Living in a Heterogenous World: How scientific workflows help automate science and what we can do better?

Ewa Deelman
University of Southern California, School of Engineering
Information Sciences Institute
deelman@isi.edu

This work is funded by NSF awards #2018074 and #1664162.

We started looking at supporting users and their applications in 2000

Virgo visit
September 2001
Pisa, Italy

- https://www.ego-gw.it/public/about/welcome.aspx

Pegasus Workflow Management System

Workflow Challenges Across Domains

Describe complex workflows in a simple way

Access distributed, heterogeneous data and resources (heterogeneous interfaces)

Deals with resources/software that change over time

Ease of use. Ability to monitor and debug large workflows

Our Focus

- Separation between workflow description and workflow execution
- Workflow planning and scheduling (scalability, performance)
- Task execution (monitoring, fault tolerance, debugging, web dashboard)
- Workflow optimization, restructuring for performance and fault tolerance.

USC Viterbi School of Engineering Information Sciences Institute

Pegasus Workflow Management System est. 2001

Automates the execution of scientific workflows across national CI

Heterogeneous Environments

Pegasus can execute workflows in a variety of distributed computing environments such as HPC clusters, Amazon EC2, Google Cloud, Open Science Grid or ACCESS

Data Management

Pegasus handles data transfers, input data selection and output registration by adding them as auxiliary jobs to the workflow

Provenance Tracking

Pegasus allows users to trace the history of a workflow and its outputs, including information about data sources and software used

Error Recovery

Pegasus handles errors by retrying tasks, workflow-level checkpointing, re-mapping and alternative data sources for data staging

Collaboration with HTCondor

USC Viterbi School of Engineering Information Sciences Institute

This work is funded by NSF, award # 1664162

1. Resource-independent Specification

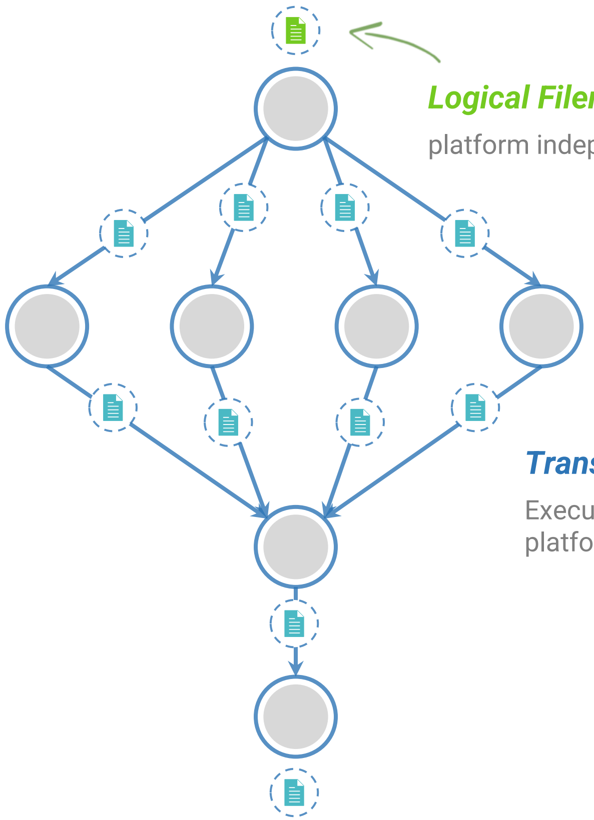
Input Workflow Specification YAML formatted

directed-acyclic graphs Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW



Logical Filename (LFN)

platform independent (abstraction)

Transformation

Executables (or programs)
platform independent

Stage-in Job

Transfers the workflow input data

Cleanup Job

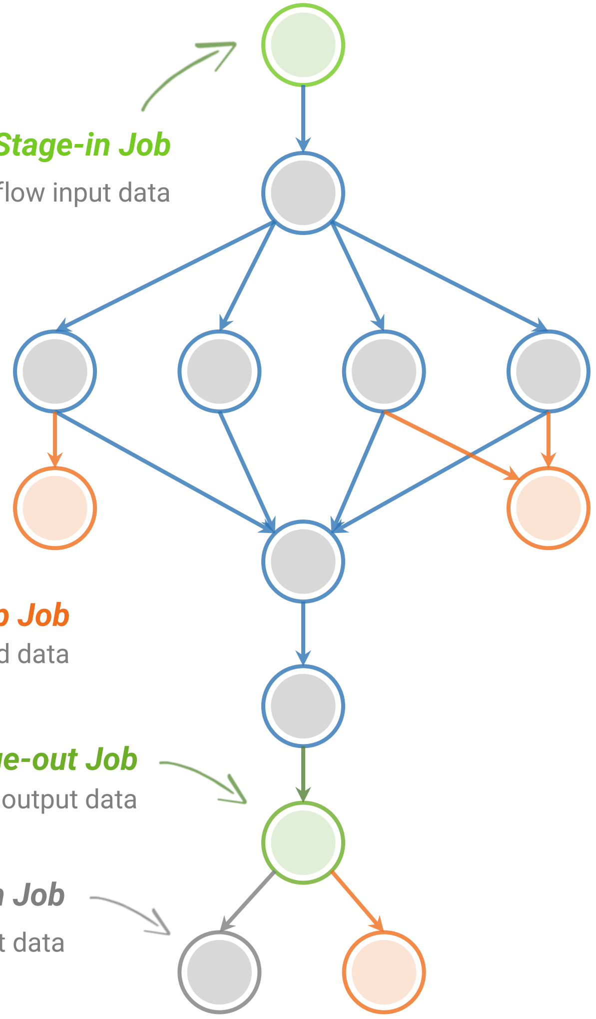
Removes unused data

Stage-out Job

Stage-out generated output data

Registration Job

Registers the workflow output data



EXECUTABLE WORKFLOW

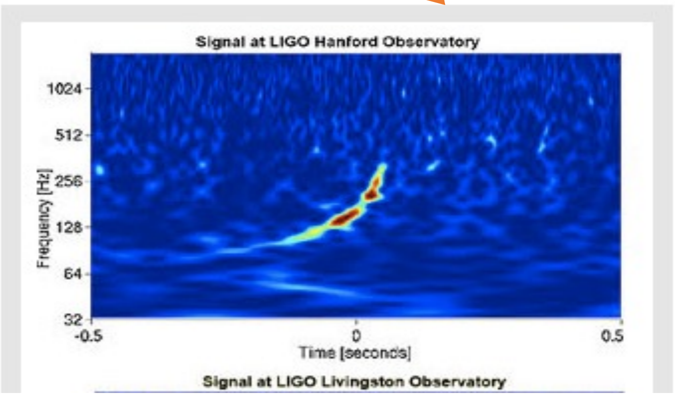
Pegasus: Support Science over Generations of CI

Nobel Prize

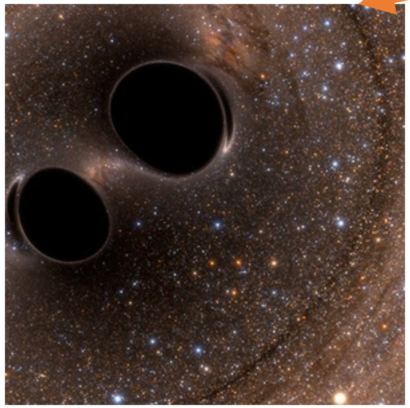
Working with LIGO (Laser-Interferometer Gravitational Wave Observatory)

