Reproducible Workflows

Gregory R. Watson (he/him)
Oak Ridge National Laboratory

Better Software for Reproducible Science tutorial @ SC23

Contributors: Patricia A. Grubel (LANL), David M. Rogers (ORNL), Gregory R. Watson (ORNL)

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is:** David E. Bernholdt, Patricia A. Grubel, David M. Rogers, and Gregory R. Watson, Better Software for Reproducible Science tutorial, in The International Conference for High-Performance Computing, Networking, Storage, and Analysis (SC23), Denver, Colorado, 2023. DOI: 10.6084/m9.figshare.24226105.
- Individual modules may be cited as *Speaker, Module Title*, in *Tutorial Title*, …

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at the Los Alamos National Laboratory, which is managed by Triad National Security, LLC for the U.S. Department of Energy under Contract No.89233218CNA000001
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

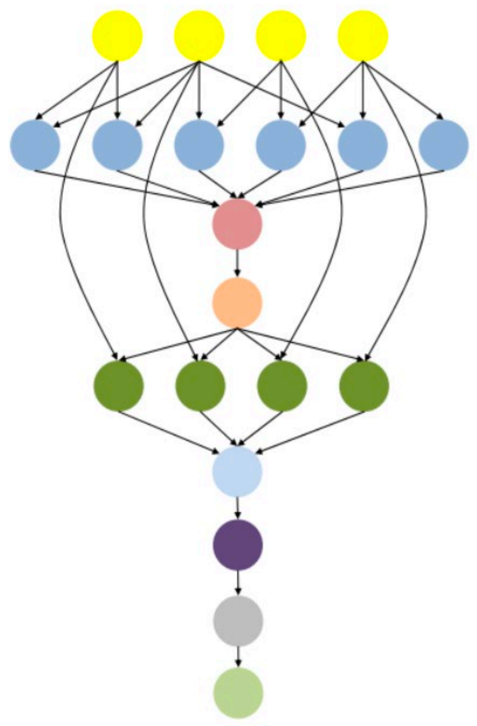
Introduction

- A *workflow* can be any series of activities needed to complete a task
- In scientific computing, a workflow can help with reproducibility by
  - Describing in a formal way how to undertake activities for scientific problem-solving
  - Might be as simple as using a script, or as complex as using a domain specific workflow description language
- A *workflow system* can help with reproducibility by
  - Providing the infrastructure required to execute the activities described by a workflow
  - Tracking the workflow and data provenance
- Workflow concepts and tools make this job easier and will be discussed in this module

Outline

- Workflow concepts
- Example workflow systems
- Common workflow patterns
- Choosing a workflow system
- Reproducibility issues and containers

What are scientific workflows?



HPC workflow == one or more steps require HPC

- Scientific applications often consist of a series of computational or data manipulation steps
- Each step may be written in different languages
  - May use different programming models, numeric methods and libraries
- Steps often have complex dependencies between them
- Steps must be done in a specific sequence to produce correct output
  - Can include cycles
- Steps may produce/consume large amounts of data
  - Data may need to be available from one step to another
- Steps may require interaction from the user

Image source: Al-Haboobi, Ali. (2017). Improving Max-Min scheduling Algorithm for Reducing the Makespan of Workflow Execution in the Cloud. International Journal of Computer Applications. 177. 975-8887. DOI:10.5120/ijca2017915684.

Reasons to use a workflow system

- I have a large complex problem that can be broken down into a number of distinct steps
- I need to scale to a large jobs/systems
- I want to document steps / run method
- I need to be able to track and restart failed components
- I want to be able to reproduce the results or have someone else reproduce them
- I need to track provenance of task execution and data movement
- My application requires building a complicated environment

Workflow activities and components

- **Resource provisioning**
  - Each step of the workflow needs to execute on compute resources
  - These may be preconfigured (e.g. a supercomputer) or dynamically allocated (e.g. cloud)
  - The execution environment needs to be established (e.g. libraries, runtimes, hardware drivers, etc.
- **Data management**
  - Steps may consume data – must be available before execution commences
  - Steps may produce data – how much, where will it be stored?
  - What to do with the data from the workflow?

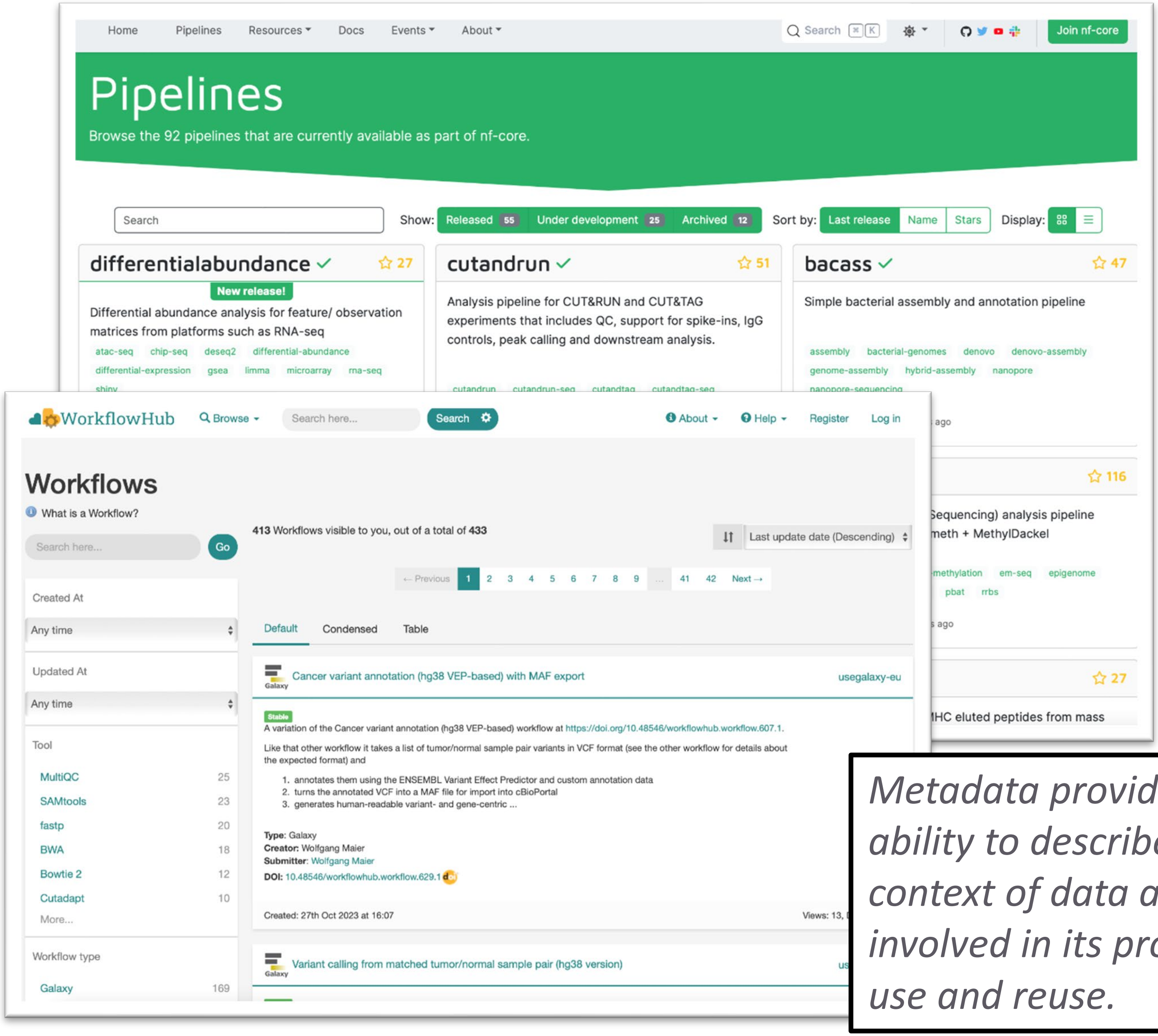
- **Job submission**
  - Access to HPC systems is typically through a batch system
  - These require scripts that determine what resources are required for the job to run
- **Workflow management**
  - Orchestrates the flow of activities needed to carry out the computation
  - Can be driven by data or by tasks
- **Workflow description**
  - Domain specific language describing workflow
  - Many workflow systems employ Common Workflow Language (CWL)

