# Research Without Borders: High Performance Computing for Discovery Across Domains

Tainã Coleman, Ph.D.

# Tainã Coleman

Schmidt AI in Science Postdoctoral Fellow at San Diego Supercomputer Center



## About Me

- From Brazil
- BS in Computer Engineering from Universidade Federal de Itajubá (2016)
- MS Computer Science from California State University, Long Beach (2020)
- Ph.D. Computer Science from University of Southern California (USC) (2023)

## Research Interests

- Workflows:
  - Effect of workflow structures in HPC environments
  - Developed algorithms, benchmarks, and data-driven approaches
- AI Workflows for AI in various domains:
  - Tools to enable domain researchers easier access to AI technology.

# Overview



Main challenges of HPC

What are Workflows

Scientific Workflows

Synthetic Workflow Generation

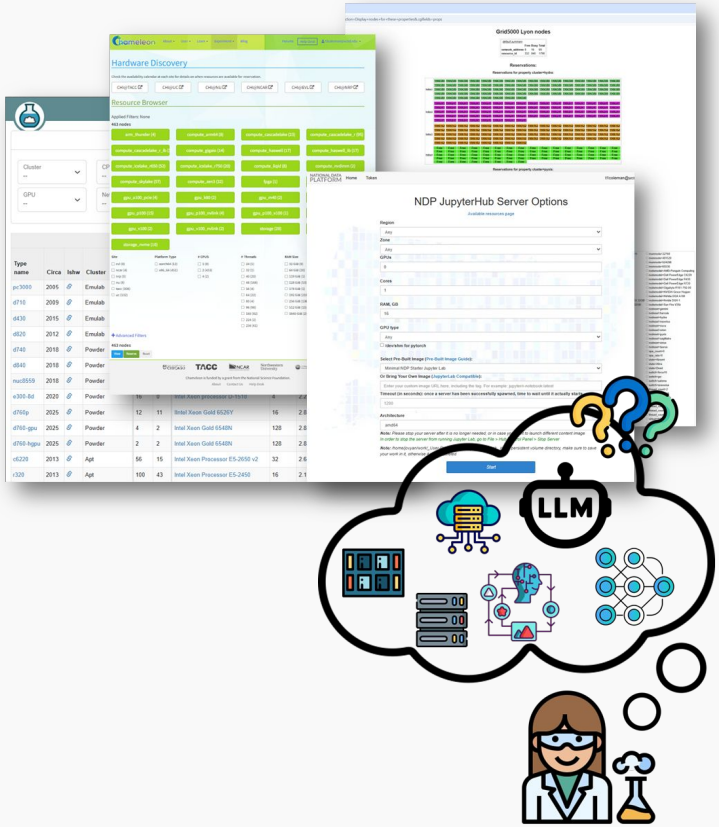
Workflow Benchmarks

AI Workflows of AI in Various Domains

Beyond the Sciences

# Challenges Computation in the Eyes of a Domain Scientist

- Domain applications are extremely complex
- Many domain scientists face significant barriers incorporating distributed computation and AI to their research.
- The **variety** of platforms, models, and processes generates unrealistic expectations.
- **Good** choices are a combination of formal training and tacit expertise.
- **Wrong** choices → slow jobs, wasted compute, unhappy users.
- Tacit expertise comes from years of experience and lots of trial and error.



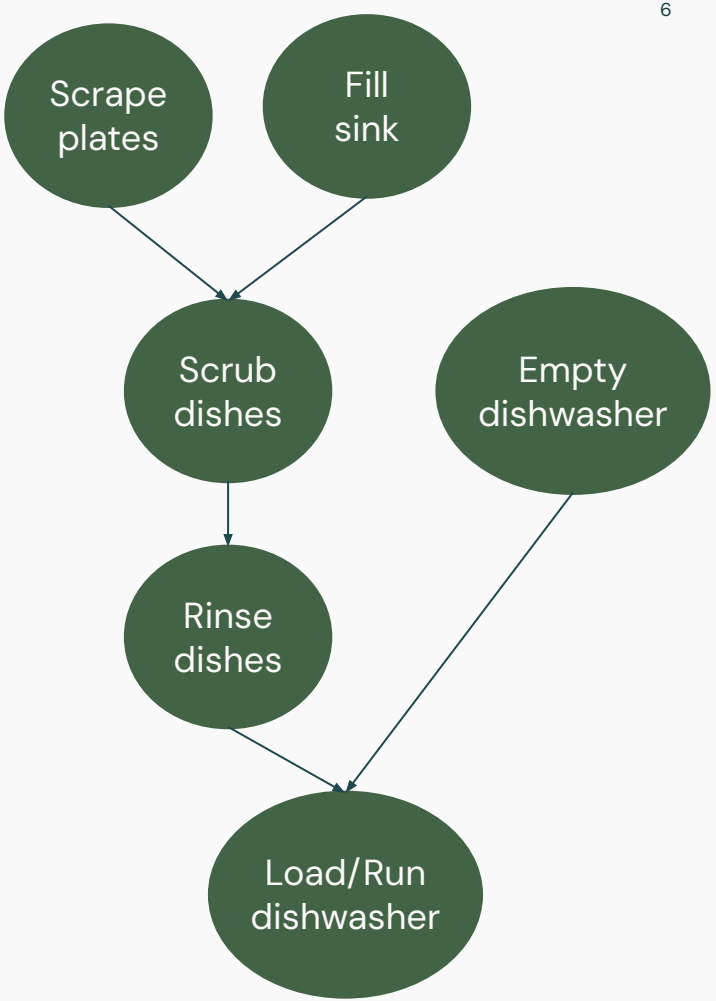
# Part I



# Workflows Background

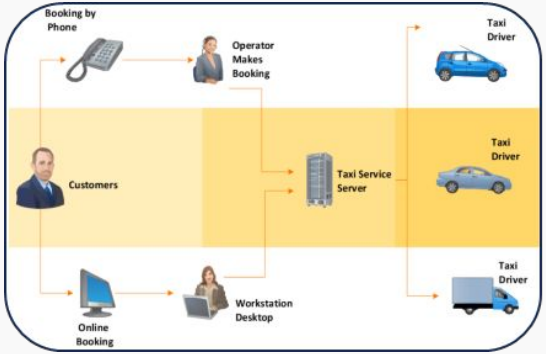
# What are Workflows?

- Workflows are a collection of atomic tasks connect by some type of  dependency
- They are used to describe events that need to respect a sequence.
- Simple example: **Dishwashing**
  - Scrape the food out of dishes
  - Fill the sink with hot soapy water
  - Scrub the dishes
  - Empty dishwasher
  - Rinse
  - Load dishwasher

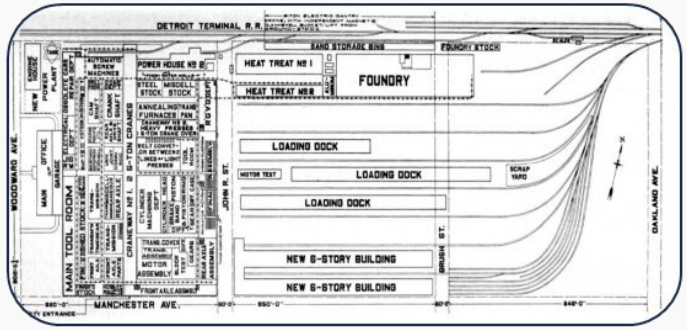


# What are Workflows?

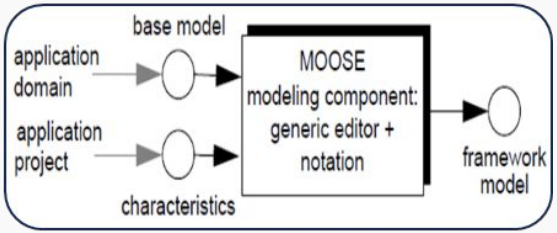
Business Workflows




Engineering/ Manufacturing Workflows




Database Workflows

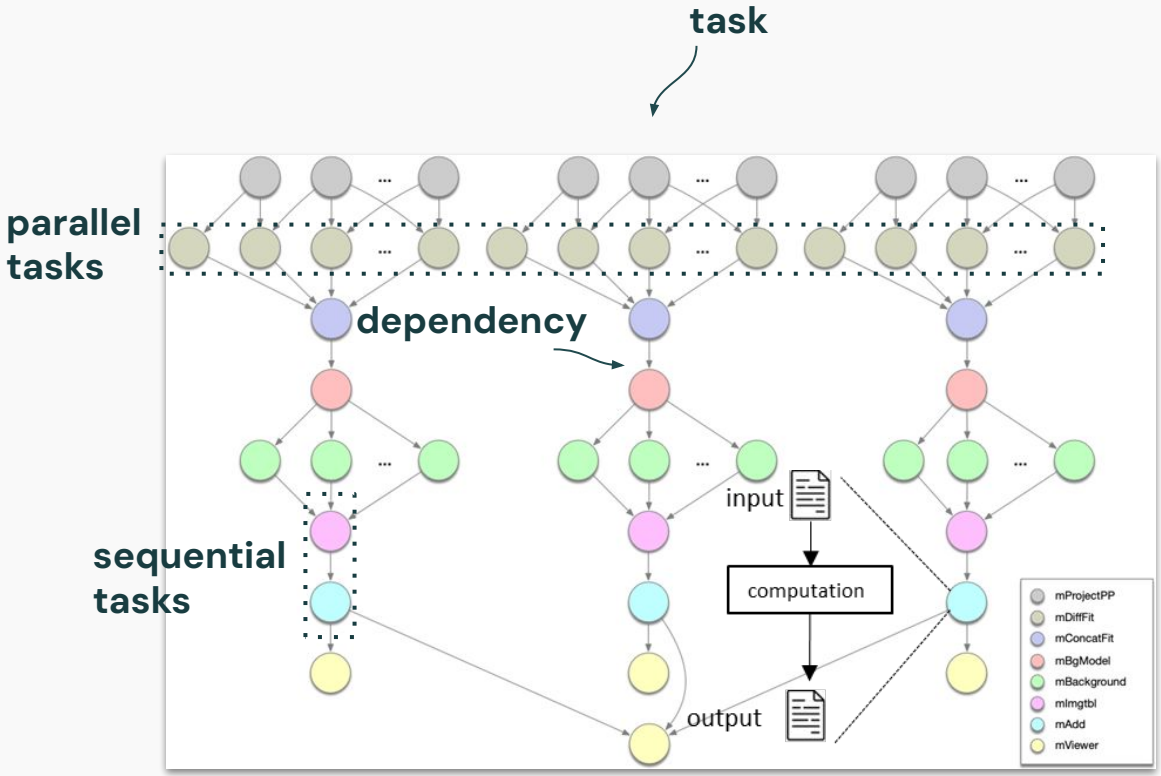



Scientific Workflows

# Scientific Workflows

# What are Scientific Workflows?

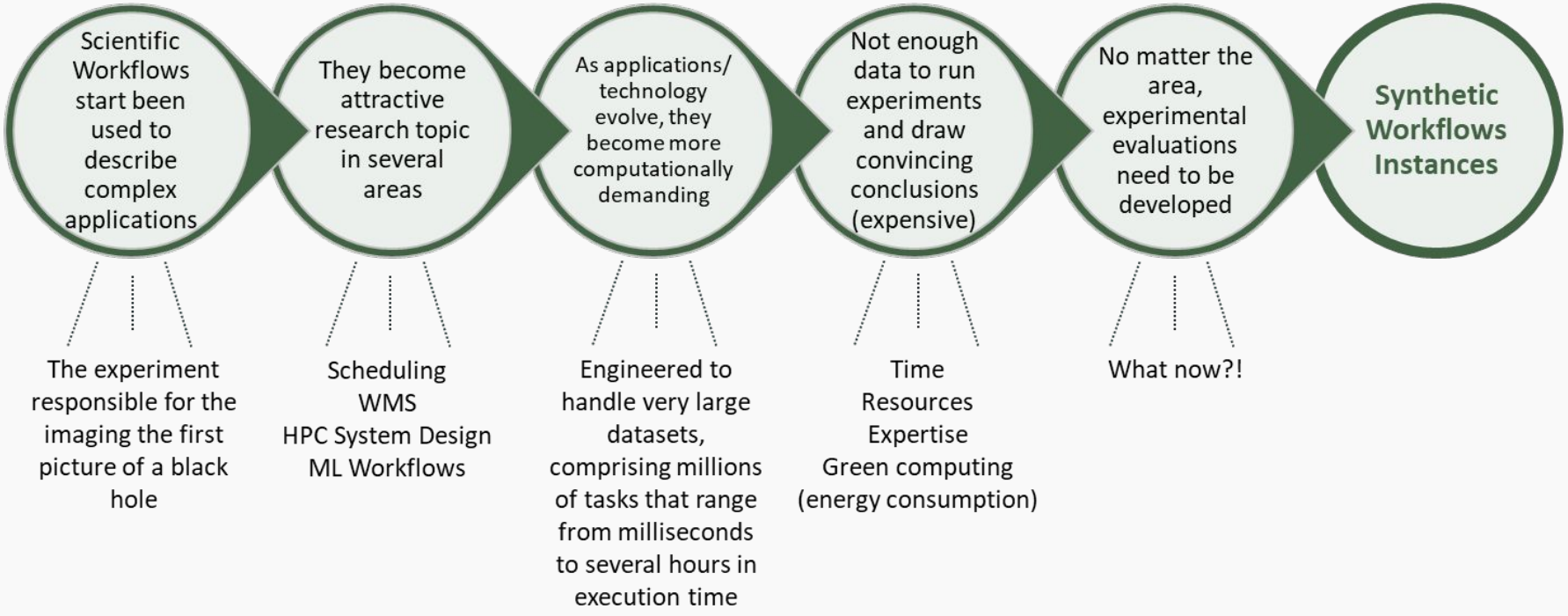
**Scientific Workflows** are task graphs that describe a scientific application.




## Advantages

Provide abstraction for effective resource allocation

Development of robust problems


Scientific Workflows start been used to describe complex applications
They become attractive research topic in several areas
As applications/ technology evolve, they become more computationally demanding
Not enough data to run experiments and draw convincing conclusions (expensive)
No matter the area, experimental evaluations need to be developed
Synthetic Workflows Instances
The experiment responsible for the imaging the first picture of a black hole
Scheduling
WMS
HPC System Design
ML Workflows
Engineered to handle very large datasets, comprising millions of tasks that range from milliseconds to several hours in execution time
Time
Resources
Expertise
Green computing (energy consumption)
What now?!