2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

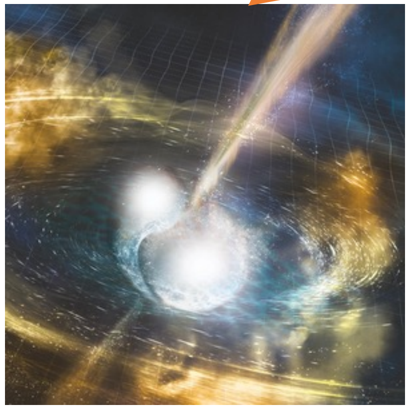
First Pegasus prototype



Blind injection detection



First detection of black hole collision



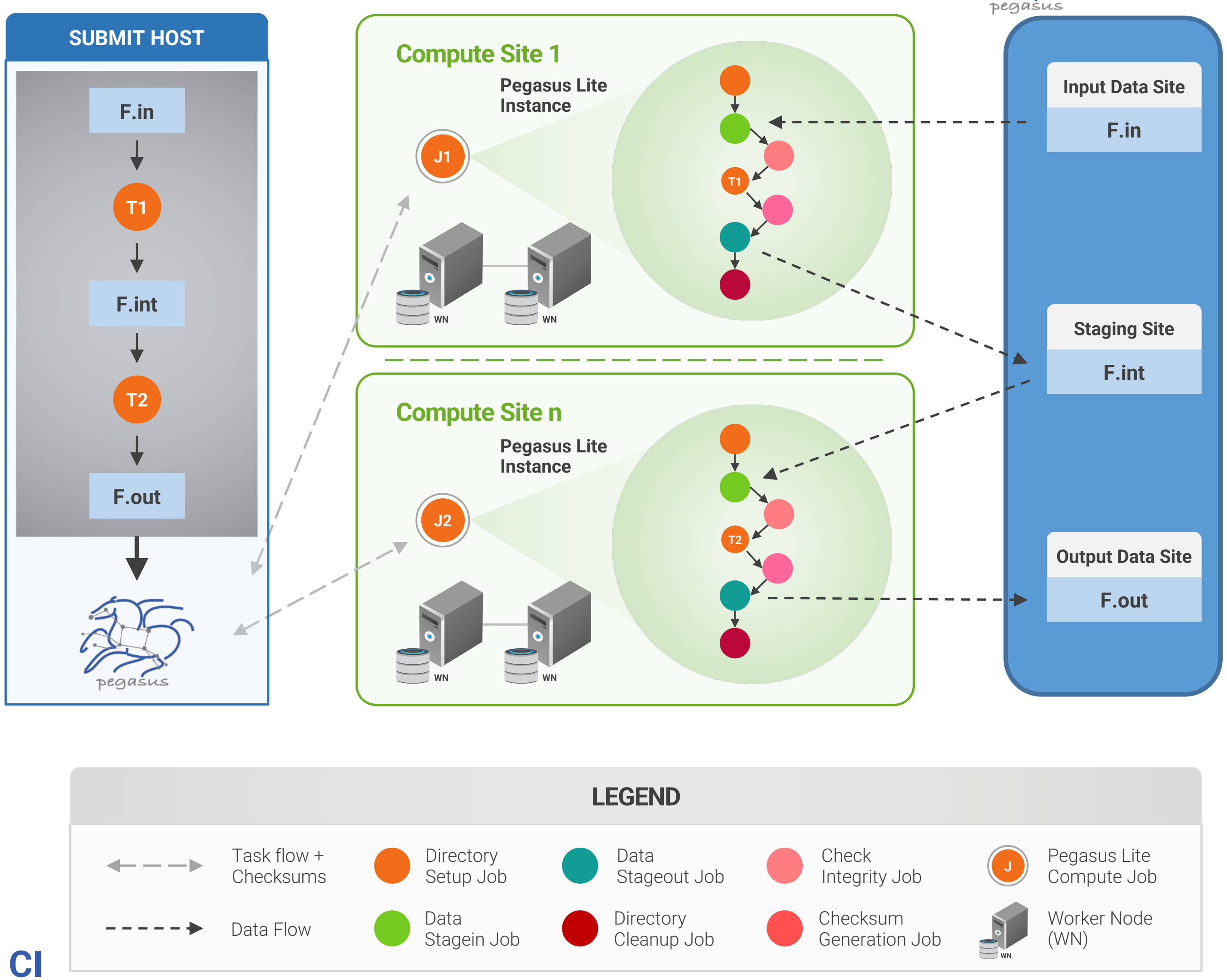
Multi-messenger neutron star merger observation

Image credit: LIGO Scientific Collaboration

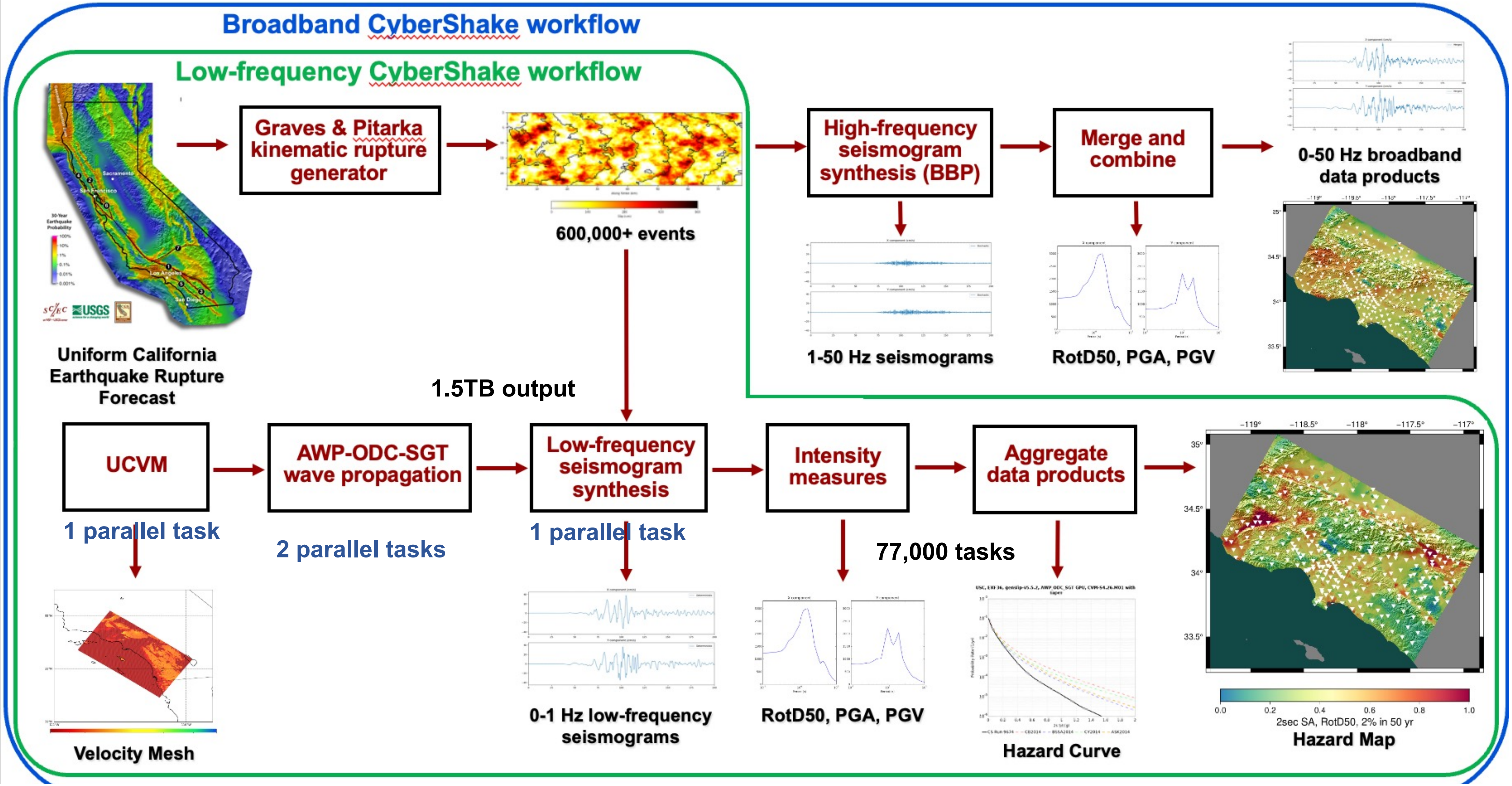
2. Submit locally, run globally

Pegasus

- **Pegasus WMS ==** **Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker**
  - Pegasus maps workflows to target infrastructure (1 or more resources)
  - DAGMan manages dependencies and reliability
  - HTCondor is used as a broker to interface with different schedulers
- **Planning converts an abstract workflow into a concrete, executable workflow**
  - Planner is like a compiler
  - Optimized performance
  - Provides fault tolerance
- **Can leverage distributed and heterogeneous CI**