Example workflow systems

https://s.apache.org/existing-workflow-systems

Distributed	In situ
TaskVine cromwell snakemake pegasus COMPSs Merlin Galaxy nextflow pyiron Covalent FireWorks AiiDA	maestro BEE swift

Workflow Repositories and Registries



Home
Pipelines
Resources
Docs
Events
About
Search
Join nf-core

Pipelines

Browse the 92 pipelines that are currently available as part of nf-core.

Search
Show: Released 55 Under development 25 Archived 12
Sort by: Last release Name Stars
Display:

differentialabundance ✓ ☆ 27

New release!

Differential abundance analysis for feature/ observation matrices from platforms such as RNA-seq

atac-seq chip-seq deseq2 differential-abundance differential-expression gsea limma microarray rna-seq shiny

cutandrun ✓ ☆ 51

Analysis pipeline for CUT&RUN and CUT&TAG experiments that includes QC, support for spike-ins, IgG controls, peak calling and downstream analysis.

cutandrun cutandrun-seq cutandtag cutandtag-seq

bacass ✓ ☆ 47

Simple bacterial assembly and annotation pipeline

assembly bacterial-genomes denovo denovo-assembly genome-assembly hybrid-assembly nanopore nanopore-sequencing

ago

☆ 116

Sequencing) analysis pipeline
meth + MethylDackel

methylation em-seq epigenome pbat rrbs

s ago

☆ 27

IHC eluted peptides from mass

Workflows

What is a Workflow?

413 Workflows visible to you, out of a total of 433

Last update date (Descending)
← Previous 1 2 3 4 5 6 7 8 9 ... 41 42 Next →
Default
Condensed
Table

Created At	
Any time	
Updated At	
Any time	
Tool	
MultiQC	25
SAMtools	23
fastp	20
BWA	18
Bowtie 2	12
Cutadapt	10
More...	
Workflow type	
Galaxy	169

Galaxy Cancer variant annotation (hg38 VEP-based) with MAF export usegalaxy-eu

Stable

A variation of the Cancer variant annotation (hg38 VEP-based) workflow at https://doi.org/10.48546/workflowhub.workflow.607.1.

Like that other workflow it takes a list of tumor/normal sample pair variants in VCF format (see the other workflow for details about the expected format) and

1. annotates them using the ENSEMBL Variant Effect Predictor and custom annotation data
2. turns the annotated VCF into a MAF file for import into cBioPortal
3. generates human-readable variant- and gene-centric ...

Type: Galaxy
Creator: Wolfgang Maier
Submitter: Wolfgang Maier
DOI: 10.48546/workflowhub.workflow.629.1

Created: 27th Oct 2023 at 16:07 Views: 13,

Galaxy Variant calling from matched tumor/normal sample pair (hg38 version) us

- Registry: indexes or catalogs of workflow metadata
- Repository: workflow descriptions and metadata are indexed and stored
- Metadata framework: help support reproducibility by managing metadata
- Data repository: permanently stores and manages data; provides digital object identifier (DOI)

Metadata provides the ability to describe the context of data and entities involved in its production, use and reuse.

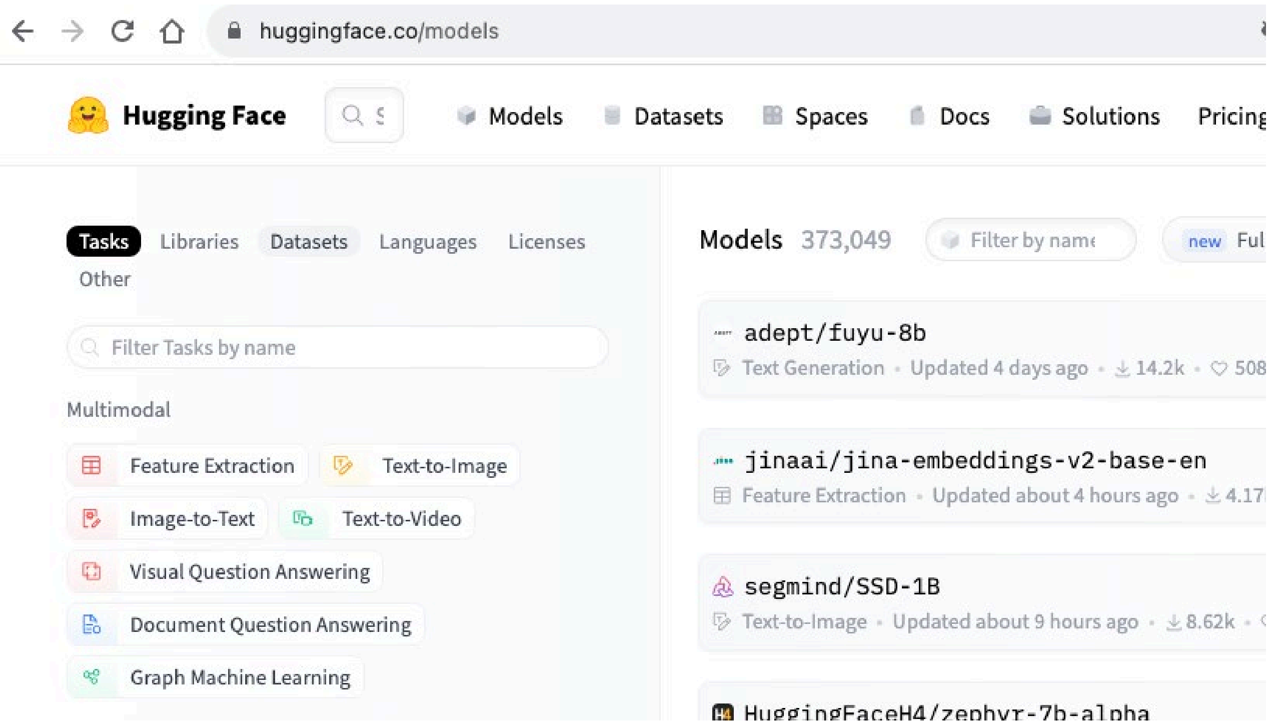
Implementations:

https://nf-co.re
https://workflowhub.eu
https://zenodo.org

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Data / Code Combinator



huggingface.co/models

Hugging Face

Models · Datasets · Spaces · Docs · Solutions · Pricing

Tasks · Libraries · Datasets · Languages · Licenses · Other

Filter Tasks by name

Multimodal

- Feature Extraction
- Text-to-Image
- Image-to-Text
- Text-to-Video
- Visual Question Answering
- Document Question Answering
- Graph Machine Learning

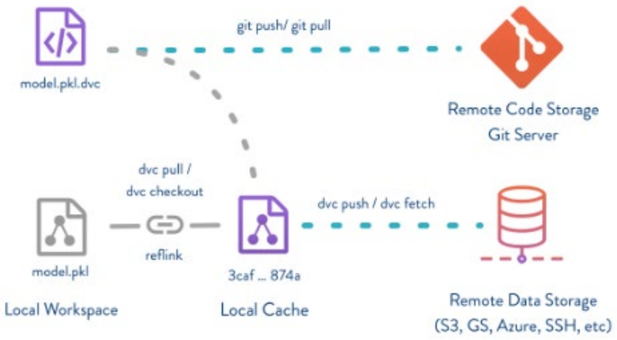
Models 373,049 · Filter by name · new Ful

adept/fuyu-8b
Text Generation • Updated 4 days ago • 14.2k • 508

jinaai/jina-embeddings-v2-base-en
Feature Extraction • Updated about 4 hours ago • 4.17

segmind/SSD-1B
Text-to-Image • Updated about 9 hours ago • 8.62k

HuggingFaceH4/zephyr-7b-alpha



- Publish data and structured code from within your code
- Offers tracking functionality for provenance (code / experiment progression)
- Popular for AI/ML models, where it facilitates model comparisons

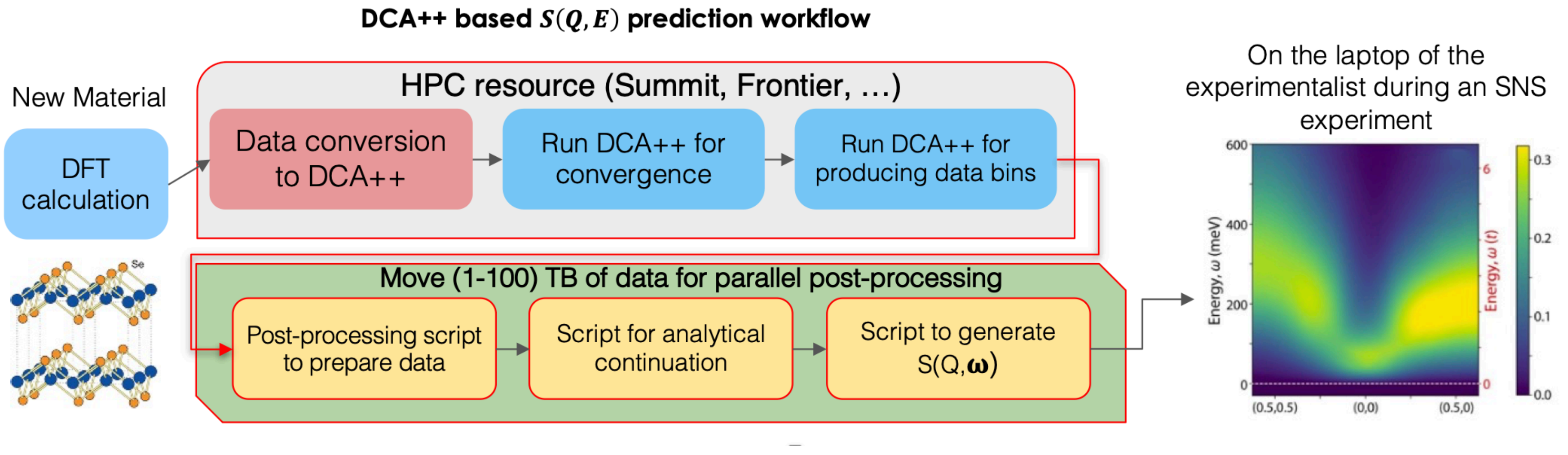
Implementations:

pypi / dvc (DVC.org)
Weights & Biases (wandb.ai)
huggingface.co

Common scientific workflow patterns

High-performance Simulation and Modeling

Figure 4. Diagram of the S(Q,ω) prediction workflow. The workflow begins with a DFT calculation based on the new material of interest to build a low-energy effective model. A first DCA++ simulation is performed for convergence to self-consistency. For the problems of interest here, which require O(1000) bins of data for the four-point vertex function, a second DCA++ simulation is run to produce the binned data. The relevant observables are extracted and manipulated into a convenient form using several Python scripts. This includes the extraction of the vertex function Γ, the solution of the Bethe-Salpeter equation to generate S(Q,ω) on the imaginary frequency axis, and the analytic continuation to perform the rotation to the real frequency axis. The final product of the Python analysis is the prediction for S(Q,ω) .