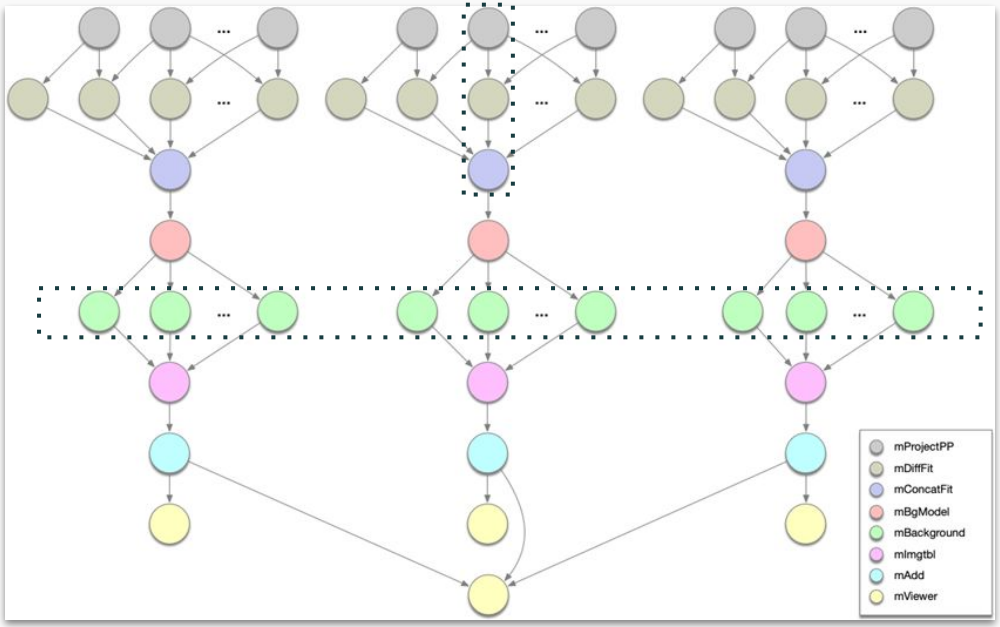
# Part I



# Synthetic Workflow Generation

# Synthetic Workflow Generation

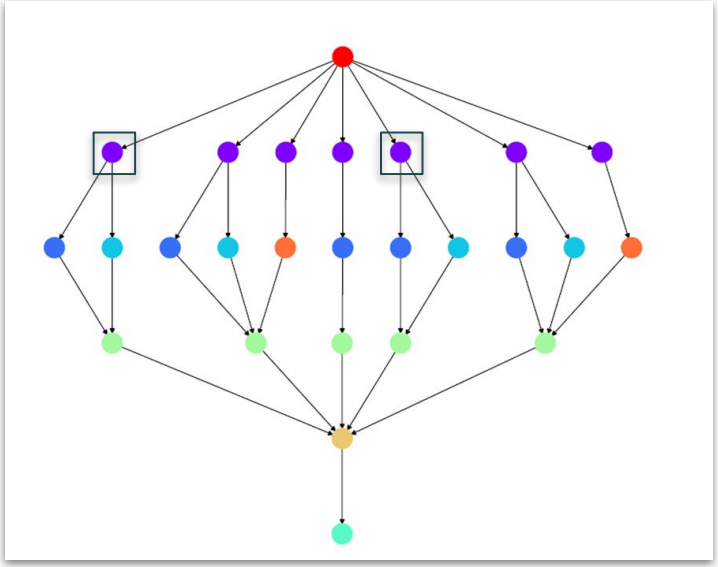
- Real workflows show repeated patterns.
- Synthetic Workflow Generators can leverage them to produce more realistic instances.



# WfChef

- Automated generator of workflow generators
- Collection of algorithms to create realistic synthetic workflow instances of any size
- Inputs
  - Set of real-world workflow instances
  - Desired instance size (number of tasks)
- Analyzes the instances
- Finds common patterns
- Produces a recipe for the application
- Replicate patterns to produce new graphs with desired size

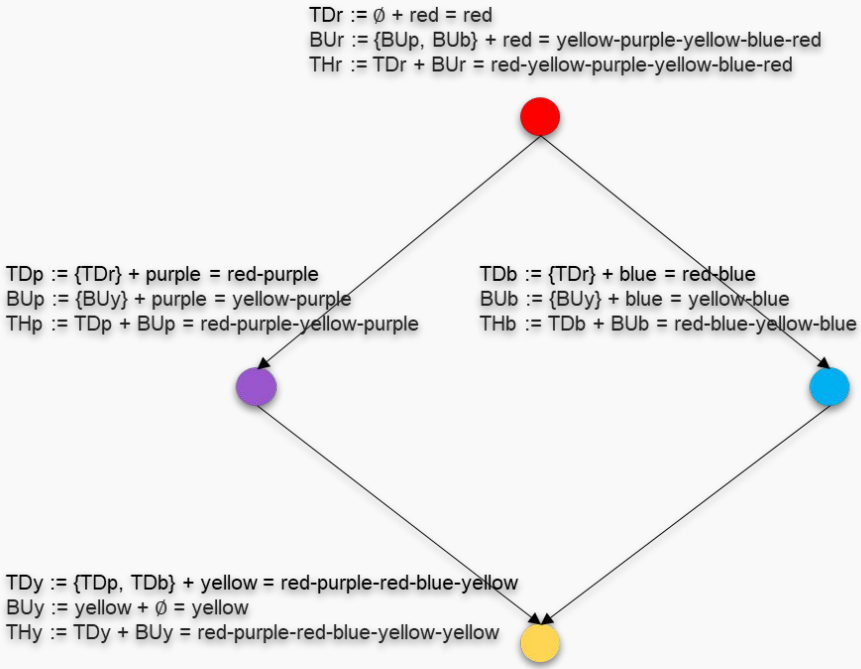
**Task Type**
Same type of computation



WfChef

## Type Hash

- Top Down (TD)
- Bottom Up (BU)
- Task Type
- Type Hash = TD + BU



# WfChef

## WfChefRecipe

Input: Set of real instances (W)
Output: Data structure that encodes information extracted

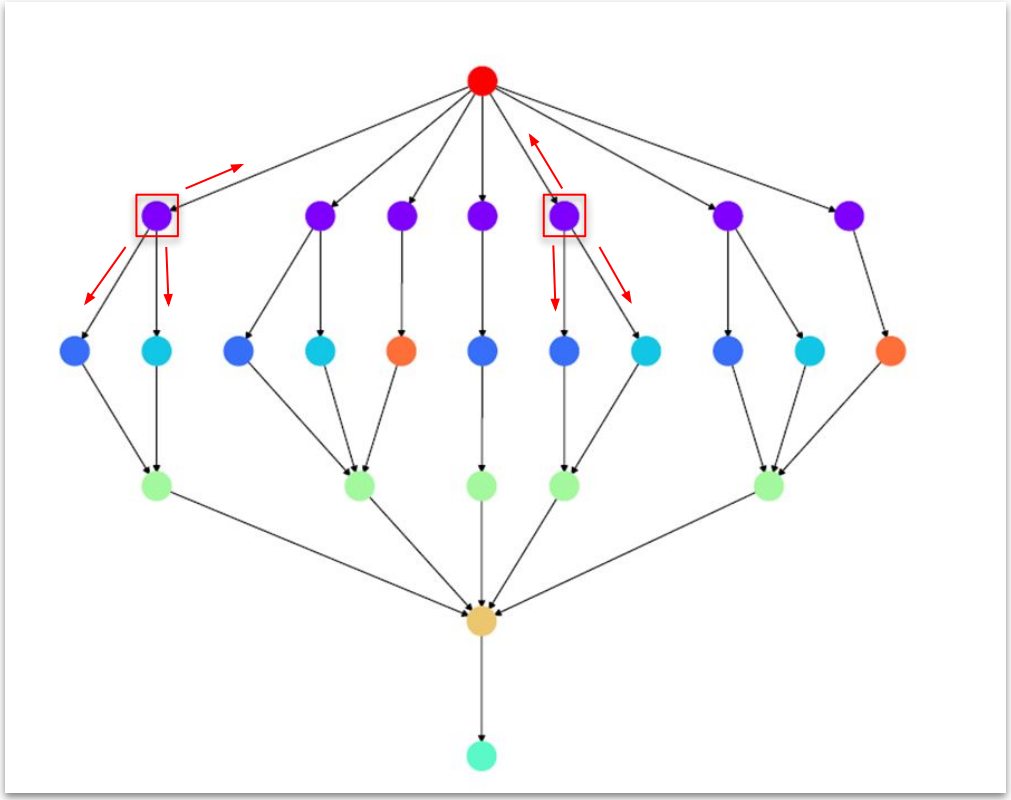
**Algorithm 1** Algorithm to compute a recipe based on a set of real workflow instances.

```
1:  function WfChefRecipe(W)
2:      POs ← {}                                   ▷ dictionary of POs
3:      for each w ∈ W do
4:          POs[w] ← []                            ▷ list of POs in w
5:          for each unvisited vertex v in w do
6:              mark v as visited
7:              v' = an unvisited vertex s.t. TH(v') = TH(v)
8:              if v' is not found then continue
9:              mark v' as visited
10:             A = ClosestCommonAncestors(v,v')
11:             D = ClosestCommonDescendants(v,v')
12:             if A = ∅ or B = ∅ continue
13:             POs[w].append(SubDAG(v, A, B))
14:             POs[w].append(SubDAG(v', A, B))
15:         end for
16:     end for
17:     Errors ← {}                                ▷ dictionary of errors
18:     for each w ∈ W do
19:         for each b ∈ W s.t. |b| < |w| do
20:             g ← ReplicatePOs (|w|, b, POs[b], POs[w])
21:             Errors[b][w] ← Error(w,g)
22:         end for
23:     end for
24:     return new Recipe(W, POs, Errors)
25: end function
```