Cutting-edge Science: Southern California Earthquake Center



Slide Courtesy of Scott Callaghan, USC

CyberShake Computational Requirements

CyberShake Stage	Number of Tasks	Node-Hours	Output Data
Velocity mesh creation (parallel)	1	10 CPU	300 GB
Wave propagation (parallel)	2	80 GPU	1500 GB
Low-frequency seismogram synthesis (parallel)	1	1000 CPU	38 GB
High-frequency seismogram synthesis (serial)	77,000	1000 CPU	187 GB
Total, 1 site (including small jobs)	77,020	2090	2025 GB
Total, full region	**25.8 million**	**700,000**	**680 TB**

- Large computational and data requirements
- Mix of large parallel CPU and GPU jobs with HTC
- High degree of automation required to support continuous execution

108 days of execution on ORNL's Summit using the the Pegasus Workflow Management System

Slide Courtesy of Scott Callaghan, USC

3. Flexible Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, …)

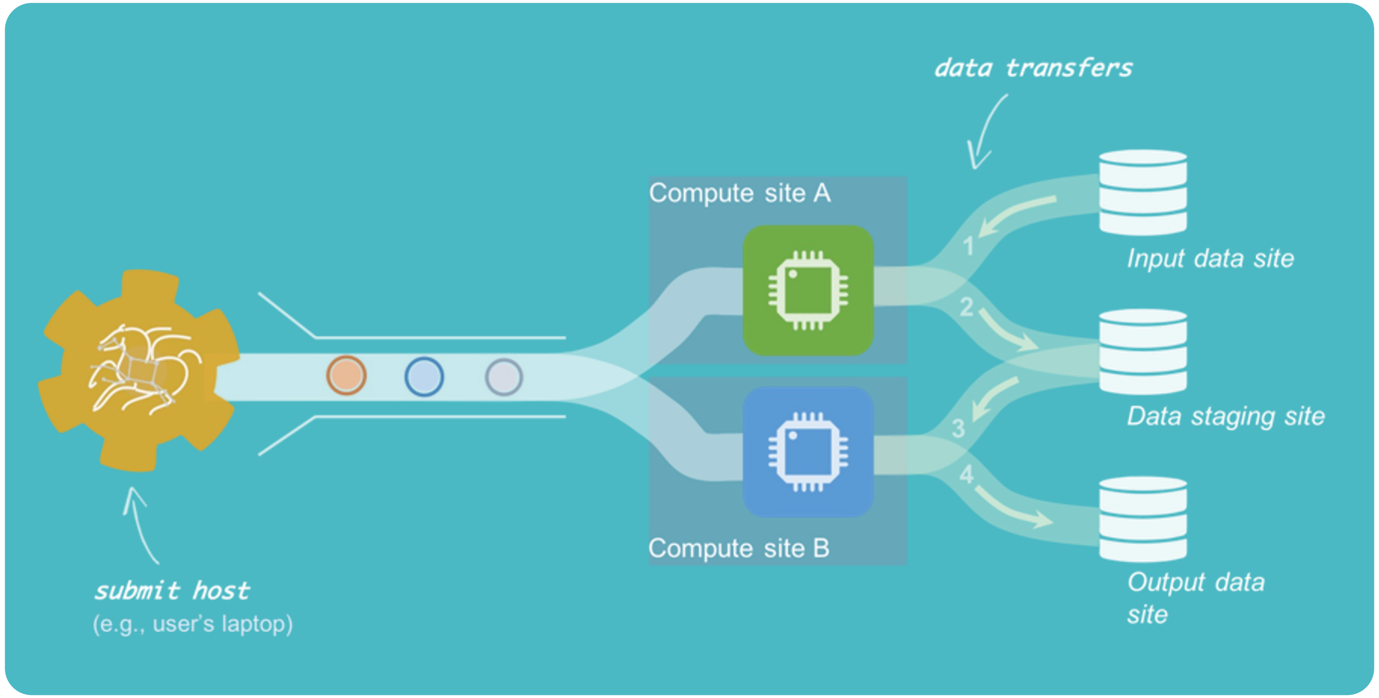
Worker nodes do not share a file system
Data is pulled from / pushed to the submit host via HTCondor file transfers
Staging site is the submit host

Shared File System
(HPC sites, ACCESS, Campus clusters, …)

I/O is directly against the shared file system

Non-shared File System (clouds, OSG, …)

Worker nodes do not share a file system
Data is pulled / pushed from a staging site, possibly not co-located with the computation