DCA++ based $S(Q,E)$ prediction workflow
New Material
DFT calculation
Se
HPC resource (Summit, Frontier, …)
Data conversion to DCA++
Run DCA++ for convergence
Run DCA++ for producing data bins
Move (1-100) TB of data for parallel post-processing
Post-processing script to prepare data
Script for analytical continuation
Script to generate S(Q,$\boldsymbol{\omega}$)
On the laptop of the experimentalist during an SNS experiment
Energy, ω (meV)
600
400
200
0
Energy, ω (t)
6
0
0.3
0.2
0.1
0.0
(0.5,0.5)
(0,0)
(0.5,0)

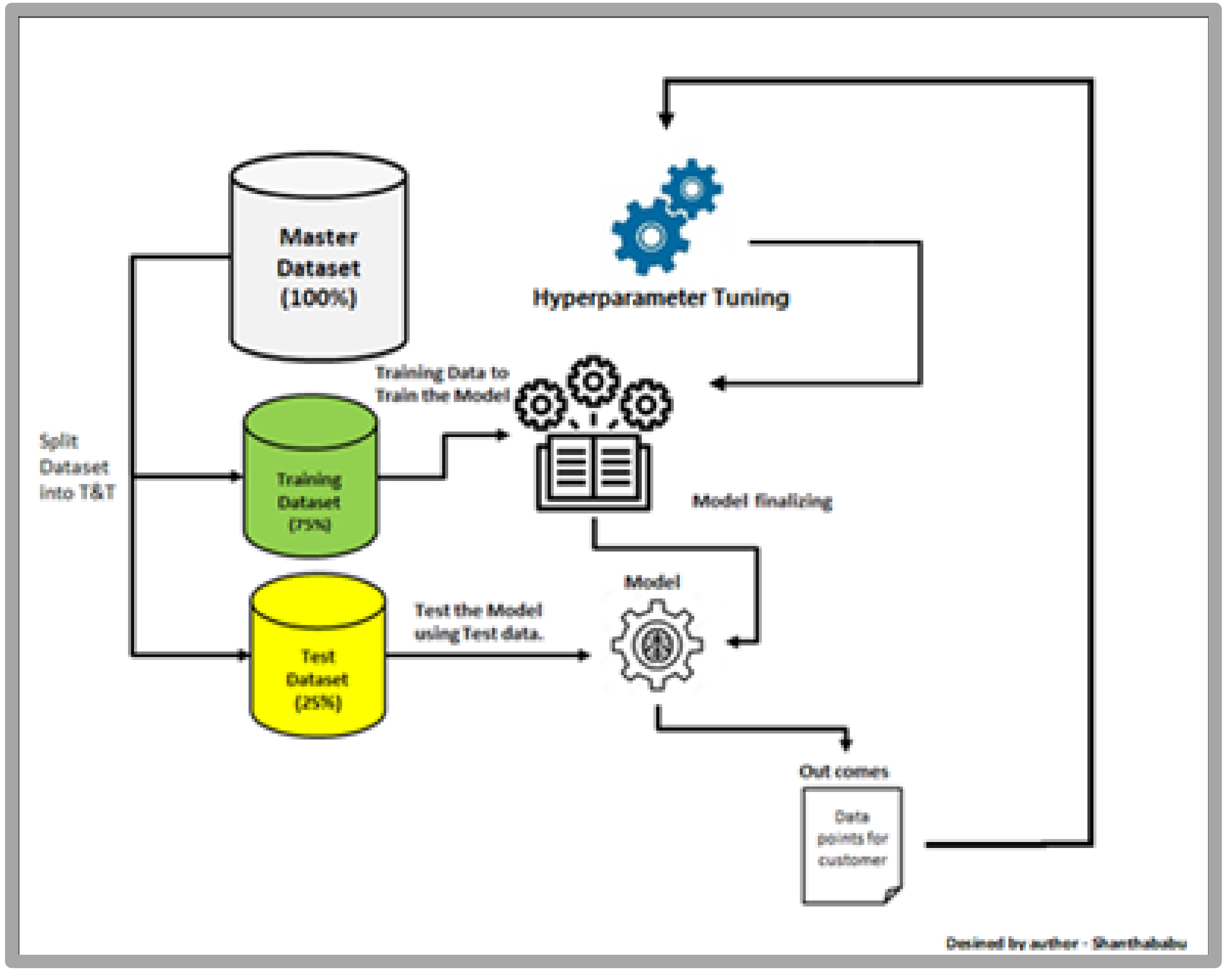
Source: https://doi.org/10.1007/978-3-031-23606-8_9

- Setup: mesh generation, pre-computation
- Simulate: physics calculations
- Down-sample: data reduction
- Post-process: format for visualization tools
- Visualize: inspect the data

Common scientific workflow patterns

High-performance AI

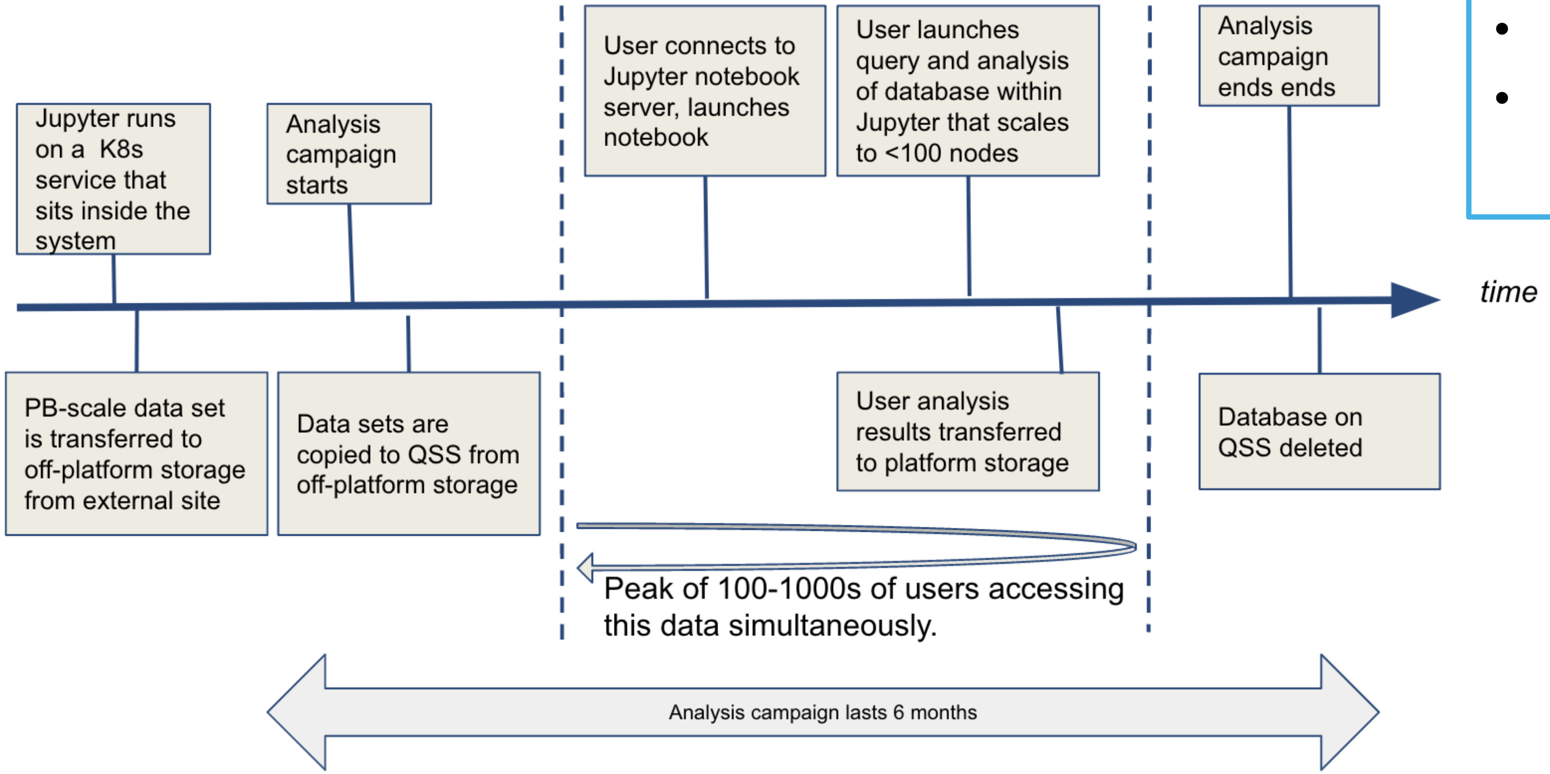
- Training: AI model training using scientific data
- Inference: Use the model to make predications
- Hyperparameter optimization: Find the best combination of hyperparameters for the problem