WfChef

## Find Pattern

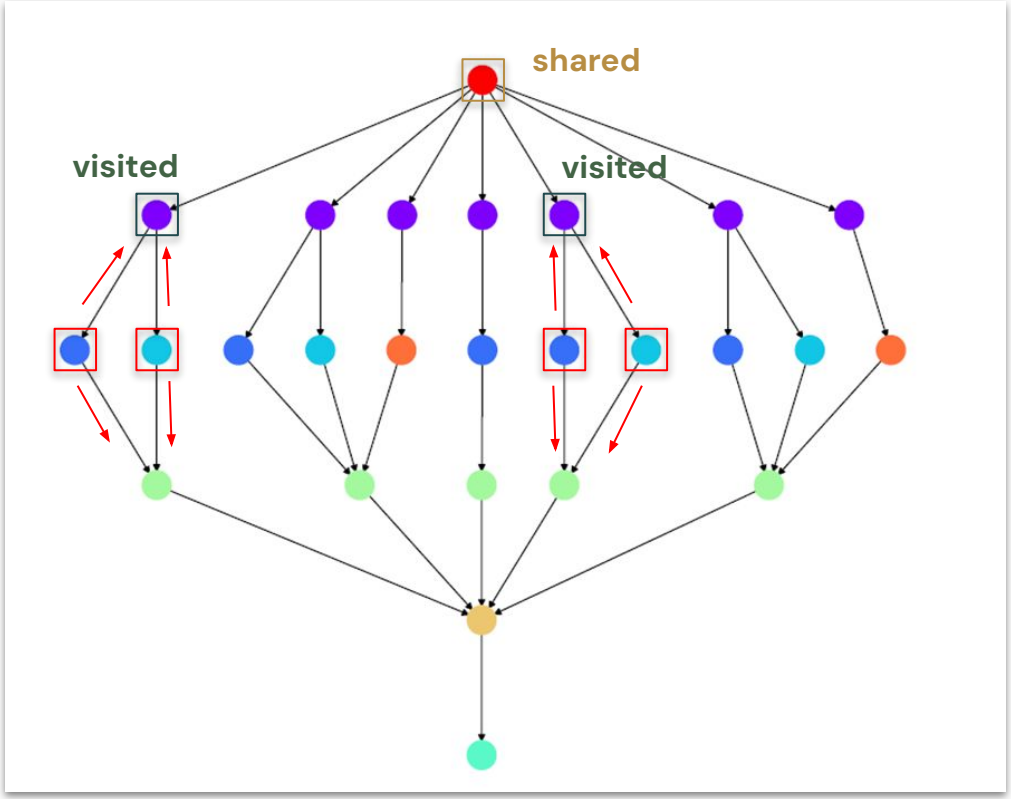
## Round 1



WfChef

## Find Pattern

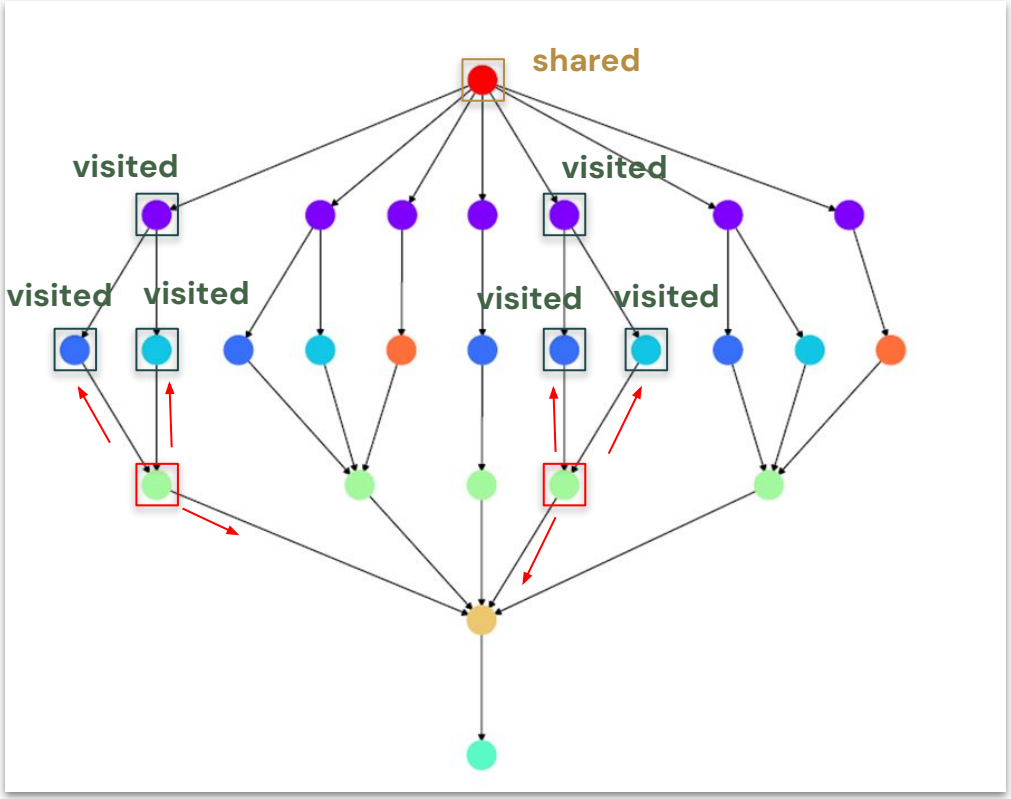
## Round 2



WfChef

## Find Pattern

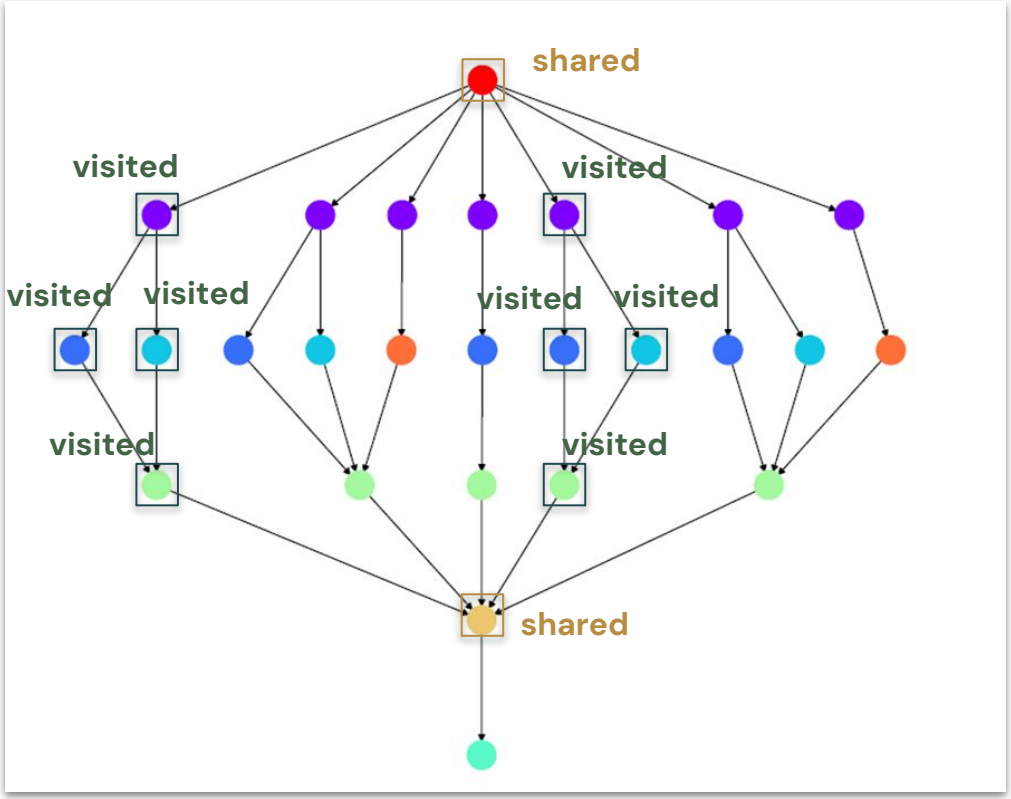
## Round 3



WfChef

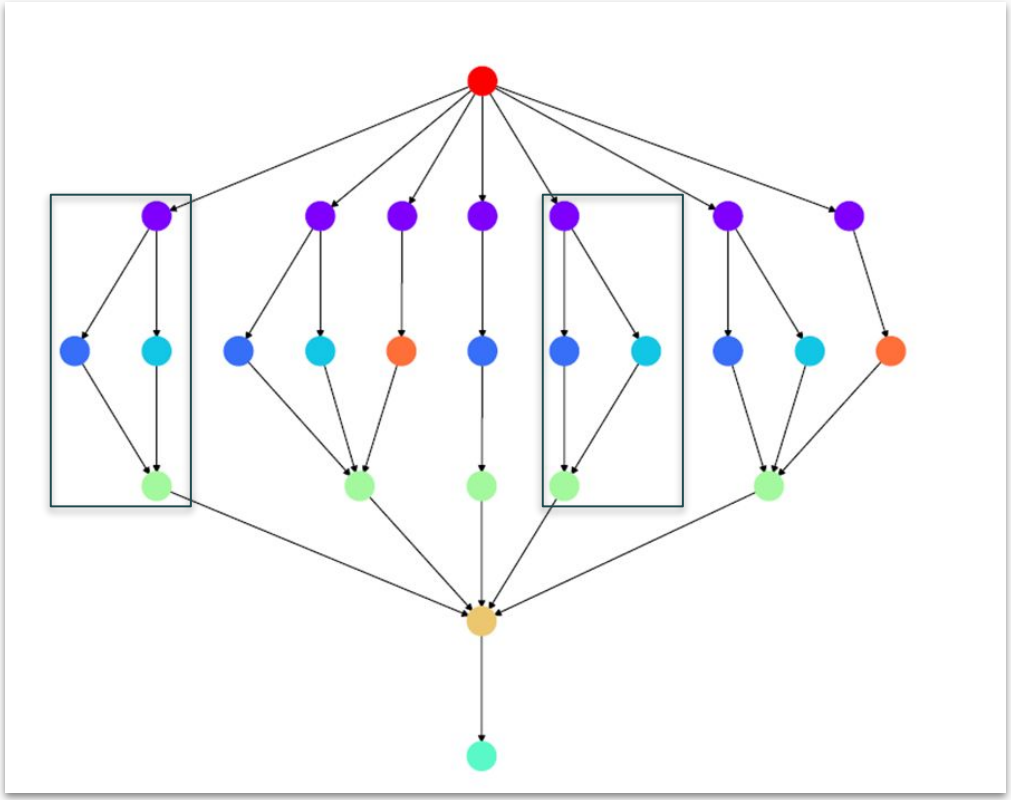
## Find Pattern

## Round 4



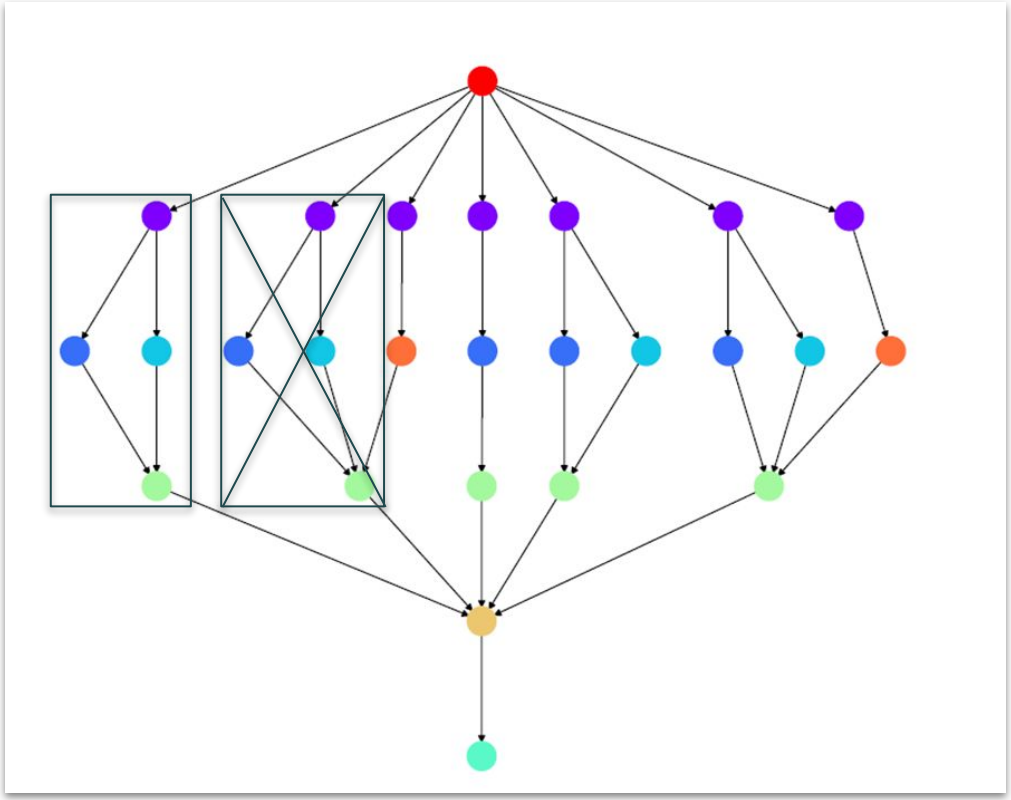
WfChef

## Find Patterns



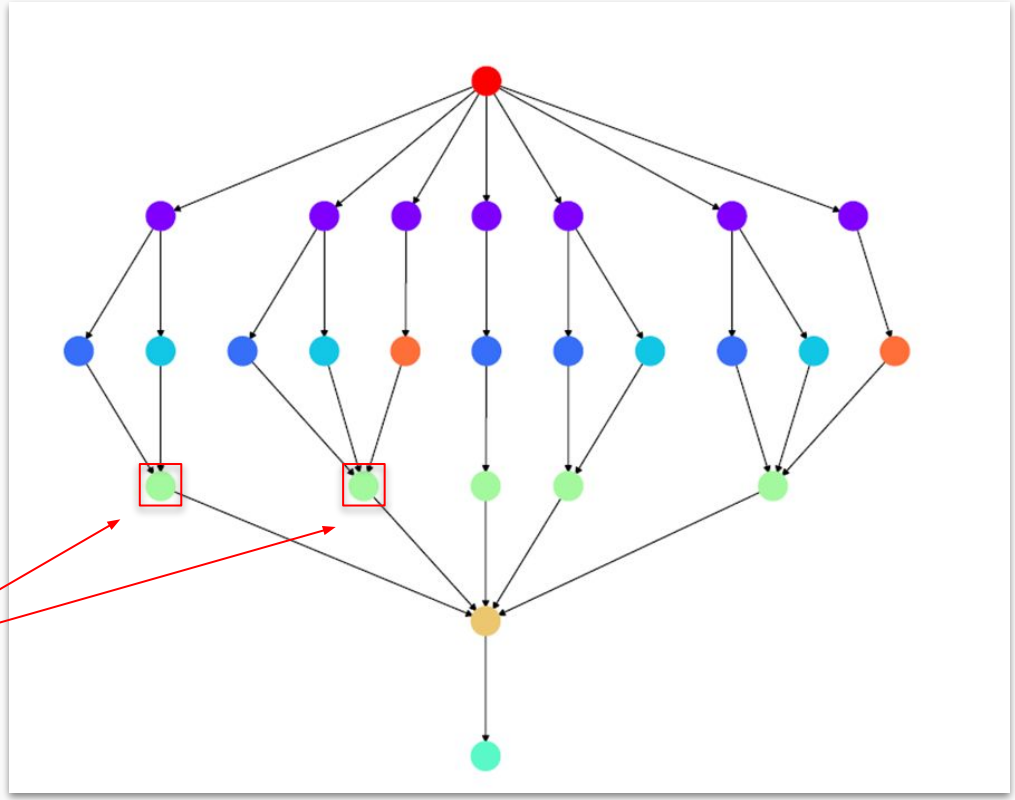
WfChef

## Find Patterns



WfChef

## Find Patterns



Different Type Hashes

WfChef 

## Replicate and WfChefGenerate

**Algorithm 2** Algorithm for generating a synthetic workflow with $n$ vertices based on a recipe.

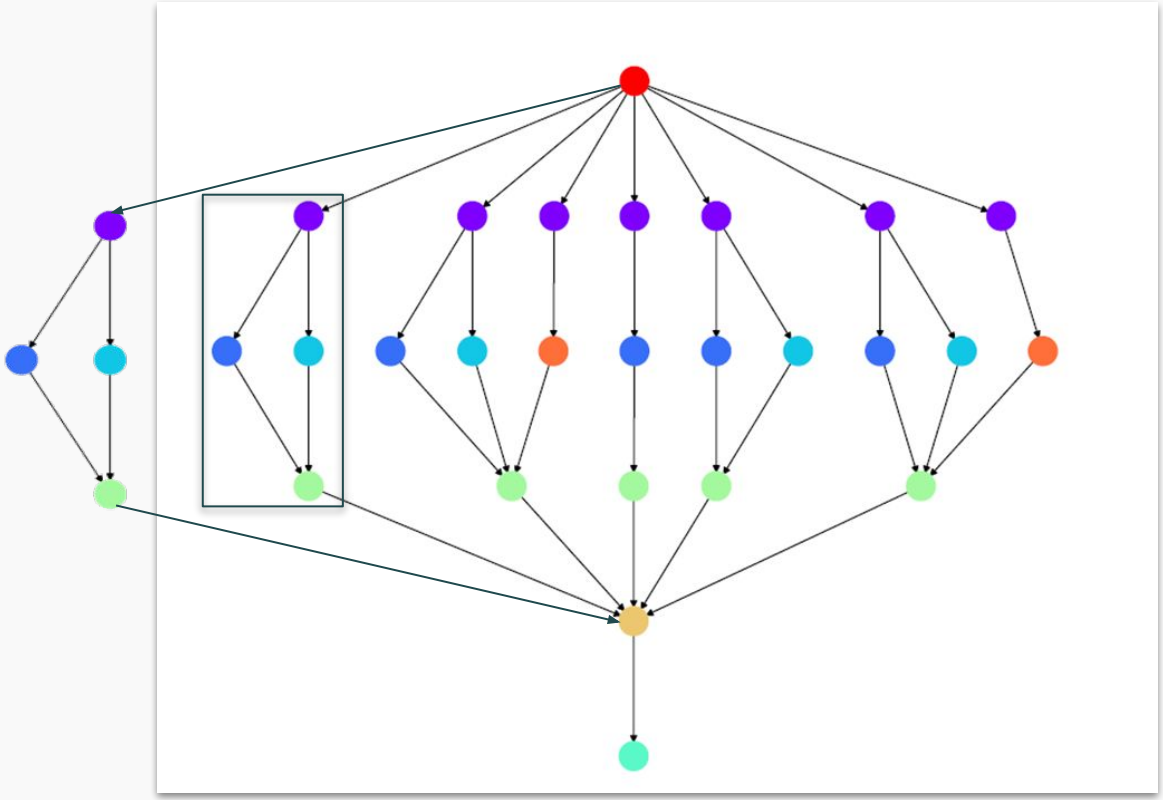
```
1: function WfChefGenerate(rcp, n)
2:     closest ← w in rcp.W s.t. ||w| − n| is minimum
3:     base ← w in rcp.W t. rcp.Errors[w, closest]
           is minimum
4:     g ← ReplicatePOs (n, base, rcp.POs[base],
           r.POs[closest])
5:     return g
6: end function
```