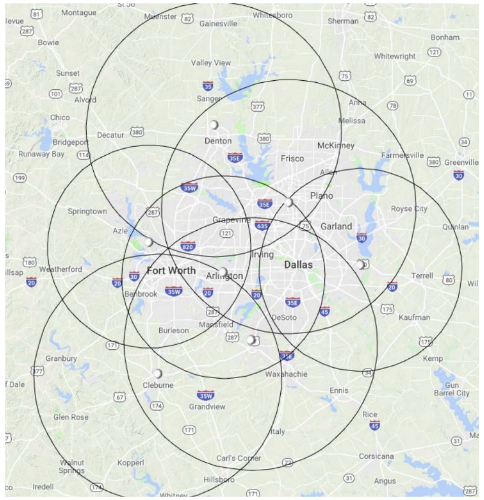


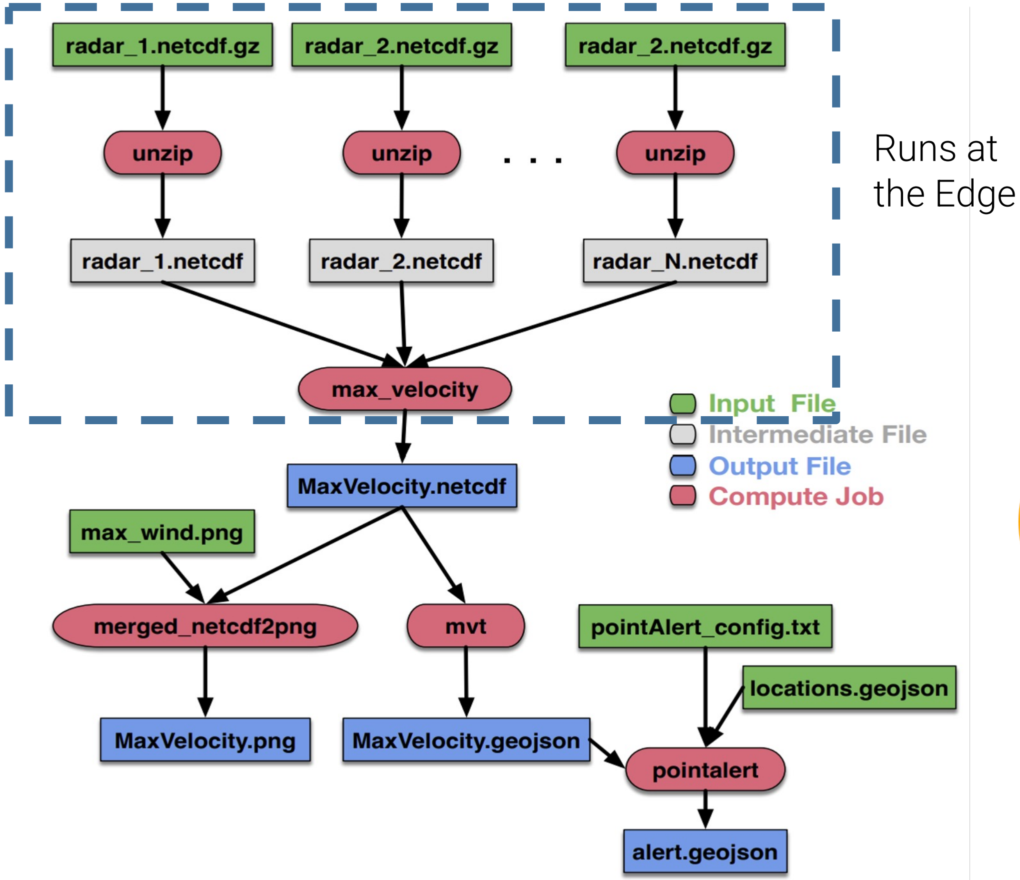
Edge-2-Cloud Applications

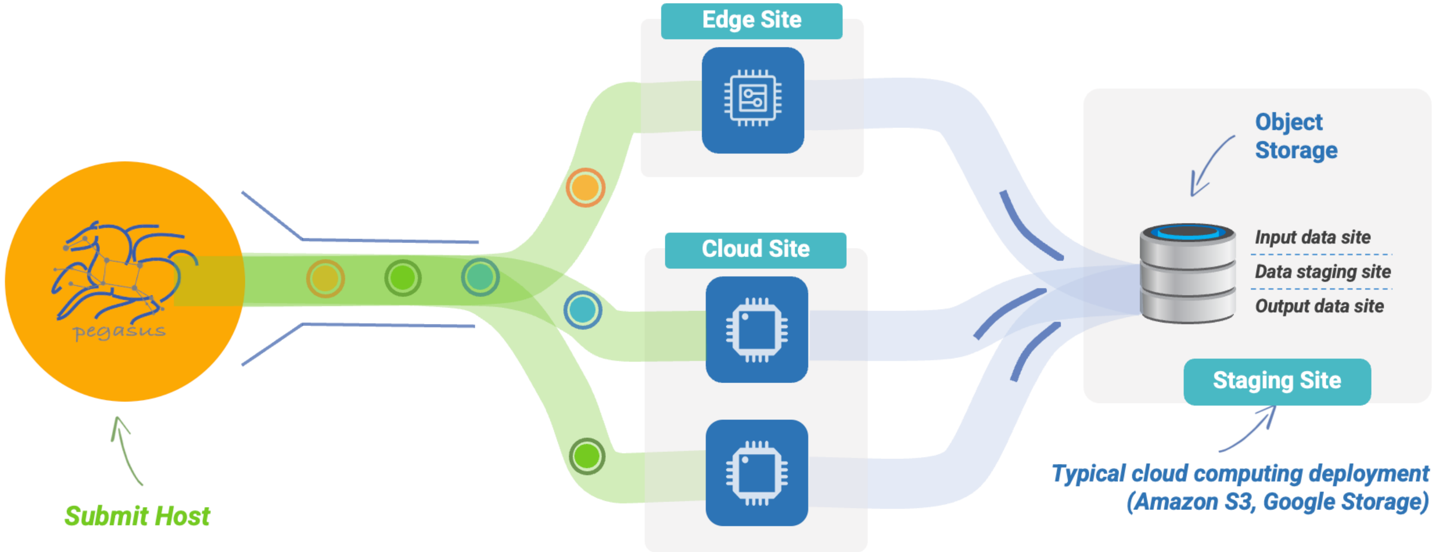
CASA: Collaborative and Adaptive Sensing of the Atmosphere

- Has deployed a network of short-range Doppler radars
- Compute and data repositories at the edge, close to the radars
- Use on demand cloud resources to scale up their computations

http://www.casa.umass.edu/







This work is funded by NSF, award #2018074

4. Flexible Data movement Pegasus-transfer

Pegasus' internal data transfer tool with support for a number of different protocols

- **Directory creation, file removal**
  - If protocol can support it, also used for cleanup
- **Two stage transfers**
  - e.g., GridFTP to S3 = GridFTP to local file, local file to S3
- **Parallel transfers**
- **Automatic retries**
- **Credential management**
  - Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

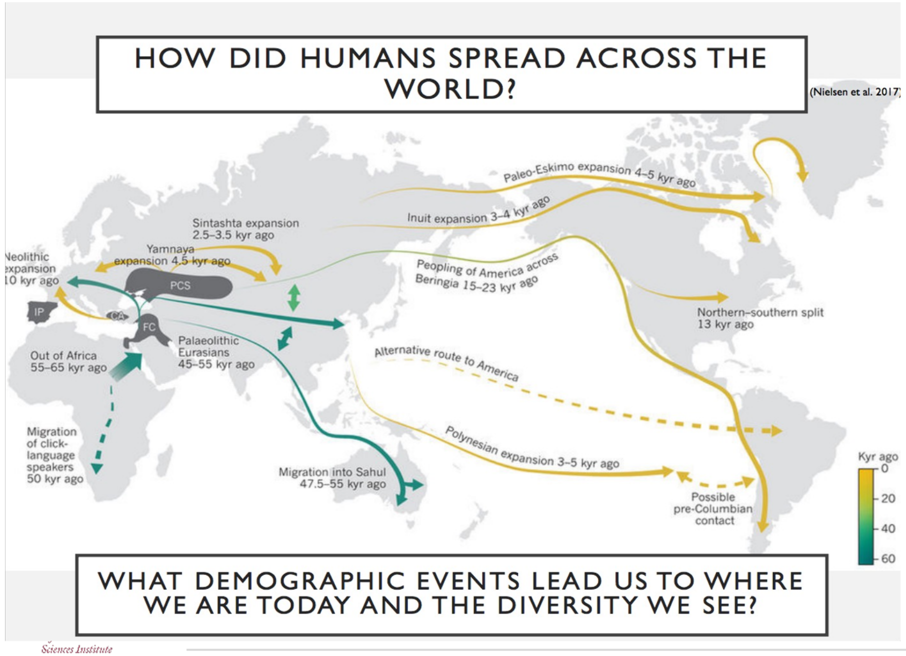
HTTP
SCP
GridFTP
Globus Online
iRods
Amazon S3
Google Storage
SRM
FDT
Stashcp
Rucio
cp
ln -s

Pegasus Workflow Applications

Automating the work of one scientist

Ariella Gladstein, Ph.D. Student
University of Arizona

- Need to perform a large amount of analysis on large-scale data sets
- Automatically adapt to the dynamic resources
- Need to have a record of how data was produced



