2021 succeeded
```

```
[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/