Source: https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

Common scientific workflow patterns
Scientific Data Lifecycle

- Interactive data-analytics: running data intensive workflows against shared datasets
- Visualization: flexible visualizations depending on requirements
- Creation of summary datasets
- Workflow replication/modification



Jupyter runs on a K8s service that sits inside the system

Analysis campaign starts

User connects to Jupyter notebook server, launches notebook

User launches query and analysis of database within Jupyter that scales to <100 nodes

Analysis campaign ends ends

PB-scale data set is transferred to off-platform storage from external site

Data sets are copied to QSS from off-platform storage

User analysis results transferred to platform storage

Database on QSS deleted

Peak of 100-1000s of users accessing this data simultaneously.

Analysis campaign lasts 6 months

Source: https://www.nersc.gov/assets/NERSC-10/Workflows-Archetypes-White-Paper-v1.0.pdf

Common scientific workflow patterns

Real-Time / Streaming

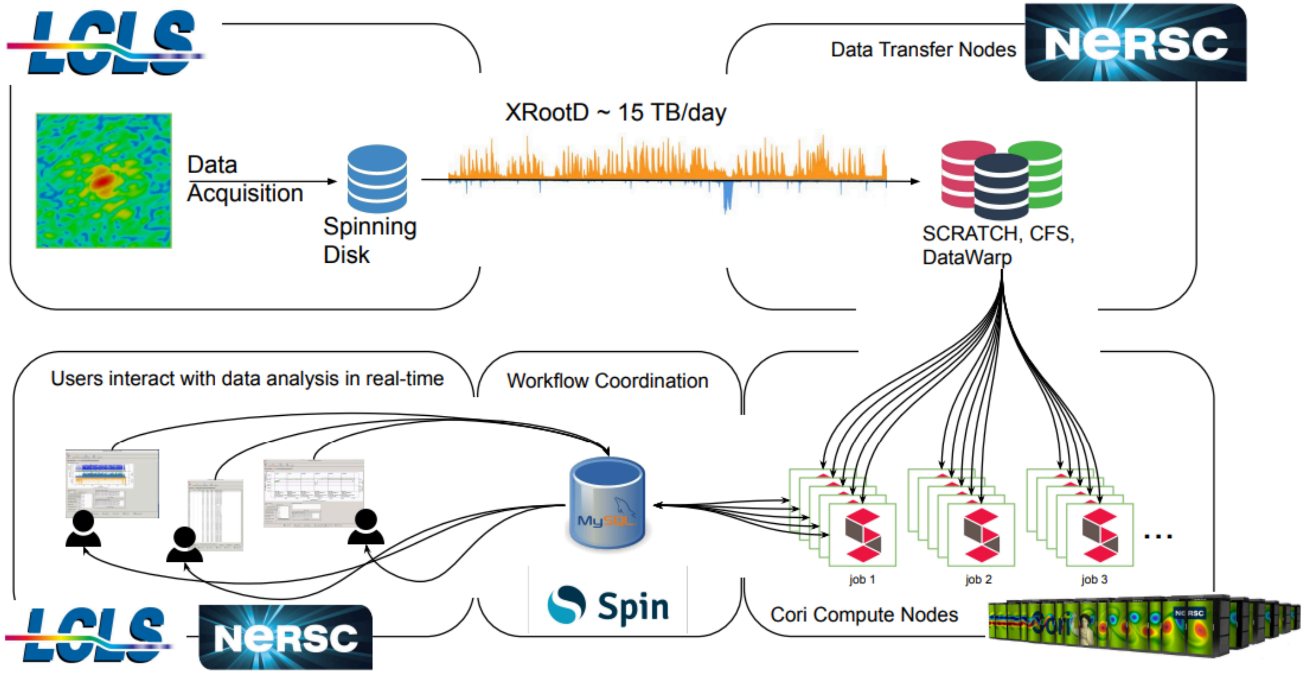


Fig. 1. Sketch of the super-facility workflow: *Top:* Data are automatically transferred from the LCLS spinning-disk storage system via XRootD to NERSC's Scratch file system (the orange and blue spikes show the data transfer rate – approx. 1.3-2.6 GB/s – over the ESNet network, with each spike being a completed run). *Bottom*: On NERSC the CCTBX workers (running in Shifter containers on the Cori compute nodes) automatically analyze new data on Scratch, using the DataWarp burst buffer as a cache. Users at LCLS and NERSC connect to a mySQL database hosted at NERSC to orchestrate the workers, review the data analysis and iterate analysis parameters.