Wall Hours by Facility

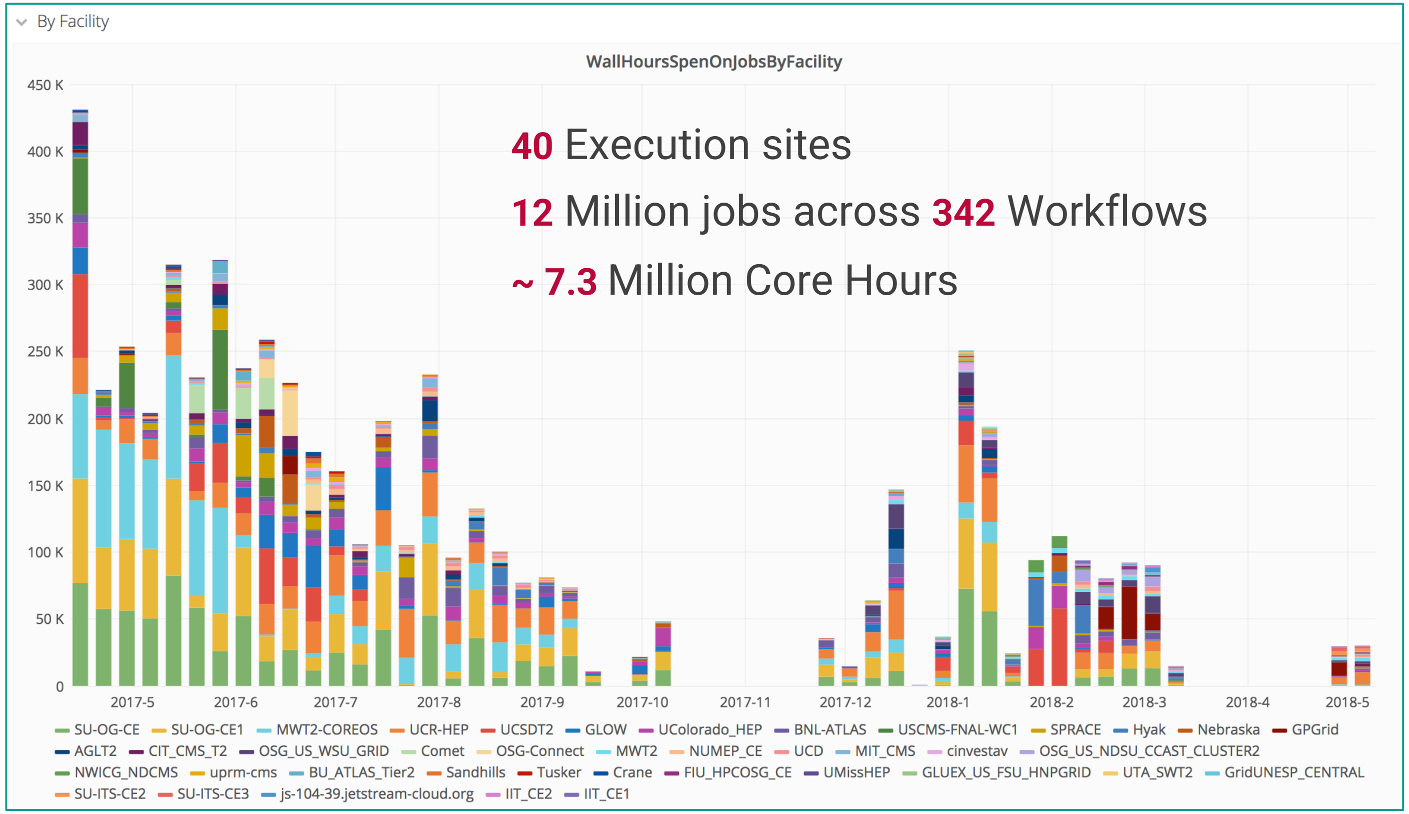
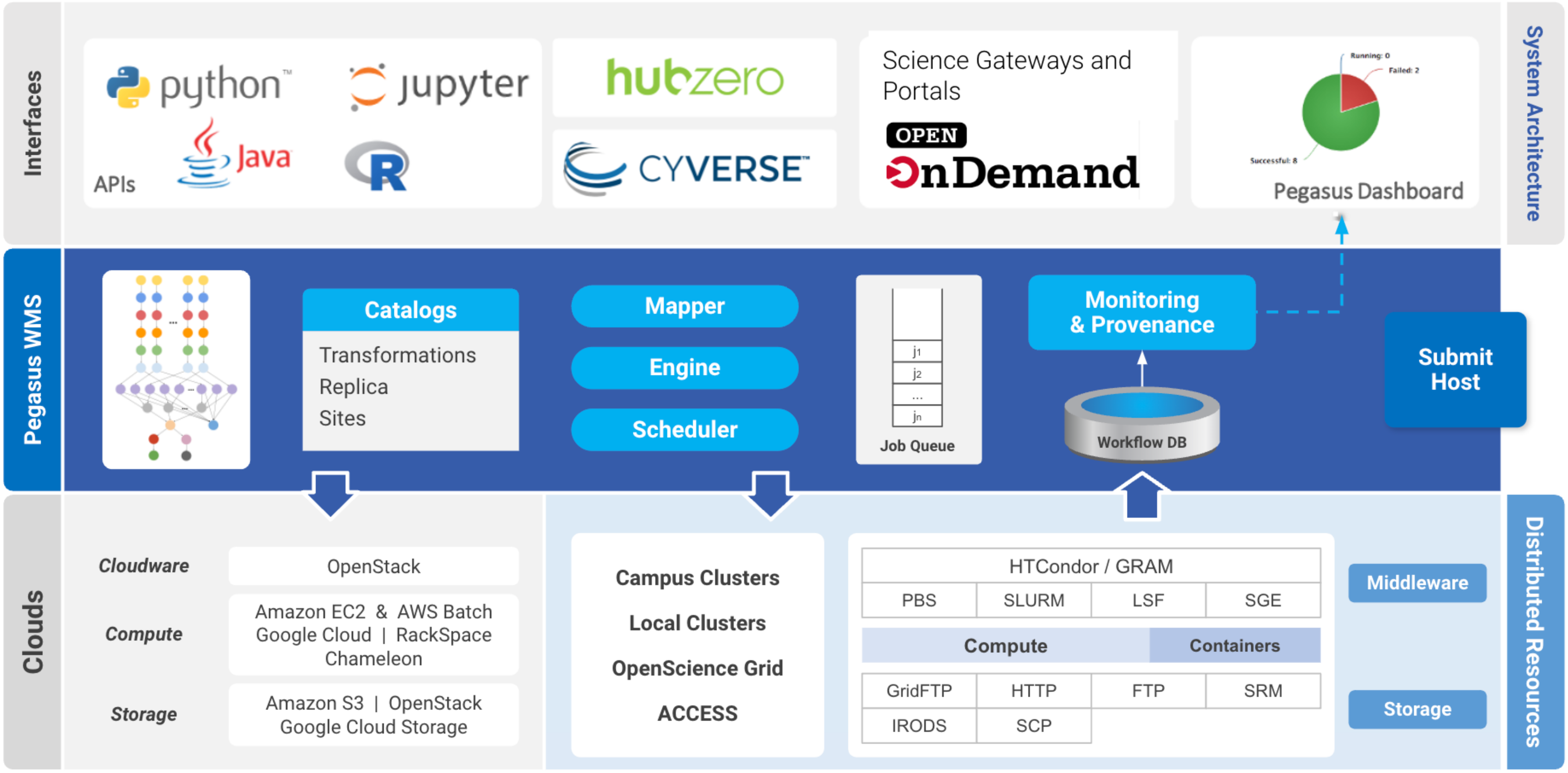
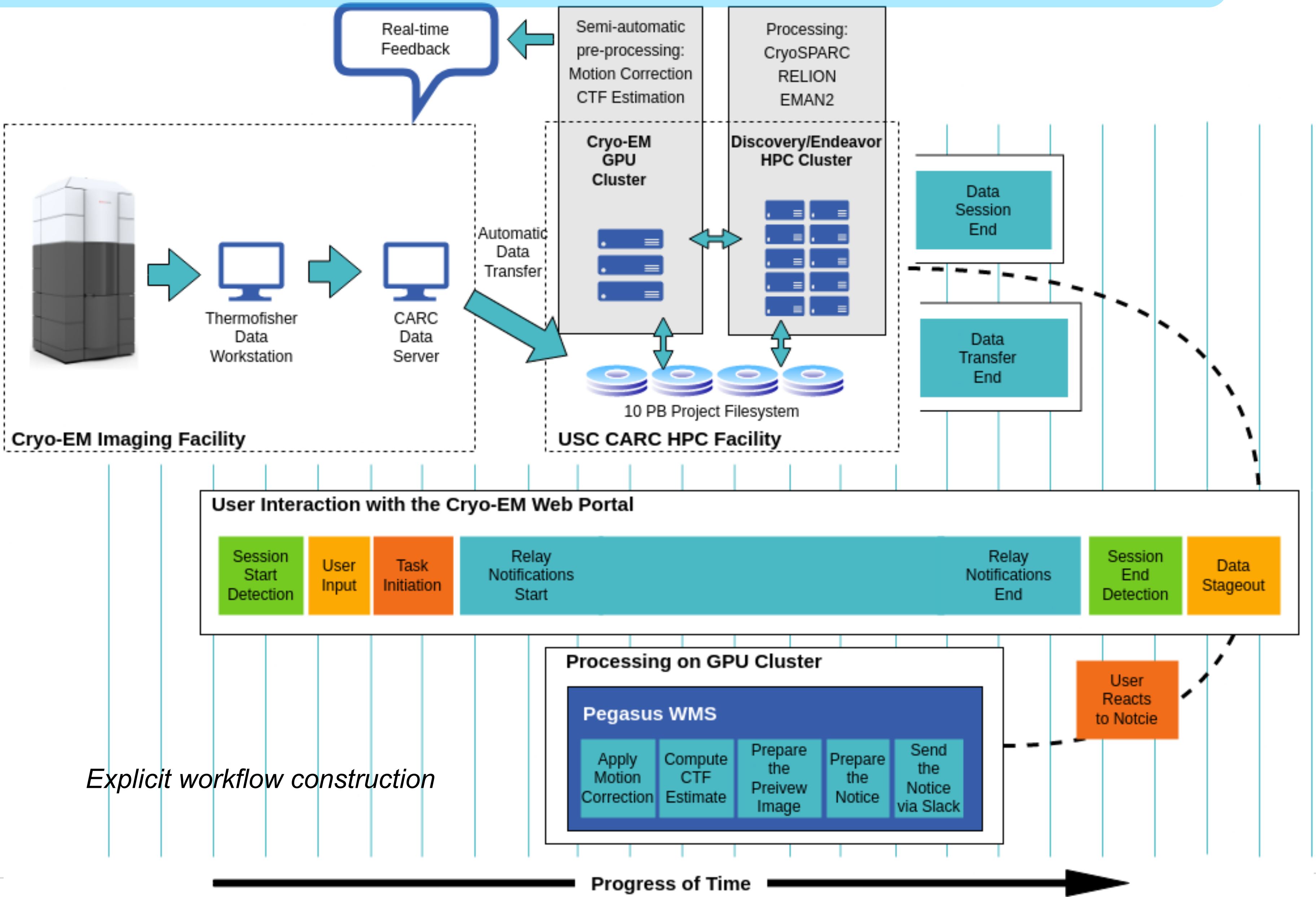