**Algorithm 3** Algorithm for replicating POs in a base workflow.

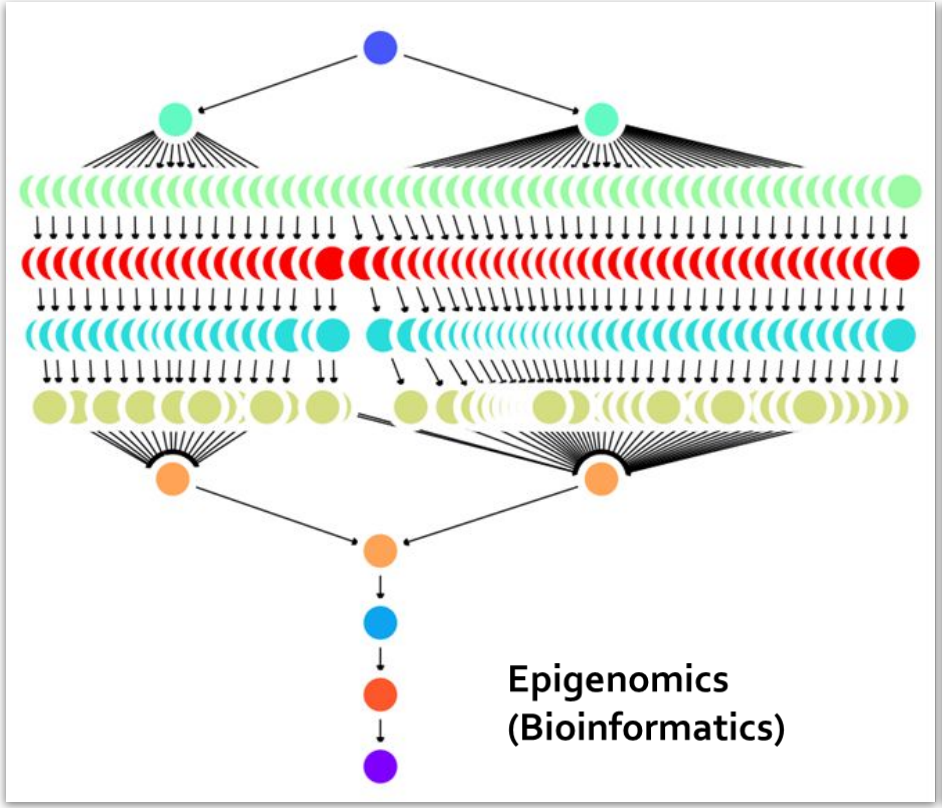
```
1: function ReplicatePOs(n, base, bPOs, cPOs)
2:     g ← base
3:     prob ← {}                    ▷ dictionary of probabilities
4:     for each po ∈ bPOs do
5:         nc = |{p ∈ cPOs | TH(p) = TH(po)}|
6:         tc = |{p ∈ cPOs}|
7:         nb = |{p ∈ bPOs | TH(p) = TH(po)}|
8:         prob[po] ← (nc/tc)/nb
9:     end for
10:    while |g| < n do
11:        po ← sample from bPO with distribution prob
12:        g ← AddPO(g, po)
13:    end while
14:    return g
15: end function
```

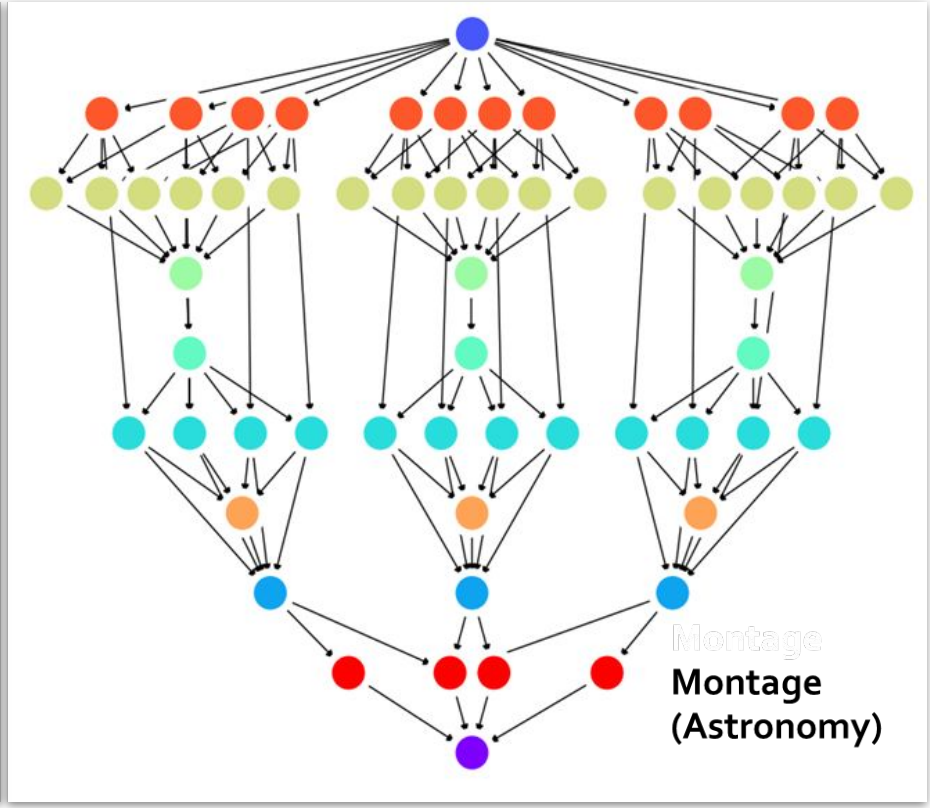
WfChef

## Replicate & Generate



# WfChef Results



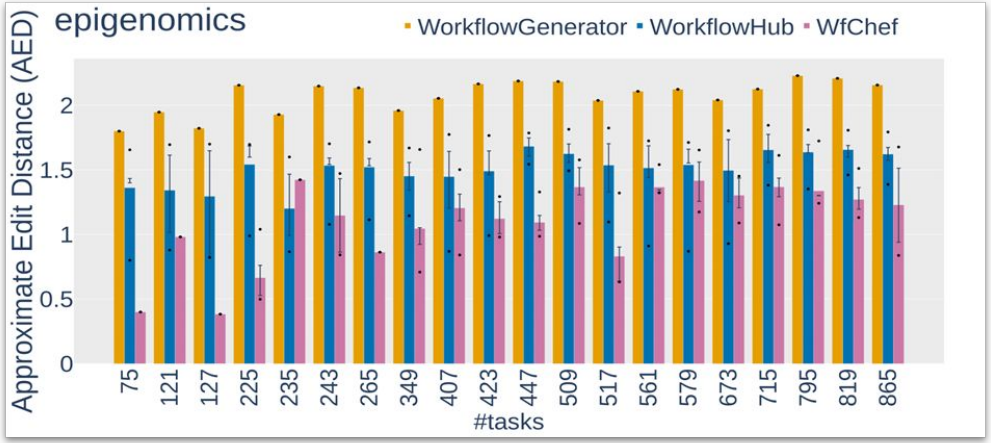



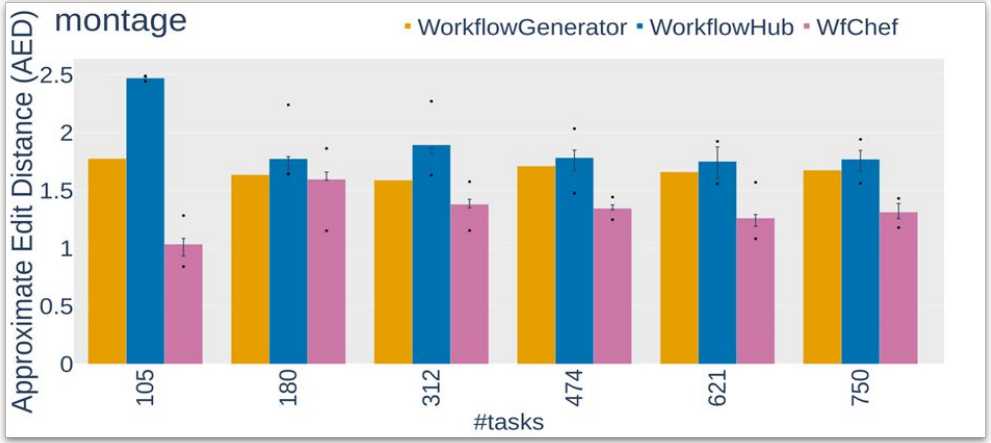
# WfChef Results

## Structure Evaluation

### Approximate Edit Distance

WfChef leads to lower average AED for its instances, which means that they are more faithful to real workflow instances



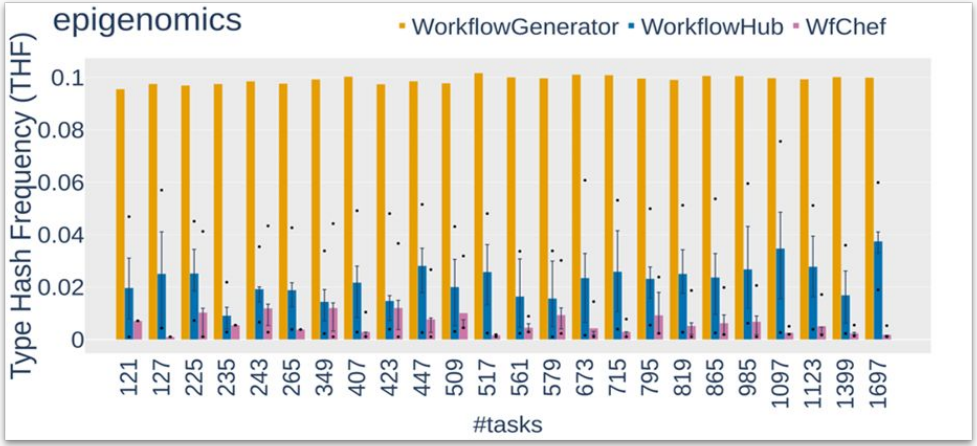
# WfChef Results

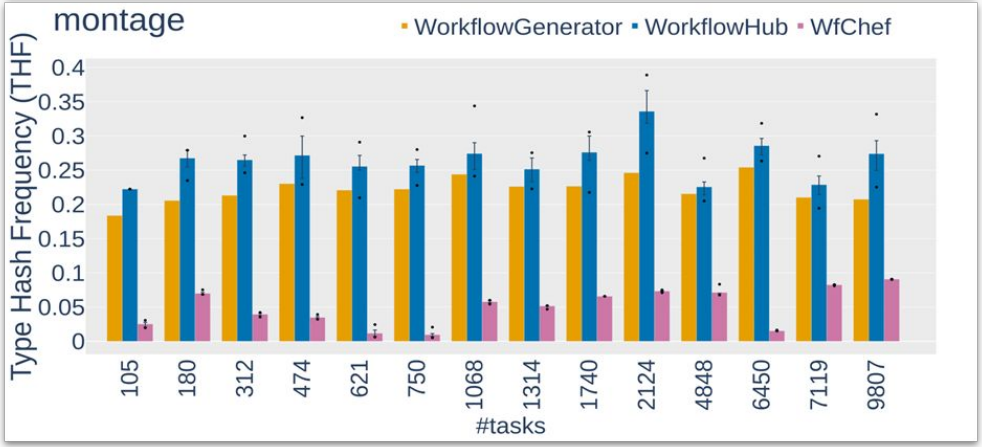
## Structure Evaluation

### Type Hash Frequency

THF metric is computed as the Root Mean Square Error (RMSE) of the frequencies of node type hashes

WfChef outperforms both baselines by wide margins despite having large variances for some workflow scales