Real-Time XFEL Data Analysis at SLAC and NERSC: a Trial Run of Nascent Exascale Experimental Data Analysis. DOI:10.48550/arXiv.2106.11469
Johannes P. Blaschke, Aaron S. Brewster, Daniel W. Paley, Derek Mendez, Asmit Bhowmick, Nicholas K. Sauter, Wilko Kröger, Murali Shankar, Bjoern Enders, Deborah Bard

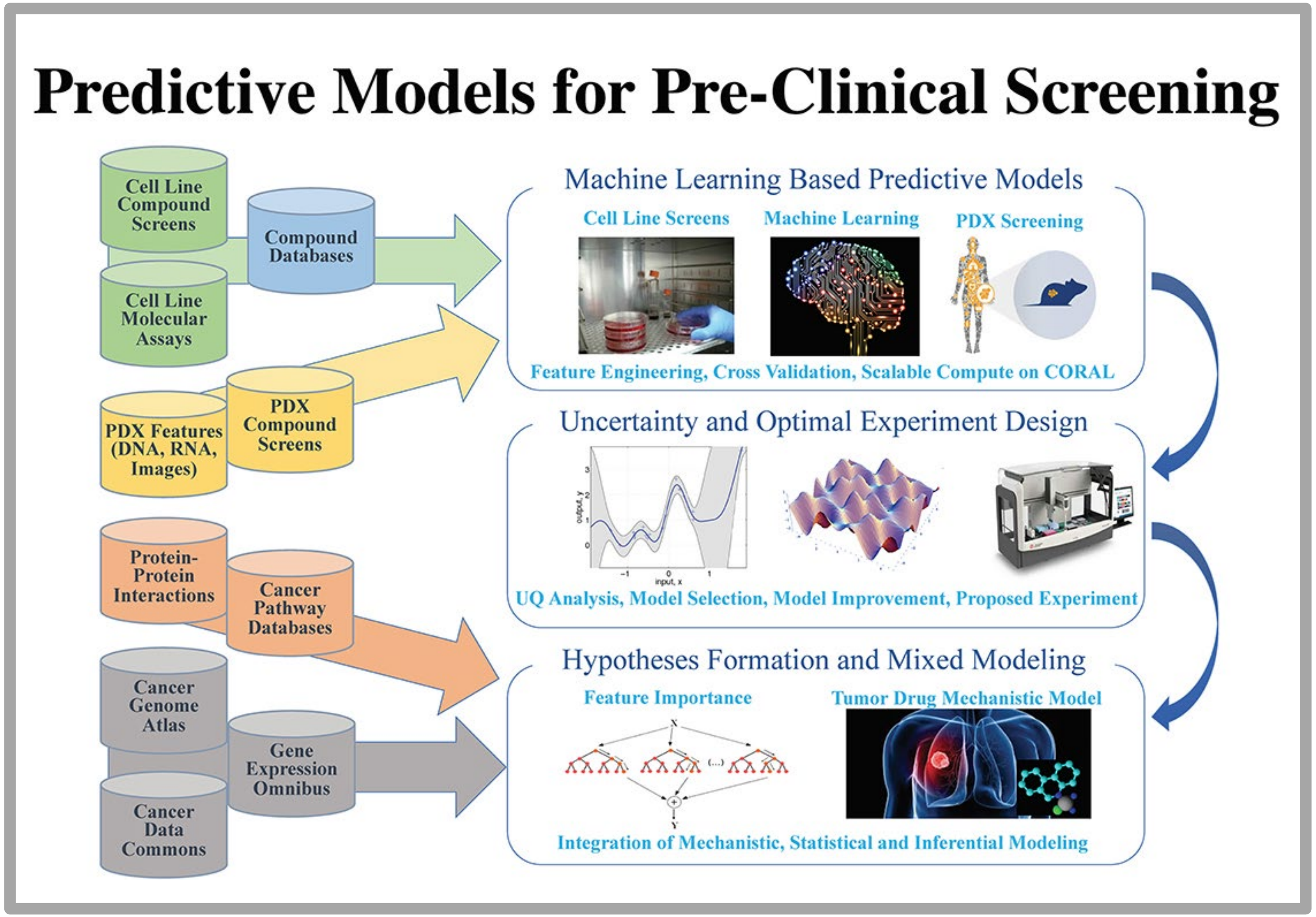
- Data transport
- Compression
- Visualization
- Buffering / final storage

Implementations:

XRootD
pypi / ray
Apache kafka
Globus
custom...

Common scientific workflow patterns
Hybrid

- Combines elements of multiple workflow patterns


Predictive Models for Pre-Clinical Screening
Cell Line Compound Screens
Compound Databases
Cell Line Molecular Assays
PDX Features (DNA, RNA, Images)
PDX Compound Screens
Protein-Protein Interactions
Cancer Pathway Databases
Cancer Genome Atlas
Gene Expression Omnibus
Cancer Data Commons
Machine Learning Based Predictive Models
Cell Line Screens
Machine Learning
PDX Screening
Feature Engineering, Cross Validation, Scalable Compute on CORAL
Uncertainty and Optimal Experiment Design
UQ Analysis, Model Selection, Model Improvement, Proposed Experiment
Hypotheses Formation and Mixed Modeling
Feature Importance
Tumor Drug Mechanistic Model
Integration of Mechanistic, Statistical and Inferential Modeling

Bhattacharya Tanmoy, Brettin Thomas, Doroshow James H., Evrard Yvonne A., Greenspan Emily J., Gryshuk Amy L., Hoang Thuc T., Lauzon Carolyn B. Vea, Nissley Dwight, Penberthy Lynne, Stahlberg Eric, Stevens Rick, Streitz Fred, Tourassi Georgia, Xia Fangfang, Zaki George, “AI Meets Exascale Computing: Advancing Cancer Research With Large-Scale High Performance Computing”, Frontiers in Oncology, vol 9, 2019, DOI:10.3389/fonc.2019.00984

Make-like (acyclic)

```
prot_dcd:
  resource:
    time: 1.0
    cpu: 4
    nrs: 1
  inp:
    - Production/remd_{n}/prot.xtc
    - Production/remd_{n}/prot.tpr
  out:
    - Production/remd_{n}/prot.psf
    - Production/remd_{n}/prot.dcd
  script: |
    . /ccs/proj/bif128/setup-env-vmd.sh
    {mpirun} vmd -dispdev text \
      -e $SRC/vmd/rewrap.tcl \
      -args {inp[0]} {inp[1]} \
           {out[0]} {out[1]}
```