Image credit: Gladstein , Graph credit: Open Science Grid (OSG)

5. "Up and down" integrations with diverse CI, common languages, and Portal/GUI interfaces



Processing instrument data in real time



Hidden workflows

Explicit workflow construction

USC Viterbi School of Engineering Information Sciences Institute

PLANNING

Workflow Planning: Challenges and Solutions for Improving Performance and Robustness

pegasus

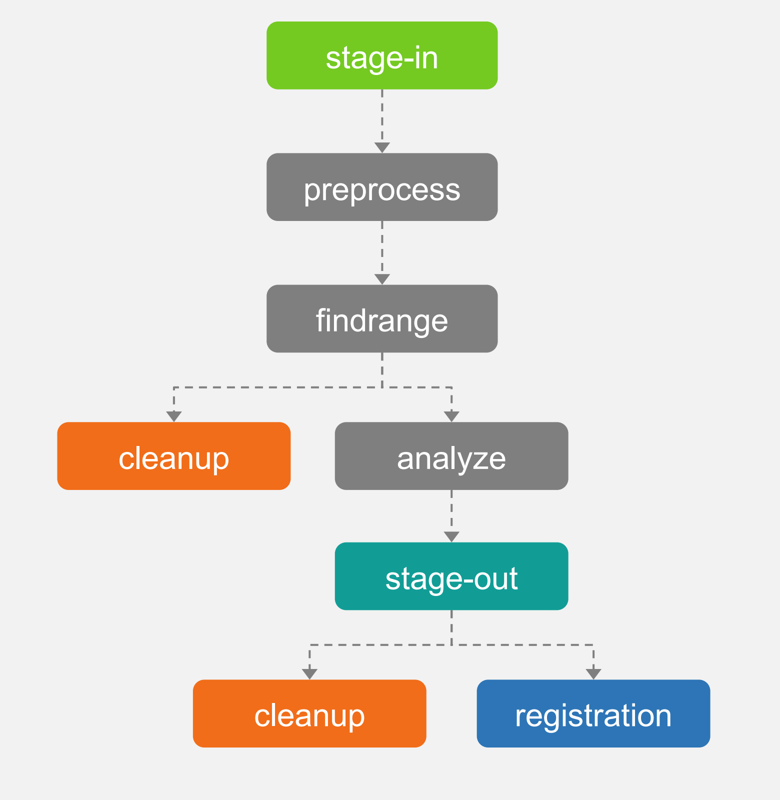
- Distributed execution environment
  - Design custom workflow engines and utilize the right tools for job submission and data management
- Workflow tasks can be small
  - Increase the tasks' computational granularity through task clustering
- Workflows can be large
  - Reduce the number of tasks through workflow partitioning
- Overcome system and network overheads in executing applications remotely
  - Provision resources and/or send more work at any one time
- Data need to be moved to the computation
  - Discover data and stage it across heterogeneous systems
- Computations need to be moved to the data (performance/privacy)
  - Make smart decisions, explore benefits/drawbacks

USC Viterbi School of Engineering Information Sciences Institute

Workflow Execution
Challenges and Solutions for Fault Tolerance

- Computations fail within a workflow
  - Automatically checkpoint the workflow, automate restart
- Resources fail
  - Automatically retry, or replan: try other resources (computing sites, data storage systems)
- Services fail (data movement, data registration)
  - Retry the action, choose a different service
- Run out of resources/Storage gets filled up
  - Analyze the workflow and clean up data no longer needed as the workflow execution progresses
- Data gets corrupted
  - Detect corruption/retry transfer

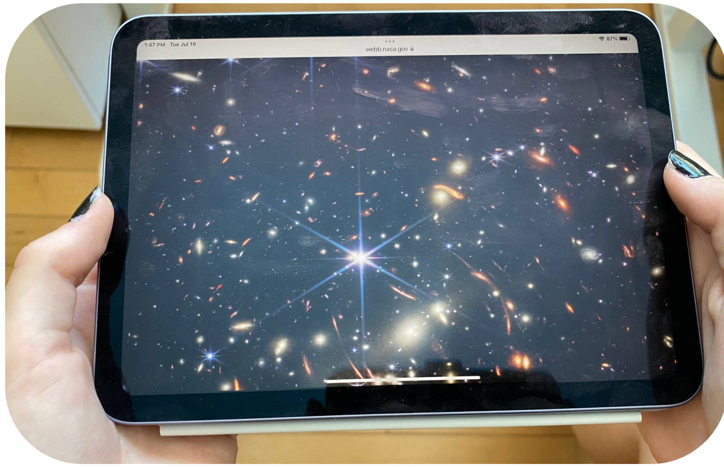
Data "cleanup"



USC Viterbi School of Engineering Information Sciences Institute

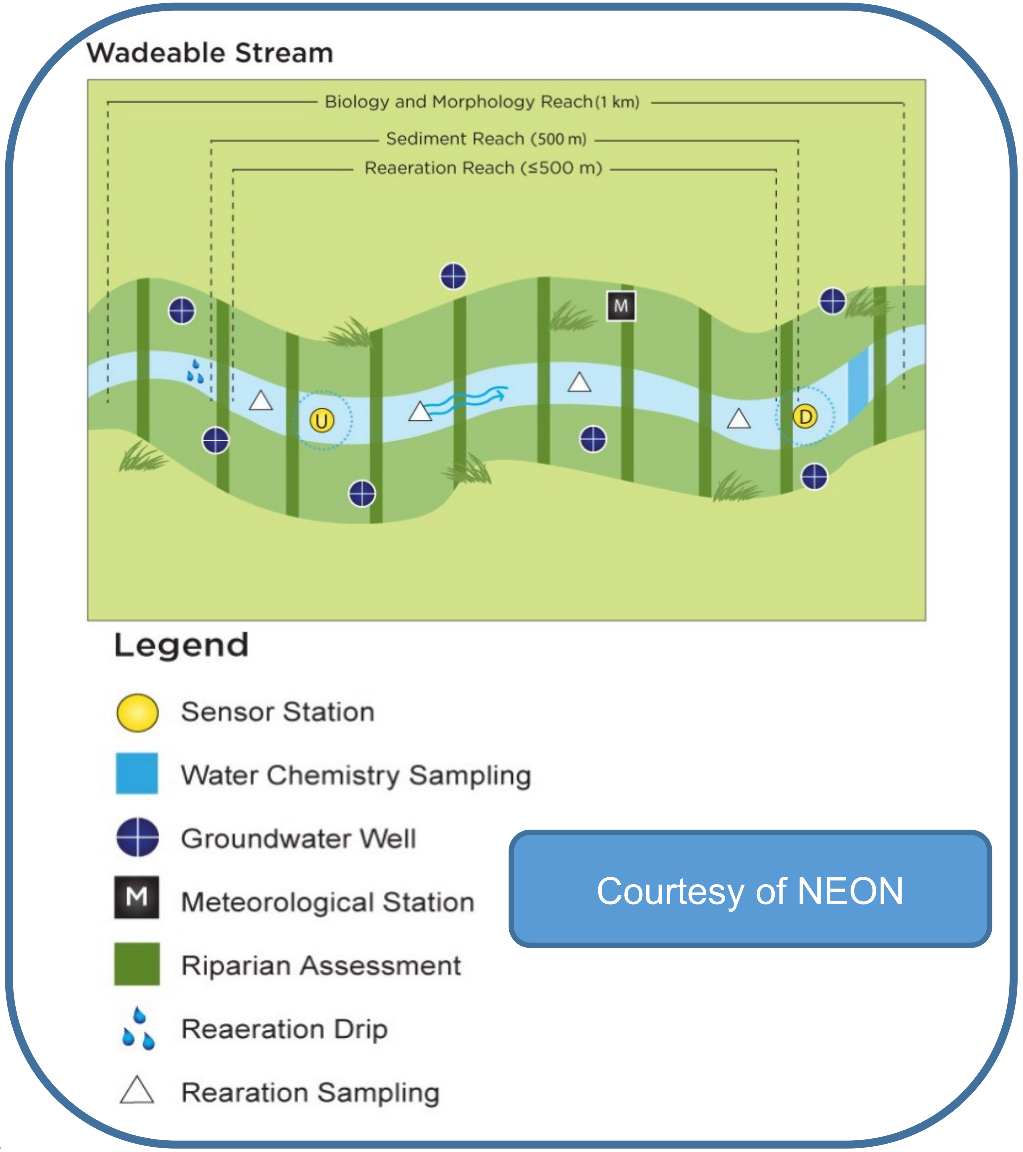
Users' Experiences and Expectations

- Users are often not exposed to complex programming
- Users are not exposed to command-line interfaces
- Users have uneven access to CI (even networks)
- Expect easy to use, intuitive interfaces
    - Graphical, conversational, common behavior
- Expect robust systems that are fault tolerant and adaptable
- Want quick response time and/or good information
- Current cyberinfrastructure (CI) is very complex, heterogeneous, and fragmented
- Even simple tasks (remote job submission, monitoring, debugging) are difficult (2-factor authentication)
- Limited support for long-running services and dynamic resource management