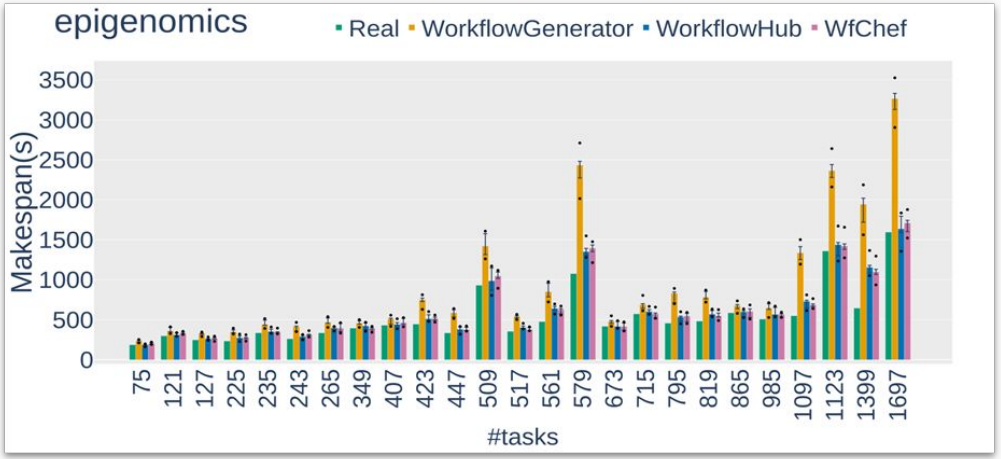
# WfChef Results

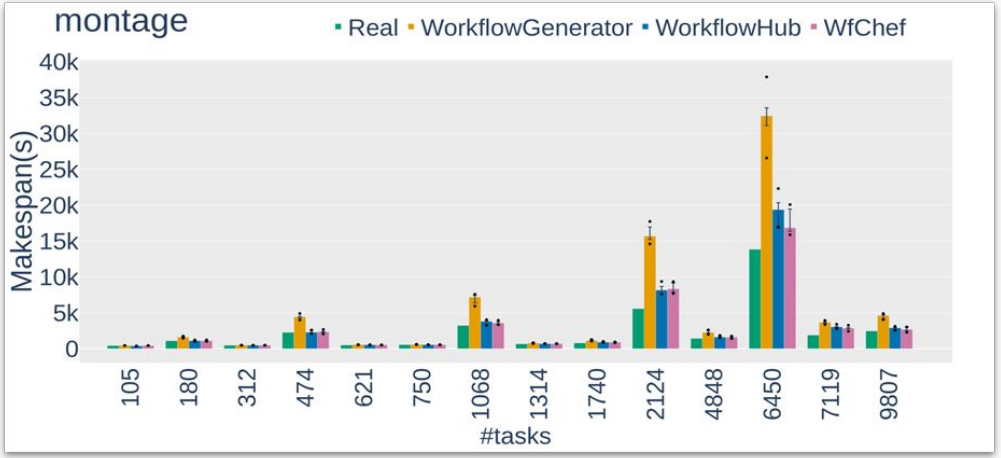
## Performance Evaluation

### Makespan

WfChef produces synthetic workflow instances with realistic makespans

WfChef is very comparable to WorkflowHub but is fully automated



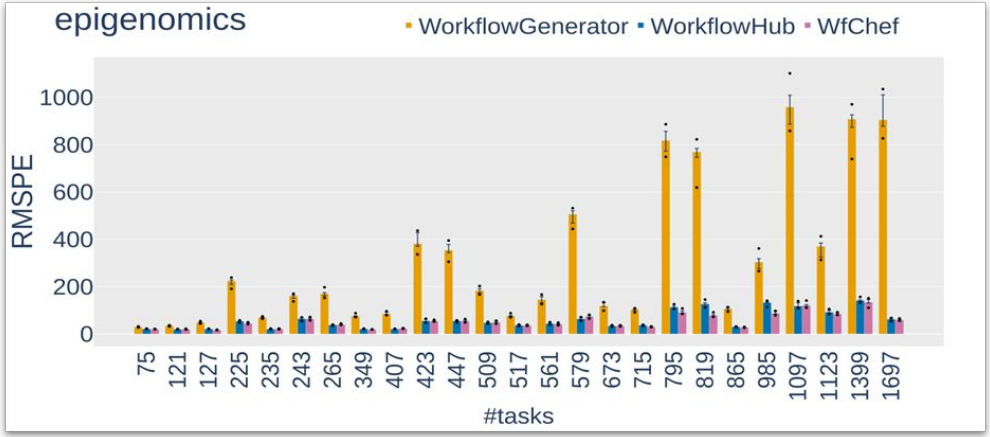



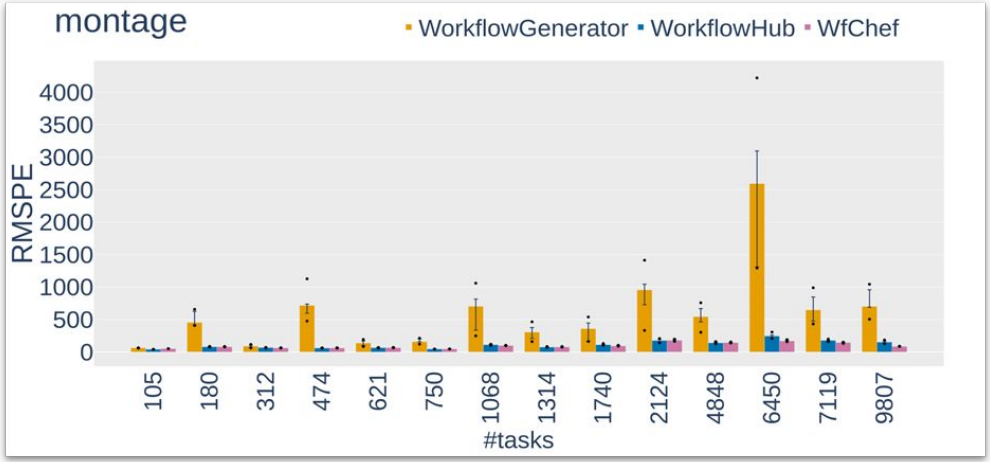
# WfChef Results

## Performance Evaluation

**RMSPE (Root Mean Square Percent Error) of tasks start dates**

WfChef and WorkflowHub are very comparable in this metric also, but WfChef has the advantage of being an automated tool.



# Accuracy vs Dataset Size

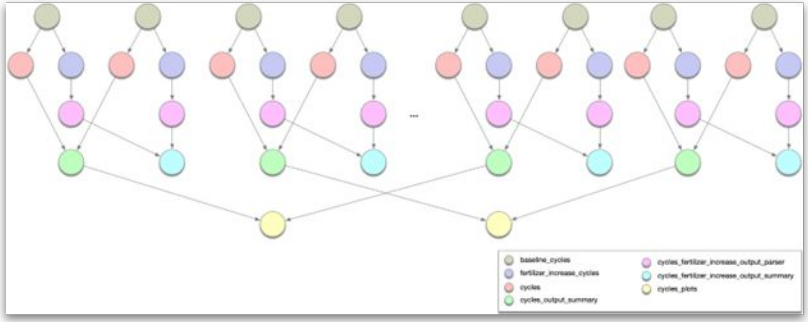
*How many real samples does WfChef need for good results?*

- WfChef does not need all the available real samples to generate accurate synthetic workflow instances.
- Findings:
  - Adding more instances did not yield better results (no new patterns)
  - Increase computational complexity
- Experiment:
  - Remove instances largest to smallest
  - Use AED and THF metrics to measure performance degradation

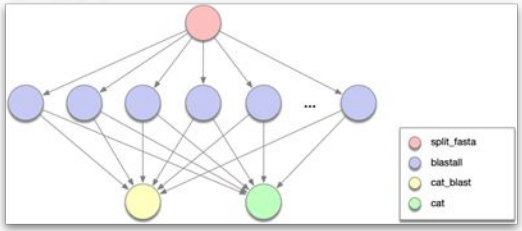
# Accuracy vs Dataset Size

*Applications*

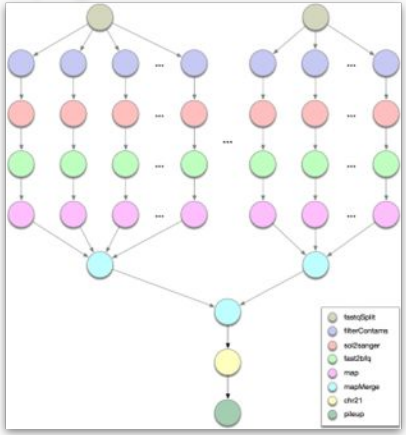
## Cycles



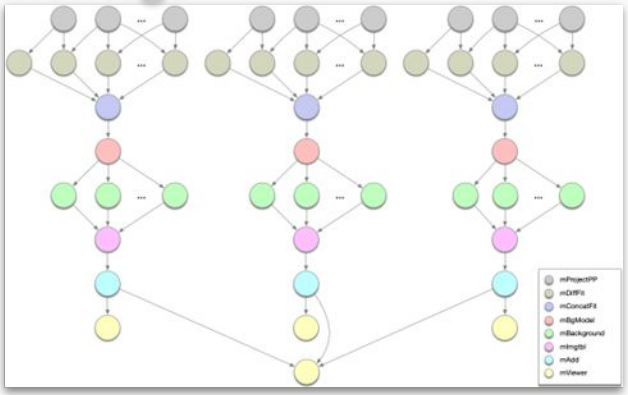

## Blast




## Epigenomics



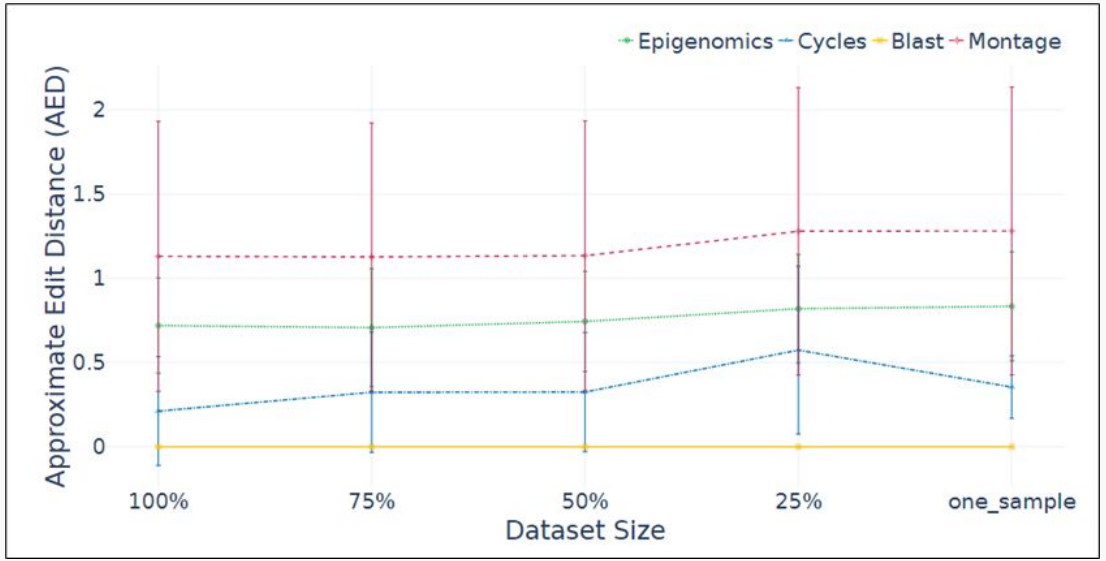

## Montage



# Accuracy vs Dataset Size

*AED values vs. the fraction of real instances used as input.*

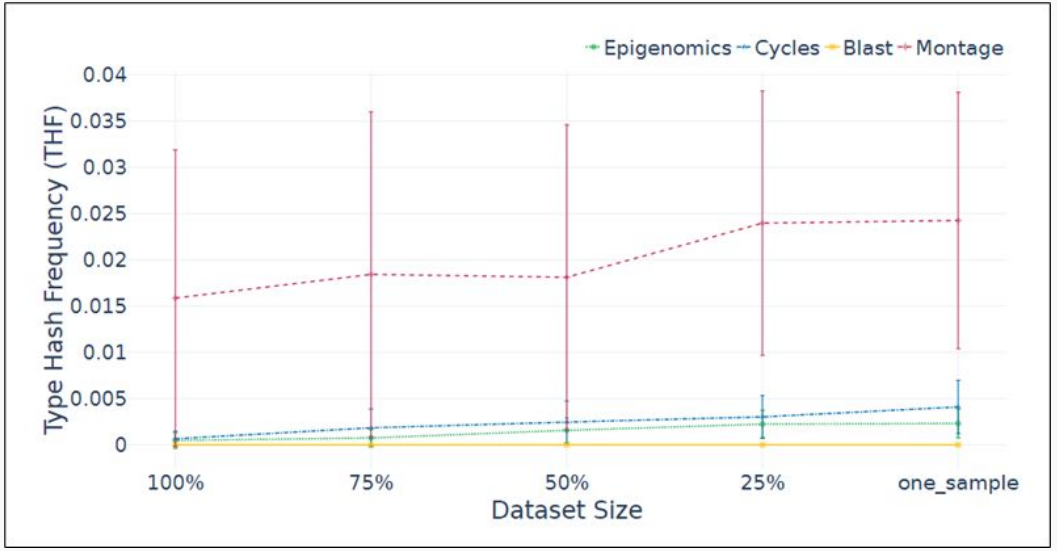
- AED – cost of editing and edge/node (edit cost = 1)
- AED = total_cost / #edges
- Best results:
  - 100% of samples
- Worst results:
  - 25% of samples or single sample
- Difference between min and max:
  - 0 – 0.362



# Accuracy vs Dataset Size

*THF values vs. the fraction of real instances used as input.*

- THF – metric computed as the Root Mean Square Error (RMSE) of the frequencies of node type hashes
- AED = total_cost / #edges
- Best results:
  - 100% of samples
- Worst results:
  - 25% of samples or single sample
- Difference between min and max:
  - 0 – 0.0084



# Accuracy vs Dataset Size

*Are all the real instances needed?*

- Using only the smallest samples (smallest 50%) gets almost as good results as using all of the samples
- The performance degradation is minimal

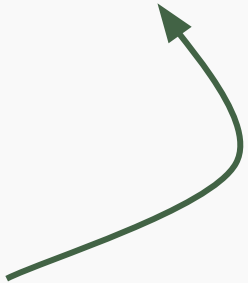
**A small number of real samples as input to WfChef is sufficient to produce realistic synthetic instances**

# What kind of science does this enable?

# Simulation Challenges

*Who came first?*

- Not enough real workflow instances, not enough real workflow executions for analysis
- Solution: execution simulation
- A good simulation model requires a reliable simulator
- To guarantee reliability in simulators: calibration
- To calibrate simulators, real executions are necessary
  - There are not enough real executions
  - To calibrate a simulator, we must use applications that we have control over the parameters and are aware of resulting behaviors



How can we create an Accurate Digital Twin (simulator)?