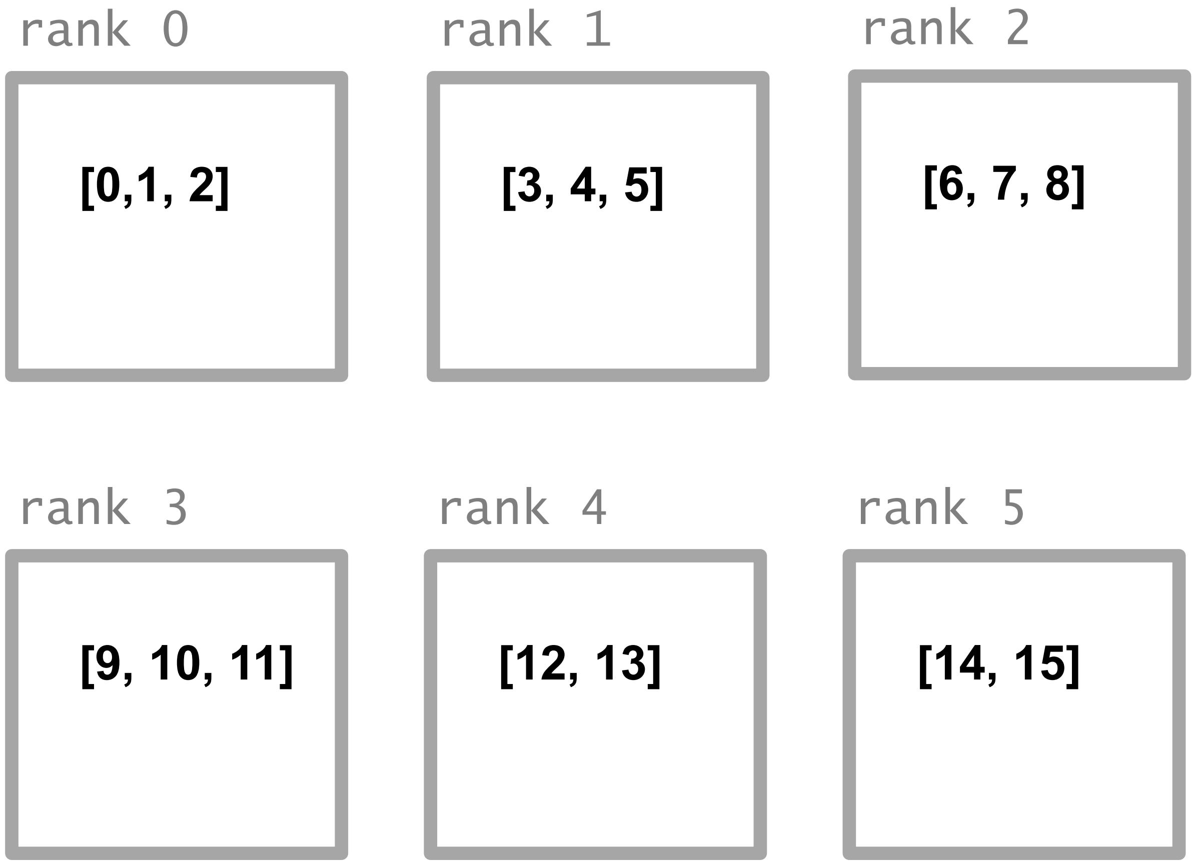
Typical characteristics:

- List of a -> b conversion rules
- File presence = workflow state
- Shell-like variable substitution
- Can be auto-parallelized

Implementations:

GNU make
plan9 mk
pmake
snakemake
fireworks
swift/T
nextflow.io
DPLASMA

Workflow-like Programming Models: Map-Reduce



rank 0

[0,1, 2]

rank 1

[3, 4, 5]

rank 2

[6, 7, 8]

rank 3

[9, 10, 11]

rank 4

[12, 13]

rank 5

[14, 15]

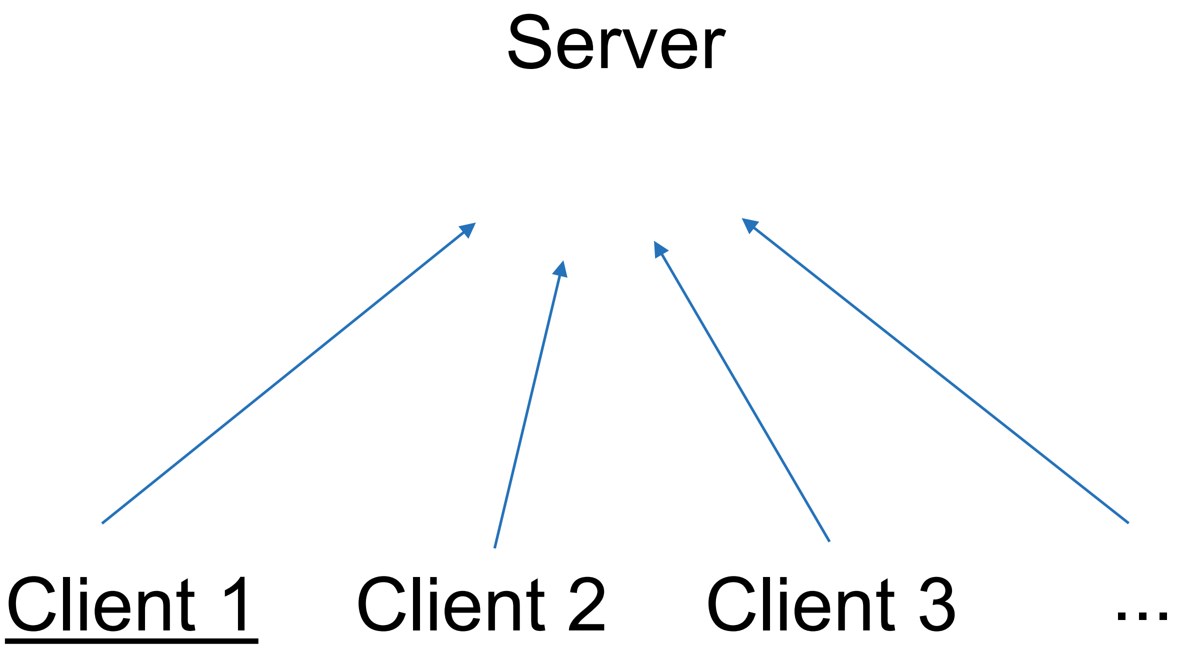
- Map: run function over all data items
- Reduce: aggregate results (sum/etc.)
- Scan: calculating running reduce
- Collect: gather info. to root rank
- Filter: conditionally remove elements
- Repartition/regroup: reorder elements

Implementations:

module / mpi
GNU parallel

pypi / mpi_list
pypi / pyspark / pysparkling
pypi / mpi4py

Workflow-like Execution Models: Bag of Tasks



1. Request task from server
2. Do work and output
3. Inform server of completion

Server stores a set of "tasks" and exports an API:

- Add task
- Fetch task
- Mark task complete
- May also track DAG state
- May be "push" instead of "pull"

Implementations:

pypi / parsl
pypi / celery
pypi / dask
RADICAL/pilot
Apache Airflow
Ruby resque
Hashicorp Nomad
Slurm / LSF / etc.
github.com/frobnitzem/dwork

Factors to consider

- Is it available on my system (discuss with your HPC center)
- How does it handle data provenance?
- Are there workflows that already do what I want?
- What are the computational requirements for each component/step?
  - Which need HPC?
  - Could I just use cloud vs. HPC (scale / other constraints)
- What data requirements does my problem have?
  - How much data is consumed/produced by each step?
  - Where do data transfers need to happen?