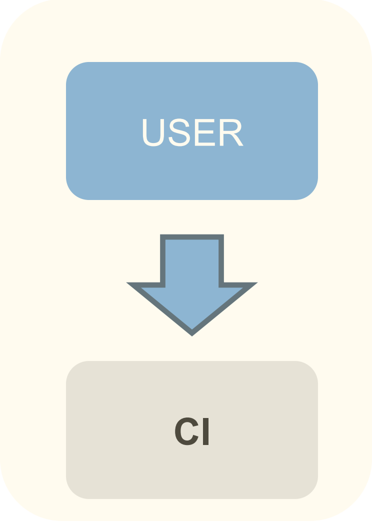
James Webb Telescope Image

Means and Methods are changing



- **More Data and Data Sources**
  - Small and cheap sensor devices
  - Sophisticated instruments (CryoEM ~7TB per experiment)
  - Large-scale data archives: LHC, NASA, (Rubin will produce 20TB/night)
- **Faster Networks**
  - 100Gbps, ESnet6 > 400 Gbps
- **Access to more computing**
  - OSG's OSPool can reach 70,000 cores in a single day, 75M jobs/year (spanning 50 campuses and organizations)
  - OLCF's Frontier reached exascale in 2022

What can we do better?



ChatGPT for workflow creation

DE

Create a Pegasus WMS Python workflow generator for a workflow that has 3 steps, first step divides the input data file into 10 pieces, the second step takes each file and does and fft on it, and the final step collects the results.

Pegasus is a workflow management system that allows users to design workflows in abstract terms and execute them efficiently and reliably on complex computing resources. Writing a workflow to divide a file into 10 pieces, apply Fast Fourier Transform (FFT) to each piece, and then collect the results can be done by defining tasks and dependencies between them in a workflow.

Here's an example of how you might set up this workflow using Pegasus WMS with Python:

1. **Step 1:** Divide the input data file into 10 pieces.
2. **Step 2:** Perform an FFT on each piece.
3. **Step 3:** Collect the results.

First, ensure you have the necessary dependencies installed, including Pegasus:

bash Copy code

```bash
pip install Pegasus-wms
```

ChatGPT for workflow creation

```
# Create the workflow
wf = Workflow("fft_workflow")

# Create the input file
input_file = File("input.txt")

# Step 1: Divide the input data file into 10 pieces
output_files_step1 = []
for i in range(10):
    output_file = File(f"part-{i}.txt")
    output_files_step1.append(output_file)
    job_split = Job(split_file)\
        .add_args(input_file, i, 10)\
        .add_inputs(input_file)\
        .add_outputs(output_file)
    wf.add_jobs(job_split)

# Step 2: Perform FFT on each piece
output_files_step2 = []
for i, input_file in enumerate(output_files_step1):
    output_file = File(f"fft-{i}.txt")
    output_files_step2.append(output_file)
    job_fft = Job(fft_transform)\
        .add_args(input_file)\
        .add_inputs(input_file)\
        .add_outputs(output_file)
    wf.add_jobs(job_fft)
```

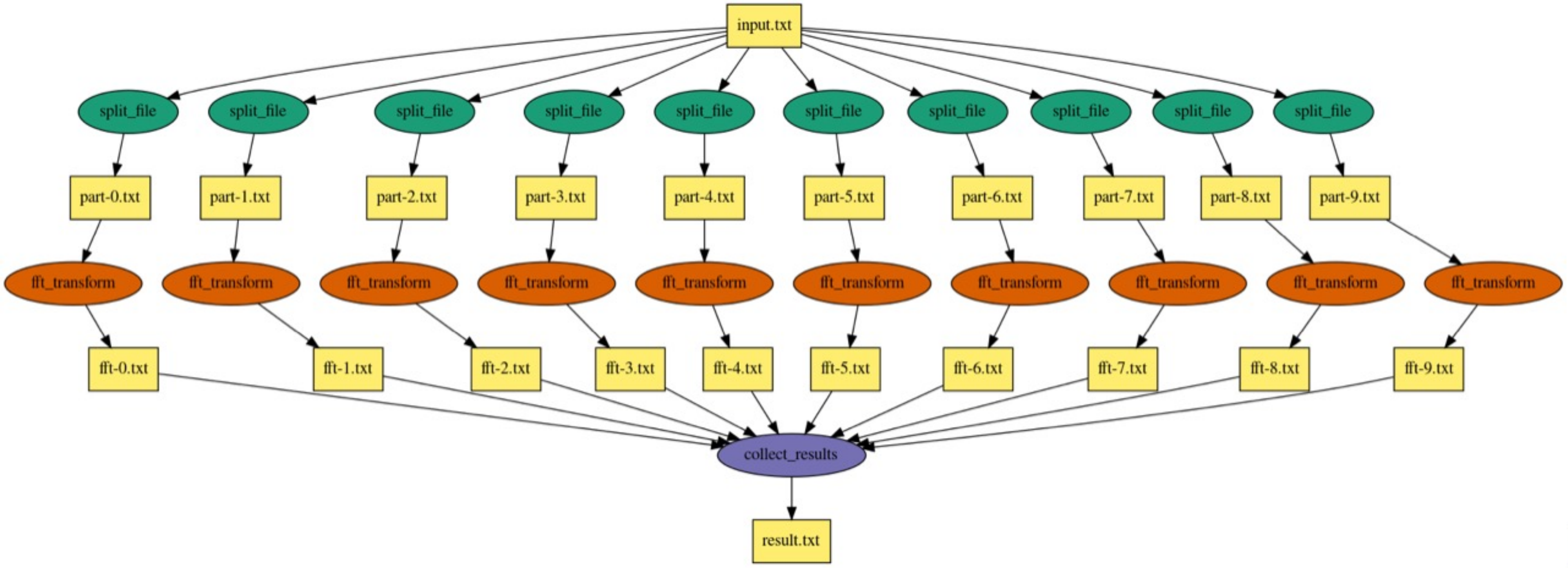
You would need to have corresponding scripts `split_file.py`, `fft_transform.py`, and `collect_results.py` to carry out the splitting, FFT, and collection processes respectively.