# Workflow Benchmarks

# Why do we need workflow benchmarks?

*Problem*

- Lack of real or simulated workflow execution
- Diversity of production workflows, execution platforms, and proliferation of workflow systems
- Need to quantify and compare workflow system performance
- Solution: Workflow Benchmarking

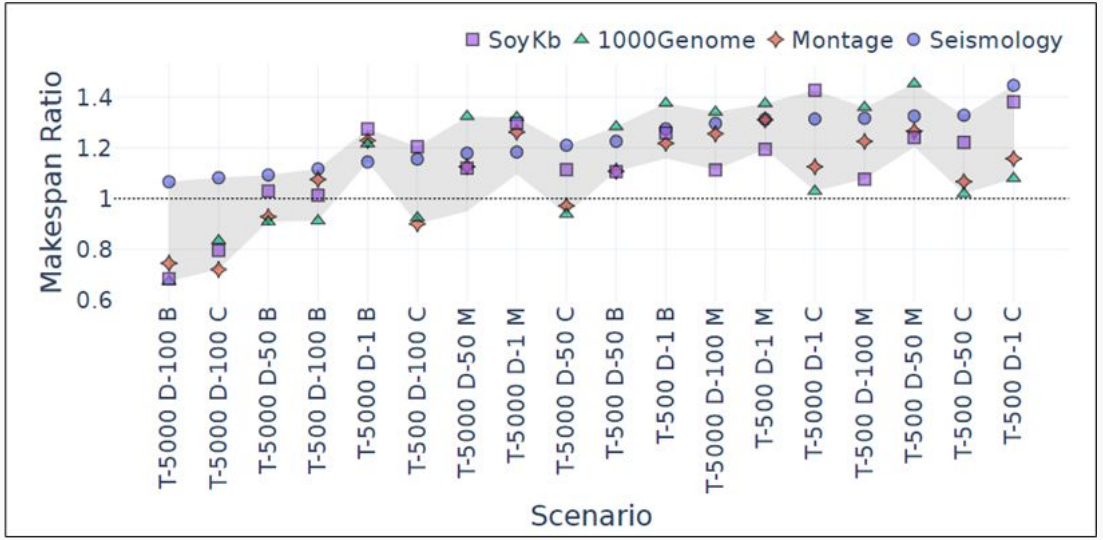
# Extra Motivation

*Result: Ratio between execution times (makespans)*

- Values above y = 1
  - Cascadelake faster
- Values below y = 1
  - Skylake faster

Horizontal axis:

T: number of tasks
D: data footprint in GB
C: cpu-bound
M: memory-bound
B: balanced

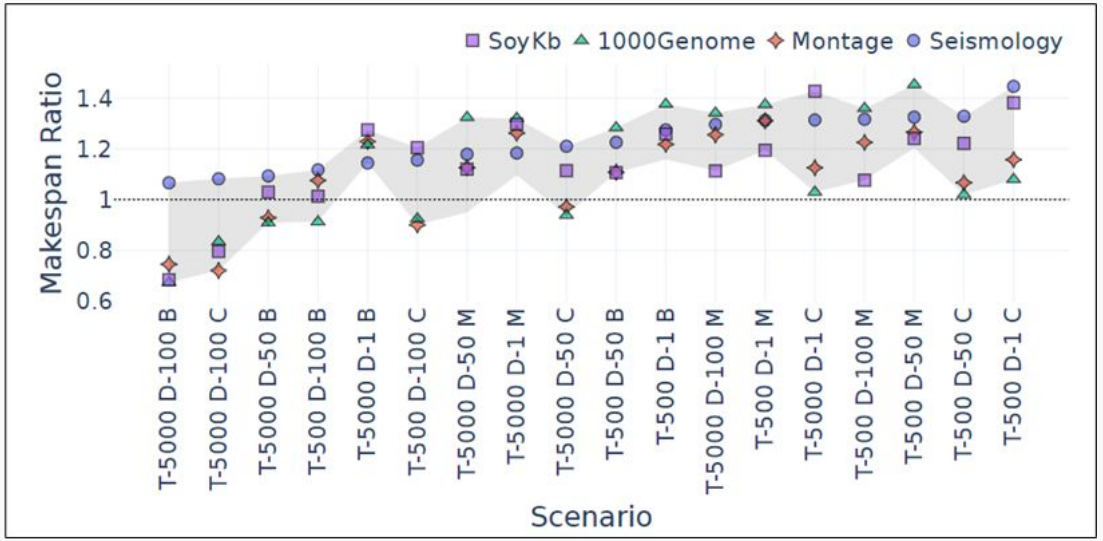



*Sorted by increasing Seismology makespan ratios

# Extra Motivation

*Result: Ratio between execution times (makespans)*

- Results differ drastically
  - Width of the envelope
- Trends are difficult to explain
  - SoyKb vs 1000Genome
- Difficult to explain/predict workflow makespans based solely on platform and workflow configurations



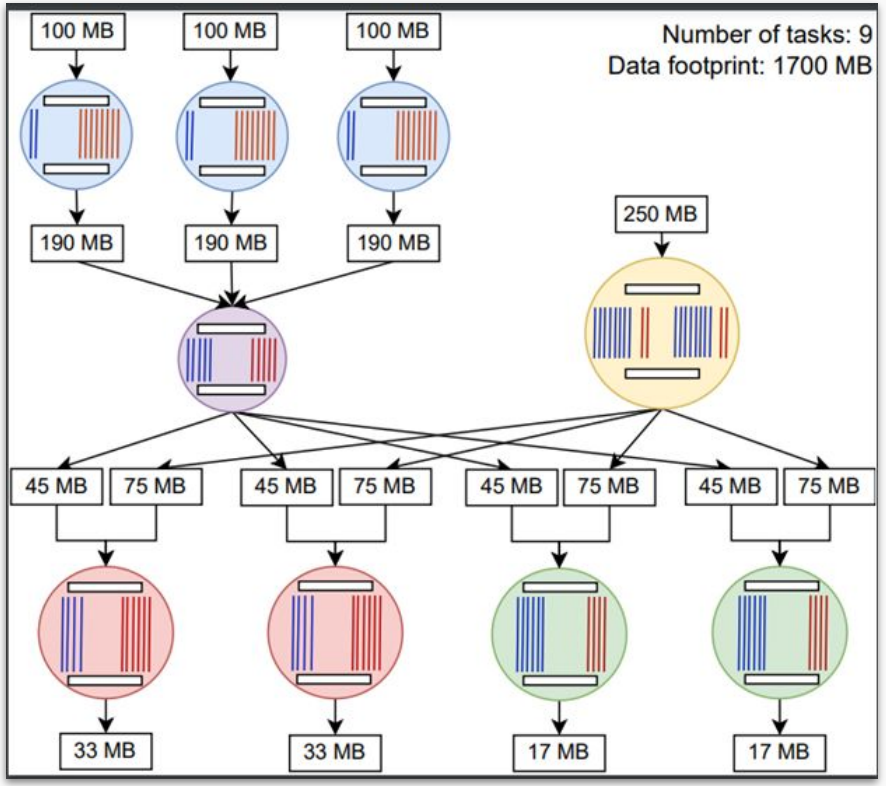

*Sorted by increasing Seismology makespan ratios

# WfBench

*Twofold Approach*

Representative task benchmarks

Representative workflow benchmarks with multiple tasks




WfBench generates realistic workflow tasks with arbitrary I/O, CPU and memory demands, and realistic workflow task graphs based on the real application, agnostic to WMS and independent of platform

# WfBench 

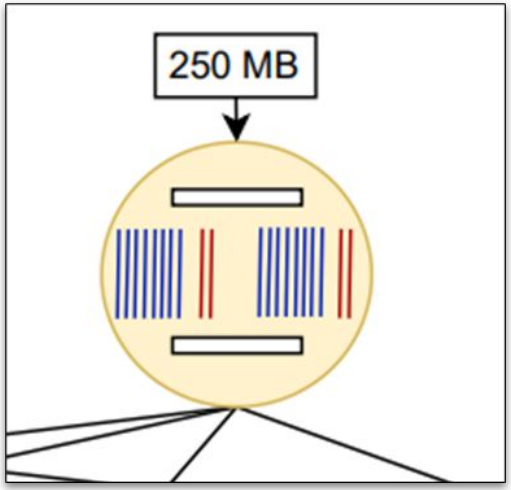
*What is expected of a good Workflow Benchmark Generator?*

- Configurable to represent a range of performance characteristics and structures
- Representative of the real-world workflow application
- Automatically translatable into executable benchmarks of arbitrary workflows systems
- Goal: Measuring the performance of workflow systems not the machines they are running on

# WfBench

*Twofold Approach – Representative Workflow Task Benchmark*

- Read input
- Compute
  - Inputs:
    - cpu work
    - memwork
    - n cores
    - non-mem computation (f)
  - CPU-intensive: Calculates π up to cpuwork
  - Mem-intensive: Access random positions in array adding one unit to it up until memwork
- Write output




**n = 2**

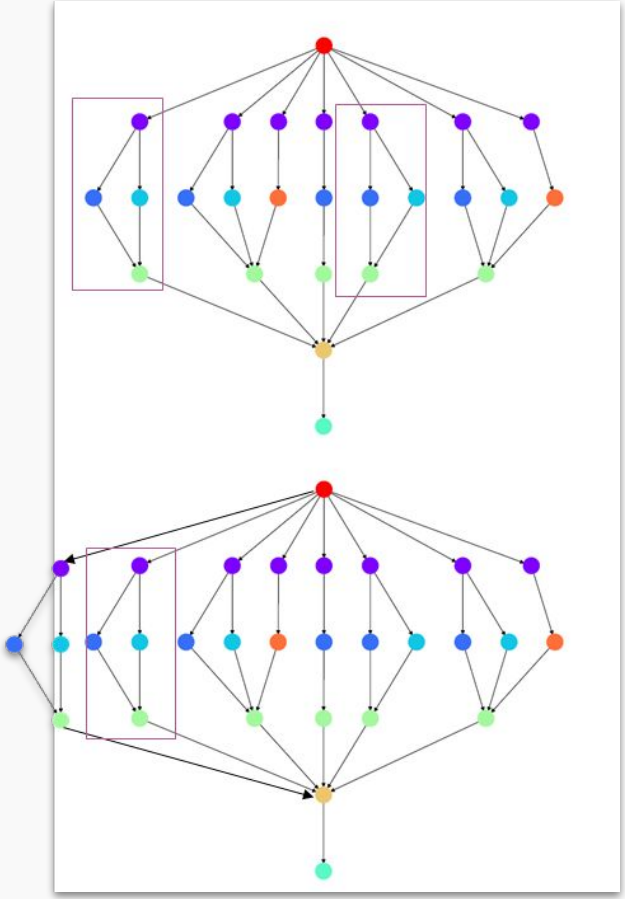
**f = 0.8**

# WfBench

*Twofold Approach – Representative Workflow Benchmark*

WfChef

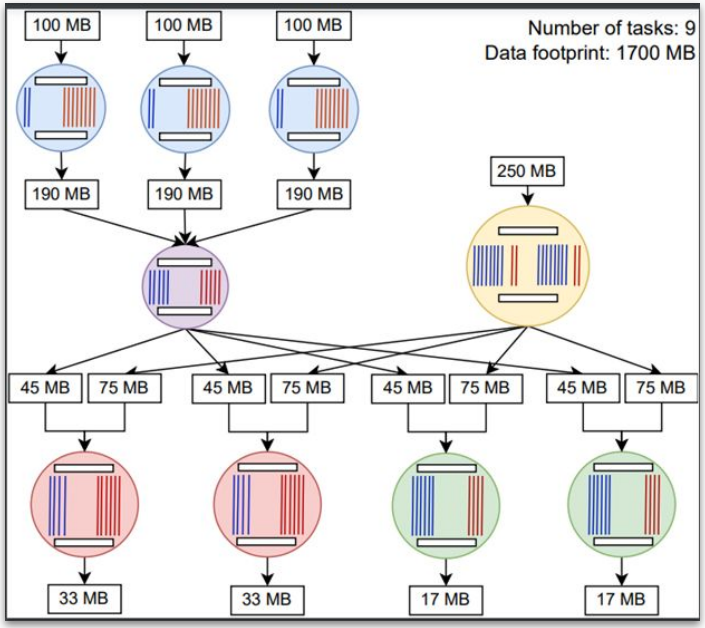
- Automatic generator of realistic synthetic workflow generators
- Inputs
  - Set of real-world workflow instances
  - Desired instance size (number of tasks)
- Analyzes the instances
- Records common patterns
- Creates a recipe
- Replicate patterns



# WfBench

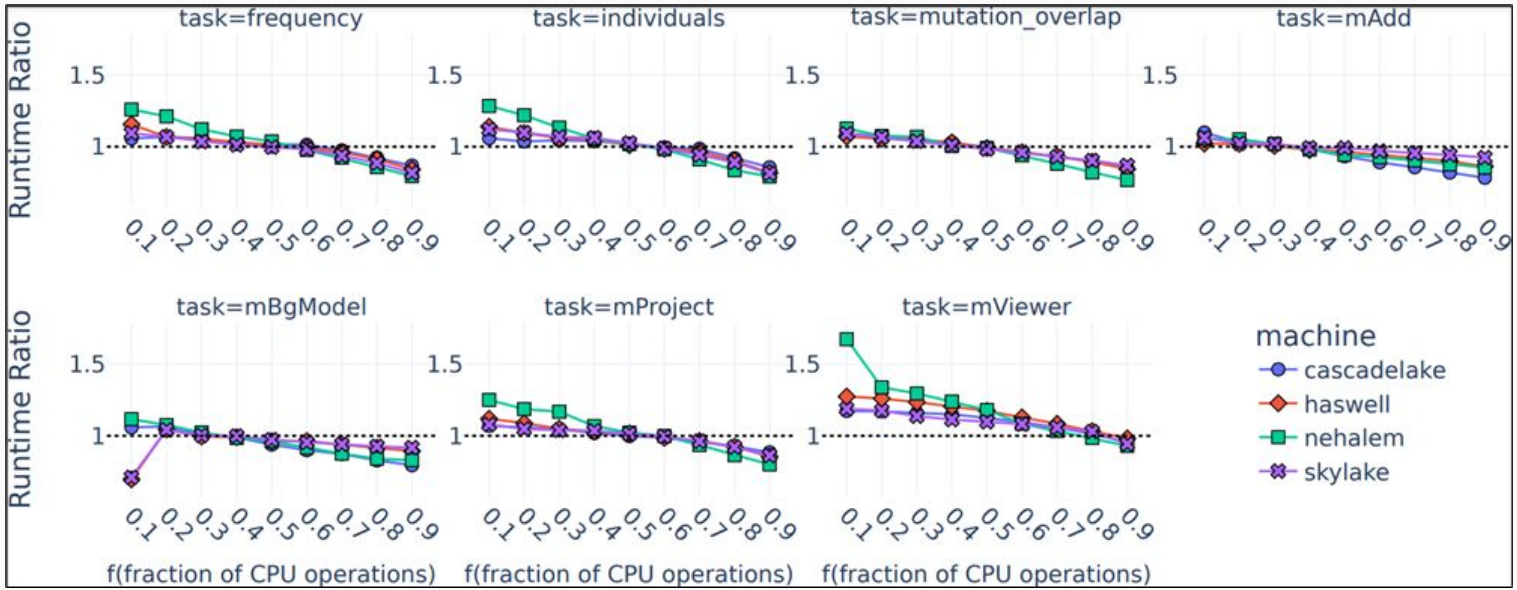
*Twofold Approach*

- Input:
  - Desired # tasks
  - *WfChef* workflow recipe
- Generation:
  - Uses the recipe to generate task graph
- For each task user can specify:
  - (n, cpuwork, memwork, f)
  - Data volume/task or total data footprint
- Output:
  - JSON object that fully describes workflow
  - JSON + Tasks benchmarks = Workflow Benchmark



# WfBench

*Validation of the approach – Task Benchmark*




Is it possible to configure our workflow task benchmark so that its performance behavior is similar to each of these real workflow tasks?