Reproducibility challenges for HPC systems

- Library and software versions are changing constantly so it is difficult to know what versions were used at build time vs run time
  - Can be changed by the user (e.g. modules) or by system admins
  - Hotfixes for system libraries
- Kernel hardware drivers can change and affect behavior
- Files on the high-performance filesystems are "purged" using a time-based algorithm. So, as data ages, it disappears unless moved to a stable location
- Hardware may experience rare, but random system failures, terminating an otherwise correct run early or, in the worst case, silently corrupting data
- Other jobs running on the system may interfere with the network or I/O performance, leading to unpredictable amounts of latency

Containers

- Containers are one way to reduce portability issues, however can come with a performance penalty
- Used to package up all code and dependencies necessary to execute single process or task
- Encapsulates the entire software ecosystem (minus the kernel)
- Uses a host operating system virtualization mechanism (not virtual machine)
- Common container systems: Docker (the first); Apptainer (was singularity); Podman (RedHat); Charliecloud (LANL); Shifter (NERSC); Sarus (CSCS)

Containers (cont…)

- Originally designed for Cloud
- Docker not a good fit for HPC
  - Requires root-level access
  - Architecture does not suit most HPC installations
- Other container technologies are more tailored to HPC
- Growing HPC center support (Apptainer/Singularity, Charliecloud, Shifter, Sarus)
- May actually decrease run-time for python apps (via reduced dependency load time)
- Utilized by GitHub actions/GitLab pipelines

Open Container Initiative (OCI)

https://opencontainers.org

- Industry standard for container formats and runtimes
- Allows containers built using one container system to be run with another
  - E.g. build with Podman and run using Singularity
- Containers can be stored in different registries
- Three specificiations:
  - Runtime Specification (runtime-spec)
  - Image Specification (image-spec)
  - Distribution Specification (distribution-spec)

Containers supported by HPC systems[1]

- ALCF
  - Theta: Singularity
  - Aurora: Singularity
- OLCF
  - Summit: Singularity
  - Frontier: Singularity
- NERSC
  - Cori: Shifter
  - Perlmutter: Shifter, Singularity, Podman

- LLNL
  - Sierra/Lassen: Singularity
  - El Capitan: Singularity
- LANL
  - Trinity: Charliecloud
  - Linux clusters: Charliecloud
  - Crossroads: Charliecloud
- SNL
  - Astra: Singularity, Charliecloud, Podman
  - Linux clusters: Shifter, Singularity

[1] Source: Containers and the Truth between HPC & Cloud System Software Convergence. 2021. DOI:10.2172/1859696

Issues using containers in HPC

- MPI with HPC containers is a mess
  - Certain vendors don't support MPI libraries via a container
  - Users have to match MPI build to host MPI configurations
  - There are solutions but at the expense of portability and specific to implementation
- Build environment does not match run environment
  - Supporting non-x86 machines (eg: Summit, Astra, ...) requires special build nodes
  - Even x86 is no guarantee of performance or portability
  - Many libraries make build-time assumptions which can be bad
- Job life is a batch in HPC
  - Containers & MPI libraries don't default to playing well with schedulers
  - Module settings (environment variables) can leach into containers

Issues using containers in HPC (cont…)

- Most HPC Container runtimes support GPUs
  - `singularity run --nv ...`
  - `shifter --module=cuda-10.1 ...`
  - `ch-fromhost --nvidia ...`
- Variation between GPUs
  - `singularity run --nv ...`
  - `singularity run --rocm ...`
- Relatively little integration to date with Intel GPUs
  - E4S image with OneAPI is planned for release next year
- None of this looks like the Cloud

Summary

- Workflow systems provide a powerful tool to undertake scientific discovery
  - High-performance Simulation and Modeling; High performance AI; Scientific Data Lifecycle; Real-Time/Streaming; Hybrid
- There are a large number of systems available
  - Choosing the right one depends on the needs of the application (computational, data, reproducibility) and what is available locally
- Workflows help with reproducibility, there are resources that help with sharing workflow codes and data
- HPC has many reproducibility issues
  - Containers help with portability, and most HPC systems support containers
  - HPC and containers are not currently a great fit

Other resources

Workflow systems

- AiiDA: https://www.aiida.net
- BEE: https://github.com/lanl/BEE
- COMPSs: https://compss.bsc.es
- Covalent: https://www.covalent.xyz
- Cromwell: http://cromwell.readthedocs.io
- FireWorks: https://materialsproject.github.io/fireworks
- Galaxy: https://galaxyproject.org
- Maesto: https://maestrowf.readthedocs.io
- Nextflow: https://www.nextflow.io
- Pegasus: https://pegasus.isi.edu
- Snakemake: https://snakemake.github.io
- Swift: http://swift-lang.org/Swift-T
- Taskvine: https://ccl.cse.nd.edu/software/taskvine

Containers

- Apptainer (singularity) https://apptainer.org
- Charliecloud: https://hpc.github.io/charliecloud
- Docker: https://docker.com
- Podman: https://podman.io
- Sarus: https://sarus.readthedocs.io/en/stable
- Shifter: https://shifter.readthedocs.io/en/latest

Other resources

Workflow repositories

- @nf-core (Nextflow)
  - https://nf-co.re
- Snakemake workflow catalog
  - https://snakemake.github.io/snakemake-workflow-catalog

Workflow registries

- WorkflowHub
  - https://workflowhub.eu
- Dockstore
  - https://dockstore.org

Metadata frameworks

- Common workflow language
  - https://www.commonwl.org
- Workflow RO-Crate
  - https://w3id.org/workflowhub/workflow-ro-crate/1.0
- Bioschemas Profiles
  - https://bioschemas.org/profiles

Data repositories

- Zenodo
  - https://zenodo.org
- Dataverse
  - https://dataverse.org

Other resources

- C. Harris, P. O'Leary, M. Grauer, A. Chaudhary, C. Kotfila and R. O'Bara, "Dynamic Provisioning and Execution of HPC Workflows Using Python," 2016 6th Workshop on Python for High-Performance and Scientific Computing (PyHPC), Salt Lake City, UT, USA, 2016, pp. 1-8, doi: 10.1109/PyHPC.2016.005.
- S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. -H. Su and K. Vahi, "Characterization of scientific workflows," *2008 Third Workshop on Workflows in Support of Large-Scale Science*, Austin, TX, USA, 2008, pp. 1-10, doi:10.1109/WORKS.2008.4723958.
- Blaschke, Johannes P., Brewster, Aaron S., et. al. 2021. "Real-Time XFEL Data Analysis at SLAC and NERSC: a Trial Run of Nascent Exascale Experimental Data Analysis". United States. https://www.osti.gov/servlets/purl/1827927.
- Younge, Andrew J. 2021. "Containers and the Truth between HPC & Cloud System Software Convergence.". United States. DOI:10.2172/1859696.