# WfBench

*Validation of the approach – Workflow Benchmark*

TABLE IV
COMPARISON BETWEEN THE REAL-WORLD 1000GENOME WORKFLOW APPLICATION AND THE CALIBRATED WORKFLOW BENCHMARK.

| #ch | #tasks | | makespan (sec) | | makespan |
|---|---|---|---|---|---|
| | 1000Genome | Benchmark | 1000Genome | Benchmark | % difference |
| 1 | 66 | 66 | 971 | 892 | -8.13 % |
| 2 | 232 | 234 | 2548 | 2531 | -0.66 % |
| 3 | 648 | 645 | 6321 | 5645 | -10.56 % |
| 4 | 1548 | 1548 | 13409 | 13553 | 1.07 % |

WfBench makes it possible to generate workflow benchmark that have structure, performance characteristics, and execution patterns, that are very similar to that of real-world workflow.

# What about existing benchmarks?

*Why do they not work?*

- Application benchmarks:
  - Identifying bottlenecks and comparing HPC platforms
  - We are interested in workflow systems performance, such as overhead, and not machine performance
- Existing individual task benchmarks
  - Not customizable enough to be set up to behave as the real workflow tasks
- Real application instances as benchmarks:
  - Limited sample (Not scalable or elastic)

# What about existing benchmarks?

*Why do they not work?*

- Current benchmark suites capture some but not all features of a workflow application:
  - Workflows: collection of different “types” of tasks (I/O, GPU, CPU, memory consumption)
  - Tasks of same type can have a different resource consumption
  - Concurrent task execution: performance interference
  - WMSs can configure workflow execution is various ways (task scheduling decisions)

Workflow benchmarks need to execute seamlessly in a wide range of WMSs rather than being implemented for a specific one.

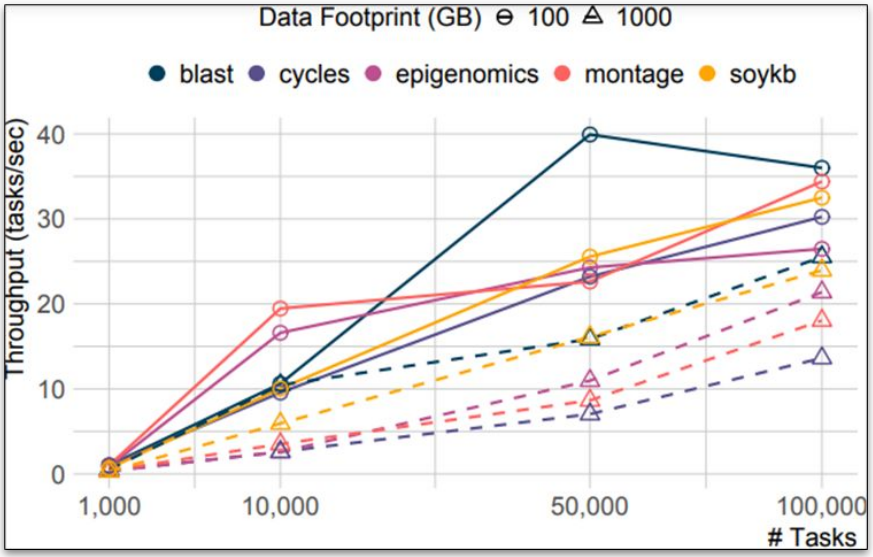
# What about existing benchmarks?

*Experimental Evaluation*

Workflow throughput (#tasks/sec)

↑ data footprint ↓ throughput

↑ #tasks ↑ throughput



Use Cases

# Simulation Energy Consumption

- Used WBench instances to calibrate a simulator and collect power consumption data
- Analytical study of accuracy of traditional energy-consumption model (Linear Model)
- Result: underestimate energy-consumption by up to 360%
- Solution: Proposed new model that includes I/O operations and idle time

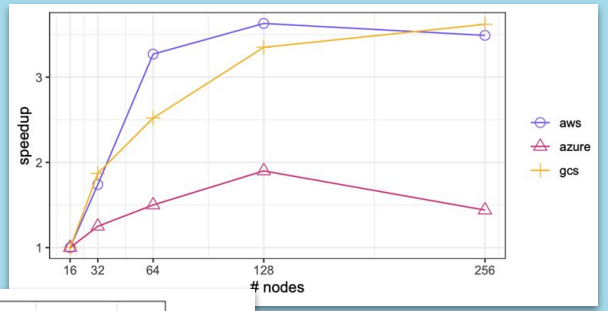
| Workflow | Algorithm | Energy Error (KWh) Max | Mean | Stand. Deviation |
|---|---|---|---|---|
| Montage | SPSS-EB | 0.45 (160.71%) | 0.09 (39.31%) | 0.17 (52.07%) |
| | EnReal | 0.22 (17.01%) | 0.06 (7.81%) | 0.01 (1.89%) |
| SoyKB | SPSS-EB | 0.66 (33.76%) | 0.13 (5.37%) | 0.37 (15.72%) |
| | EnReal | 0.12 (10.46%) | 0.11 (9.17%) | 0.02 (1.65%) |
| 1000Genome | SPSS-EB | 5.88 (360.57%) | 1.95 (187.46%) | 1.80 (120.50%) |
| | EnReal | 0.35 (26.29%) | 0.11 (14.65%) | 0.10 (7.08%) |

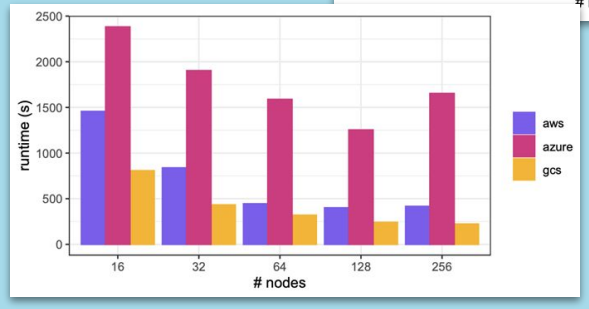
**Table 1.** Energy consumption error (maximum, mean, and standard deviation) for both algorithms and for each set of workflow instances.

# Evaluation of Cloud Capability



- Does an HPC cloud system have the same or better capability of running HPC workloads than supercomputers?
- WfBench is the official benchmark of OLCF-6 procurement process



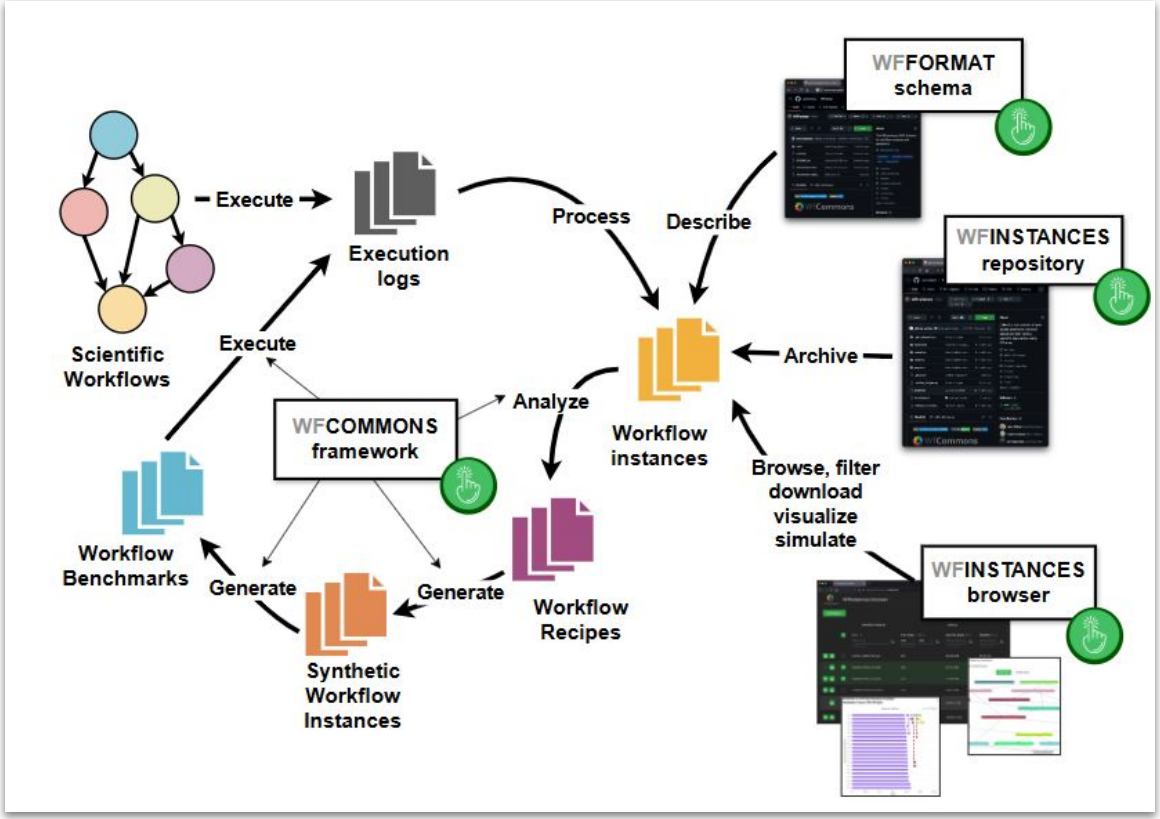



# WfCommons

# WfCommons

*wfcommons.org*

**WfCommons** is a framework that provides a collection of tools for analyzing workflow execution traces, producing realistic synthetic workflow traces, and benchmarking / simulating workflow executions.



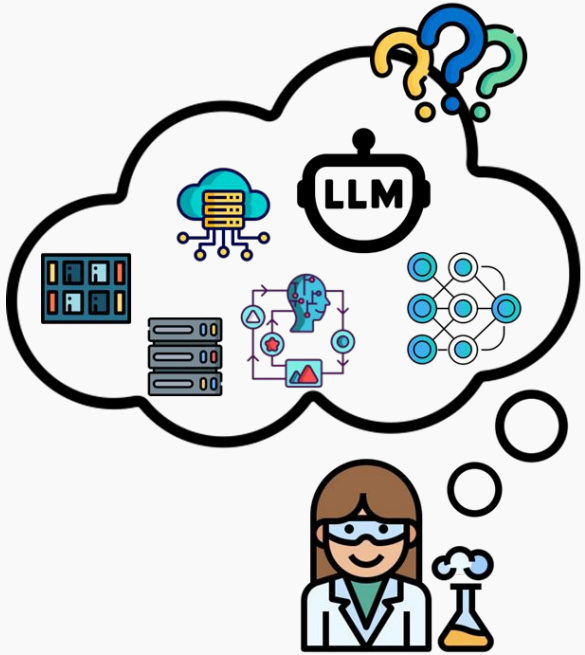
# They use WfCommons

- University of Southern California
- Oak Ridge National Laboratory
- Academic Computer Centre Cyfronet AGH (Poland)
- The University of Western Australia
- Imperial College London
- Beijing Institute of Technology
- Universidad Politécnica de Cartagena
- Iran University of Science and Technology
- University of California, Irvine
- University of Innsbruck (Austria)
- HPE
- NVidia

# AI Workflows for AI in Various Domains

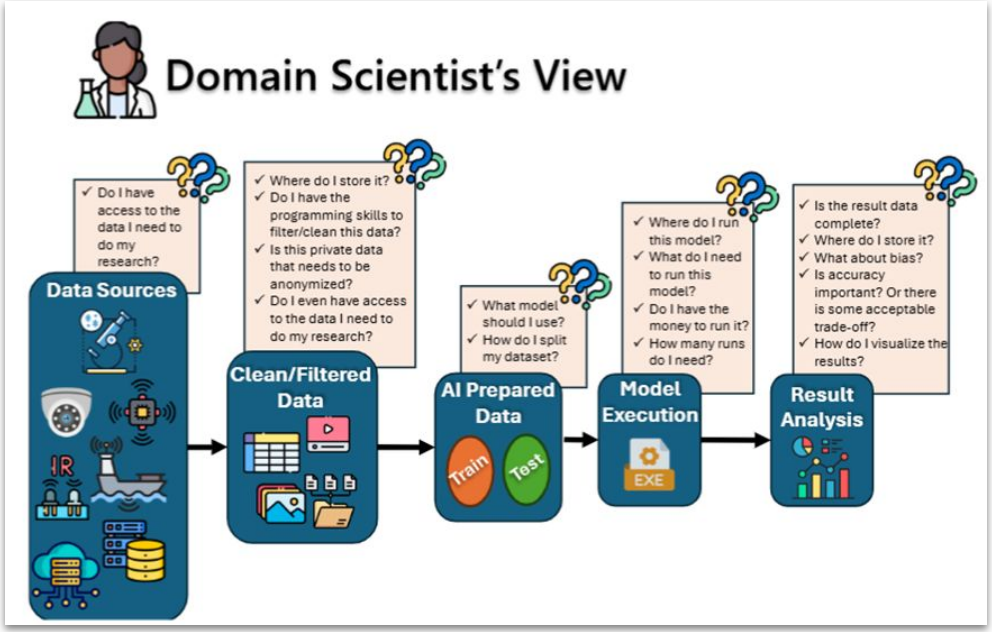
# AI is accelerating discovery but ...

- Many domain scientists face significant barriers incorporating AI in their research.
- The variety of platforms, models, and processes generates unrealistic expectations.
- Good choices are a combination of formal training and tacit expertise.
- Tacit expertise comes from years of experience and lots of trial and error



# AI expertise gap

- Key challenges:
  - Lack of AI/ML formal training
  - Computational resource constraints
  - Data prep & metadata issues
  - Workflow fragmentation



# Insights from domain scientists



- We interviewed 11 Schmidt AI postdoctoral researchers at UCSD
- Barriers are consistent across all domains

*" Interviewee 1 struggled with basic infrastructure tasks, such as uploading data to a supercomputer, while Interviewee 7 found the training and implementation process intimidating without curated examples or documentation."*

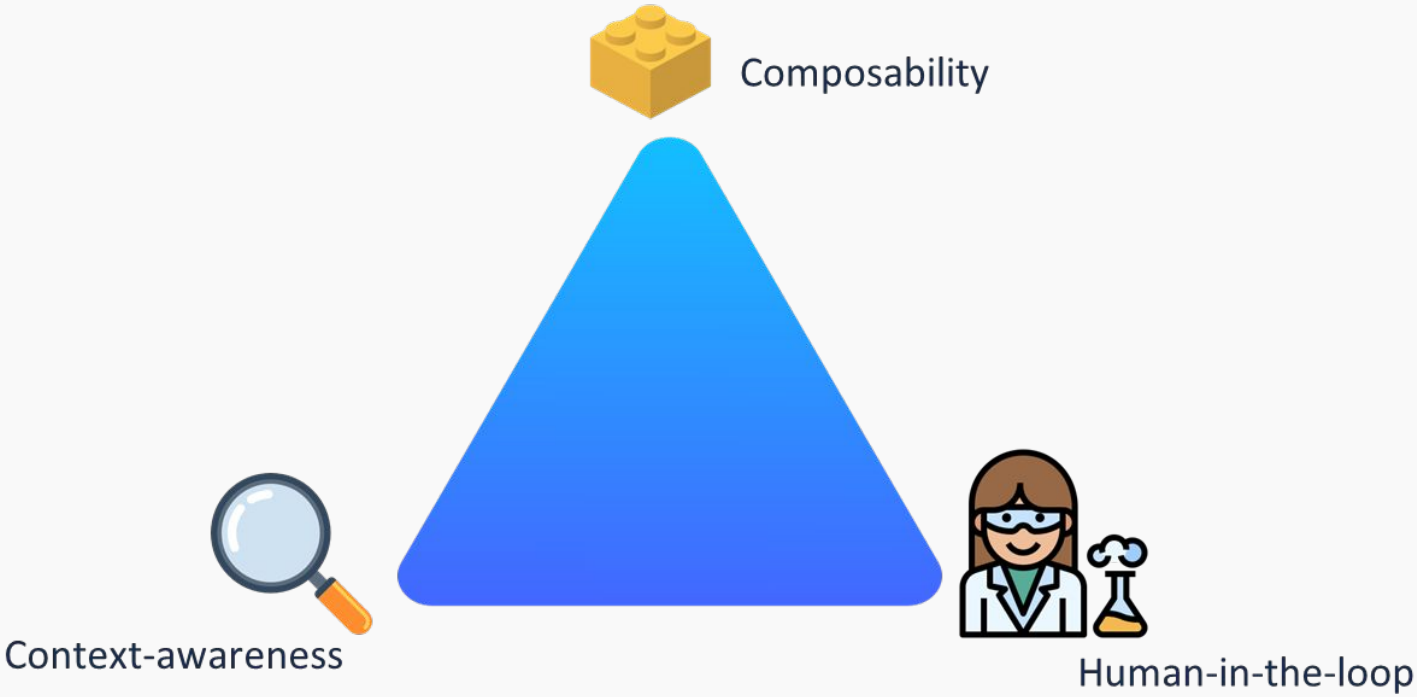
TABLE I: Anonymized Interviewee Disciplines

| Interviewee ID | Scientific Domain / Focus |
|---|---|
| Interviewee 1 | Medical Imaging / Cardiac CT |
| Interviewee 2 | Behavioral Neuroscience / Sleep Tracking |
| Interviewee 3 | Behavioral Neuroscience (Visualization Focus) |
| Interviewee 4 | Molecular Modeling / Protein Binding |
| Interviewee 5 | Ecology / Root Systems and Microscopy |
| Interviewee 6 | Structural Biology and Genomics / HIV Evolution |
| Interviewee 7 | Earth Systems Modeling |
| Interviewee 8 | Autism and Protein Oscillation / Transfer Learning |
| Interviewee 9 | Quantum Chemistry / Neural Potentials |
| Interviewee 10 | Evolutionary Genomics / ML Benchmarking |
| Interviewee 11 | Molecular Dynamics / COVID-19 Protein Systems |

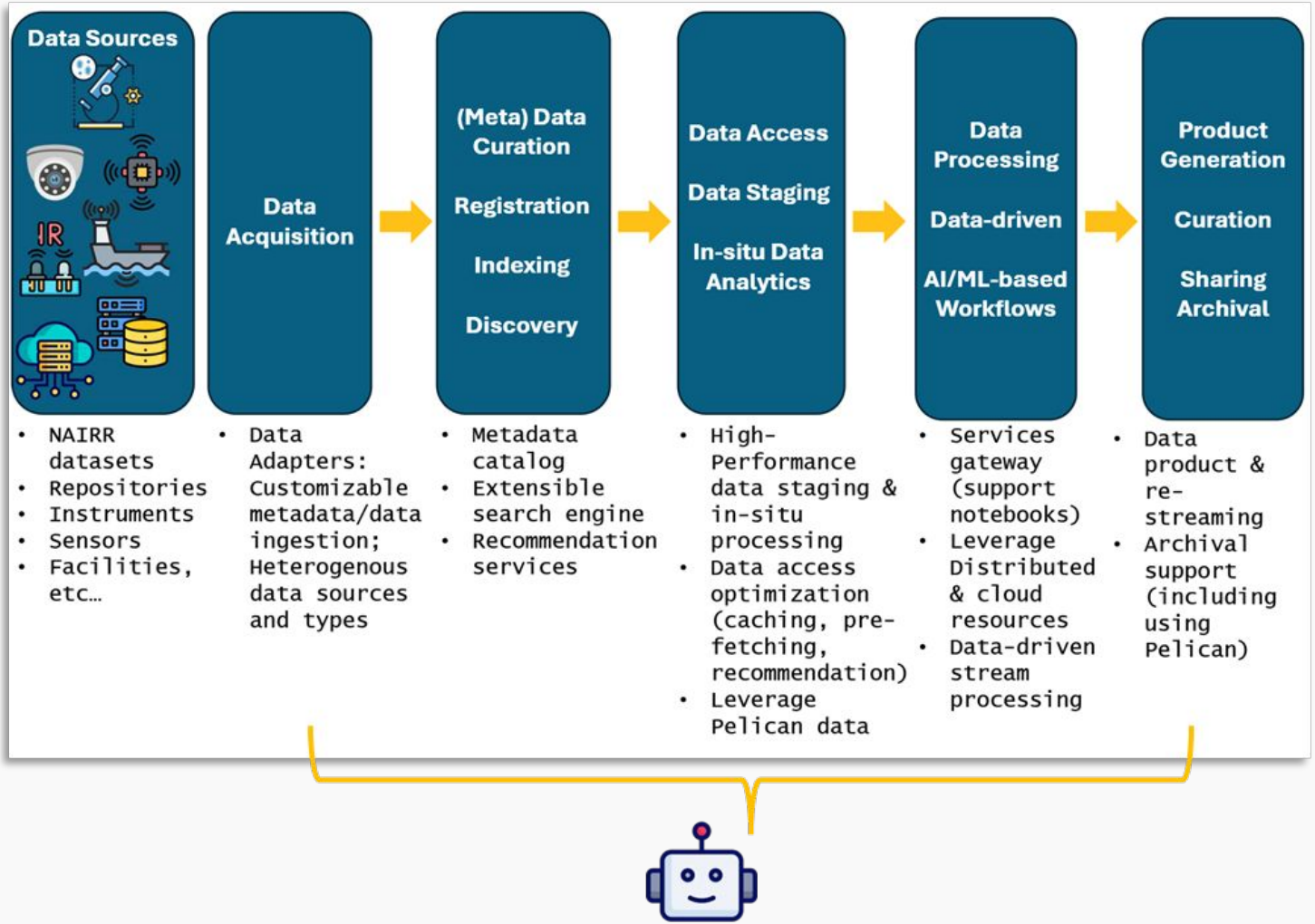
# What are Agents of Intelligence?

- Modular, context-aware, human-in-the-loop companions
- We focus on 4 capabilities:
  - Sensing: awareness of the current workflow state
  - Automating: help reduce cognitive load
  - Recommending: able to suggest appropriate models, resources, visualizations, or workflow steps based on best practices
  - Learning: improving over time based on user feedback and usage patterns across domains
- **Our goal:** encode scientific intuition into intelligent agents that support decision-making across the research lifecycle.

# Design Principles

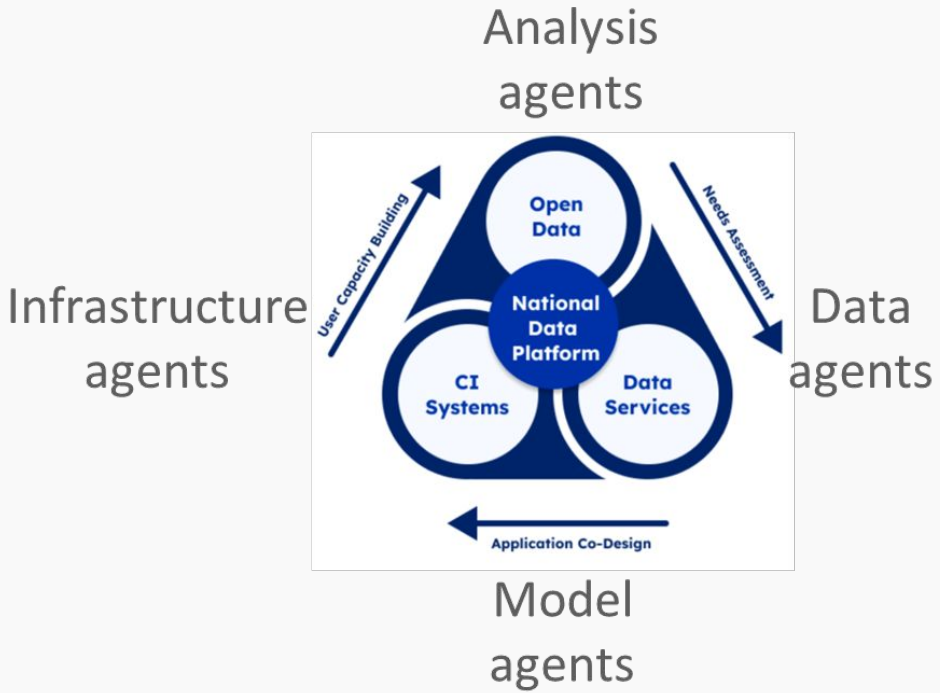


# Across the data flow



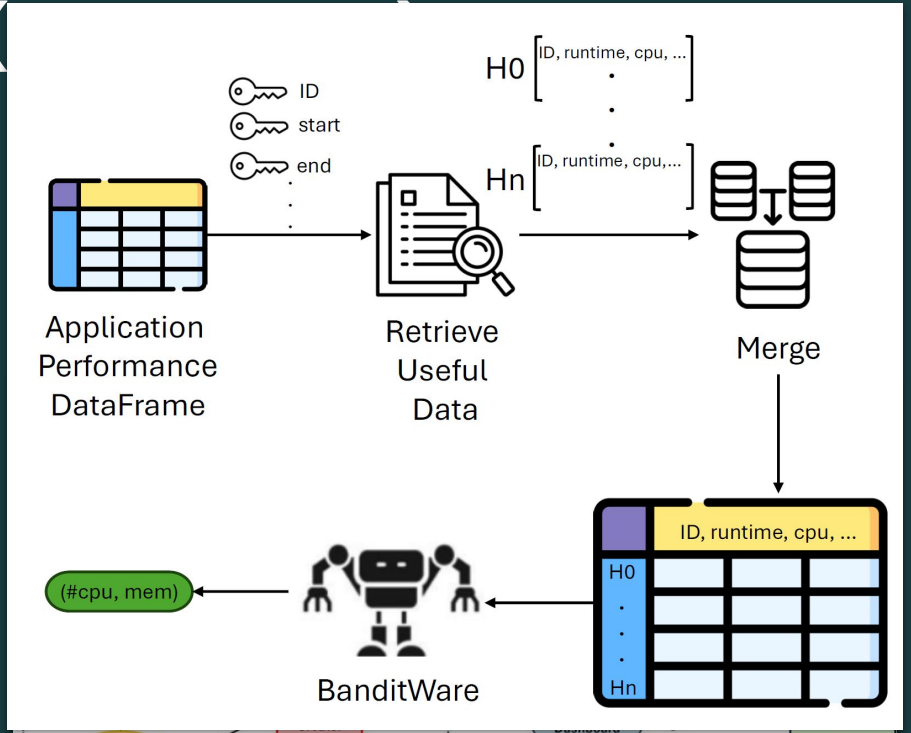
# Integration with NDP

- **Why the National Data Platform?**
  - Federated metadata-rich, workflow-awareness, domain-agnostic
  - Deployment Modalities: Embedded, Orchestrated and On-Demand
  - Existing and extensible policy framework



# Use Cases

## Wildland Fire Science (




# BanditWare



- An online recommendation system for hardware selection, built on contextual multi-armed bandits.
- Key idea: Learns hardware workflow fit in real time without needing large historical datasets.
- How it works:
  - Collect workflow features (e.g., size, tasks).
  - Predict runtimes across hardware options.
  - Balance exploration (try new hardware) and exploitation (pick best-so-far).
  - Update model continuously.

# Broader Implications

The framework is not domain-specific, the same categories of agents (Data, Model, Infrastructure, Analysis) can support many domains.

- **Life Sciences:** Assist with genomics pipelines, model selection for protein structures, metadata curation
- **Neuroscience & Behavioral Sciences:** Data labeling, time-series model recommendations, visualization of neural activity
- **Physics & Chemistry:** Simulation setup, parameter tuning, resource optimization (HPC/cloud)
- **Environmental & Earth Systems:** Large-scale data ingestion, cross-domain learning, uncertainty-aware analysis
- **Humanities and Language:** Transfer learning for low resource language revitalization, data unlearning for policy enforcement

# Beyond the Sciences

# Humanities Research and AI

- Interest around AI for the humanities is exploding
  - Semantic search in historical archives
  - Historical artifacts reconstruction
  - Manuscript decipherment
  - Authorship attribution
- How to support humanities researchers' computational needs?
  - The variety of tools available and the expertise gap can be overwhelming for these researchers

Use Cases

# AI for Endangered Language Revitalization

- Humanities applications often have a cultural aspect to them
- Faculty from LMU and USC are actively working on using Generative AI to support language revitalization efforts
- Large barrier: lack of infrastructure for hosting/accessing data, models, etc. that take cultural protocols into consideration
    - Some stories from Dr. Coleman's tribe can only be told in the winter time

# Research Questions



- Can we successfully use existing transfer learning models and federated data to represent the grammar and vocabulary of an extremely low-resource language?
- What are best practices for creating digital corpora of endangered languages while respecting cultural protocols?
    - Data unlearning
    - Policy implementation
- How can AI-based tools (e.g., speech recognition, chatbots) support endangered language learning?
- How can the success of AI-supported language revitalization projects be quantified?

**And many more …**

# Bridging expertise gaps makes research more inclusive & effective

Schmidt Sciences

U.S. National Science Foundation

Societal Computing and Innovation Lab

SDSC

workflows community initiative

NATIONAL DATA PLATFORM

UNIVERSITY OF SOUTHERN CALIFORNIA 1880

UNIVERSITY of HAWAI'I MĀNOA

OAK RIDGE National Laboratory

WfCommons

# Thank you!

NATIONAL DATA PLATFORM