Make sure to customize the paths to the transformation scripts and modify the transformations to suit your specific requirements. The above example sets up the structure of the workflow, but the details of each task would depend on the specific operations you want to perform on the da

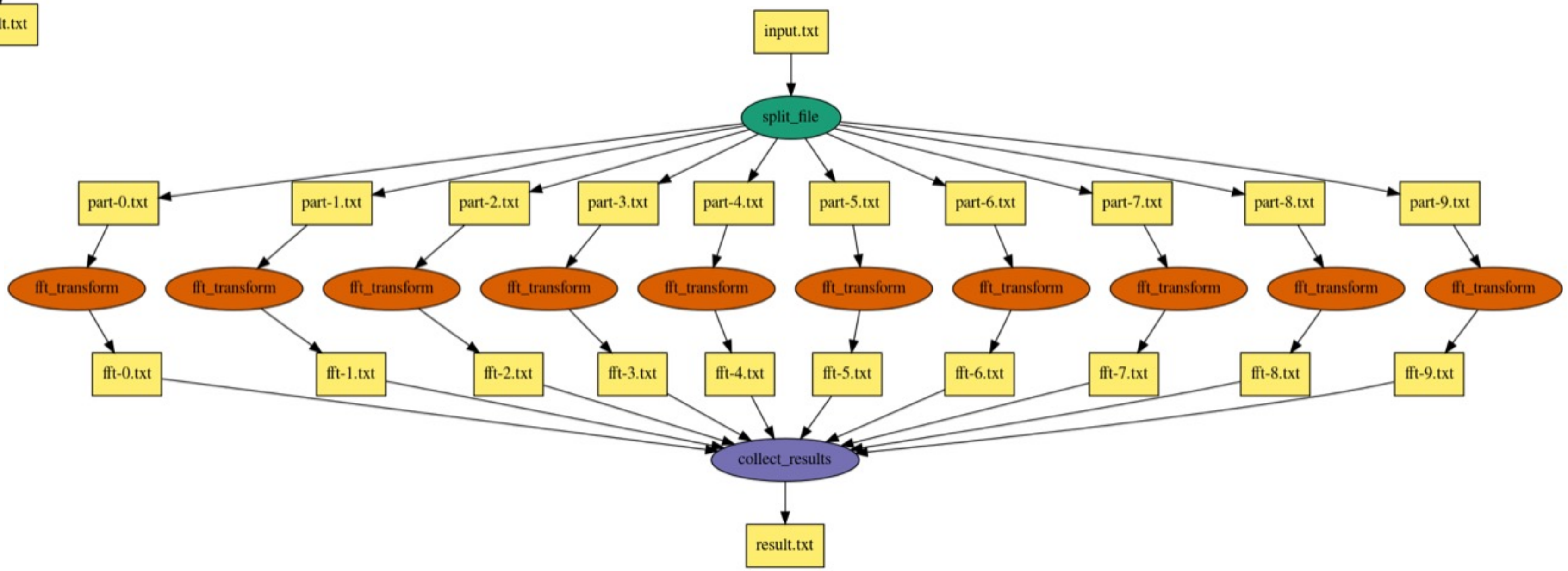
Magic number
You can ask ChatGPT to fix

```
# Define the number of pieces
n = 10 # Change this to the desired value

# Step 1: Divide the input data file into n pieces
output_files_step1 = []
for i in range(n):
    output_file = File(f"part-{i}.txt")
    output_files_step1.append(output_file)
    job_split = Job(split_file)\
        .add_args(input_file, i, n)\
        .add_inputs(input_file)\
        .add_outputs(output_file)
    wf.add_jobs(job_split)

# ... (rest of the code remains the same)
```

ChatGPT for workflow creation



ChatGPT



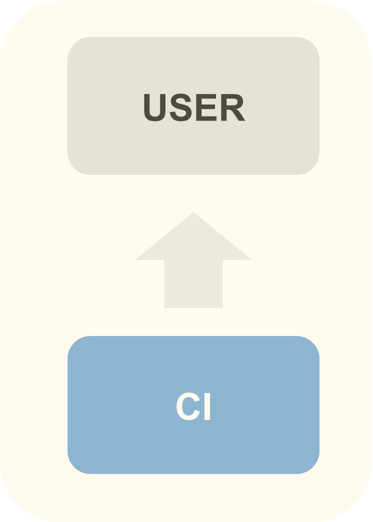
Karan Vahi, M.S.
SENIOR COMPUTER SCIENTIST

USC Viterbi School of Engineering
Information Sciences Institute

What can we do better? Can we use ML to make our systems "smarter"/more autonomous

PoSeiDon

- Anomaly detection
- Anomaly/error classification and attribution
- Predictive models of performance
- Better workflow adaptation based on failures and anomalies
- Challenges:
  - Collect enough (quality data, richness, balanced class representation)
  - Enough labeled data, need to augment data
  - Structure (normalize, scale, transform) the data in a way that is amenable to the application of current techniques (or develop new ones)
  - Select the appropriate ML algorithms or architectures
    - DL hyperparameter optimization (learning rate, #epochs, hidden layers, activations functions..)



This work is funded by DOE under grant #DE-SC0022328

Anomaly Detection Framework

- **Data processing**: process simulated anomalies on workflows, parse logs as
  - **Tabular** (features as columns)
  - **Image** (Gantt charts)
  - **Graph** (nodes as jobs, edges as dep.)
  - **Text** (sentences describing jobs)
- **Build base models**: supervised / unsupervised learning to identify the anomalies by deep learning
- **Analytics**: improve the performance, quantify uncertainty, provide explanation, etc.

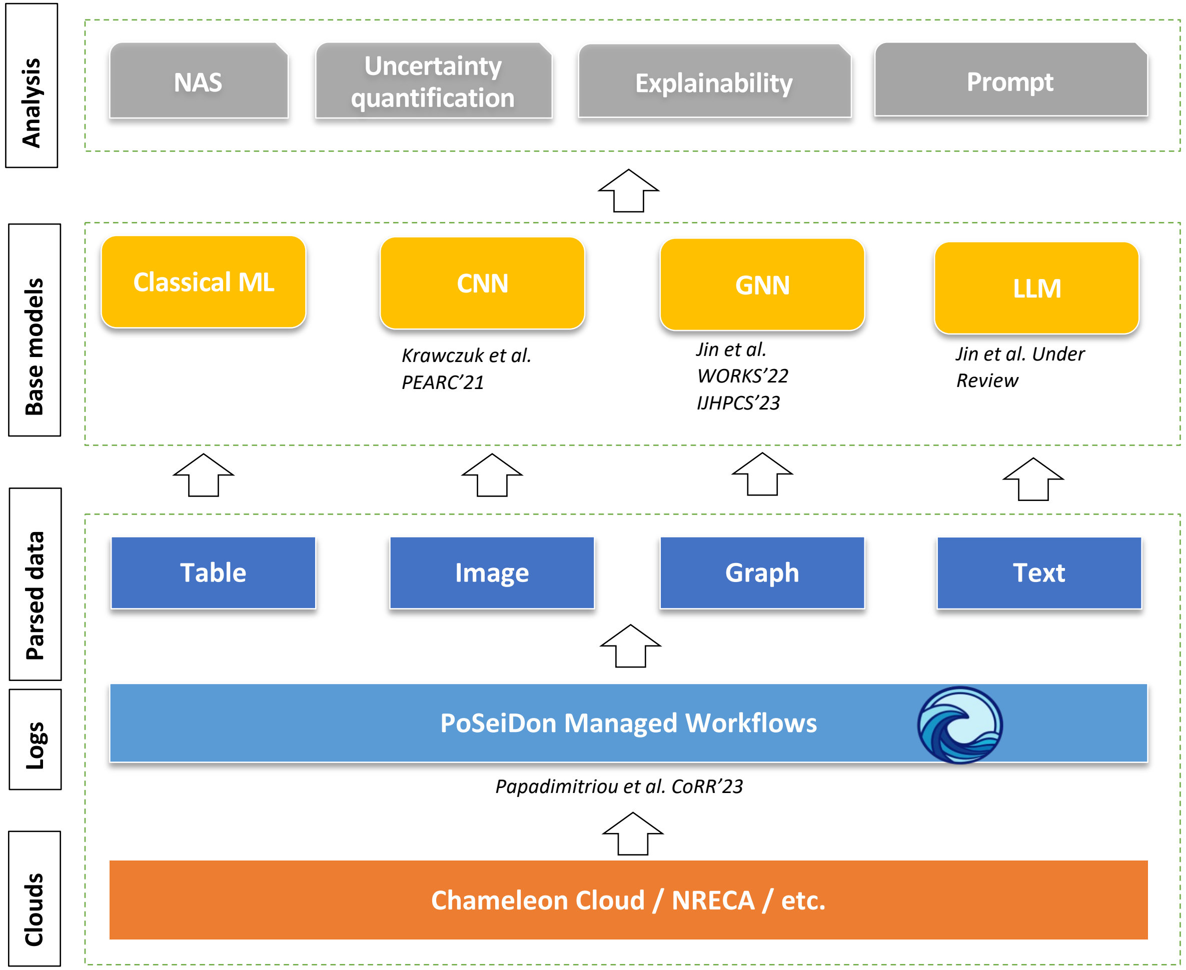
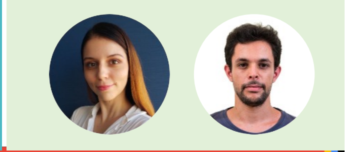


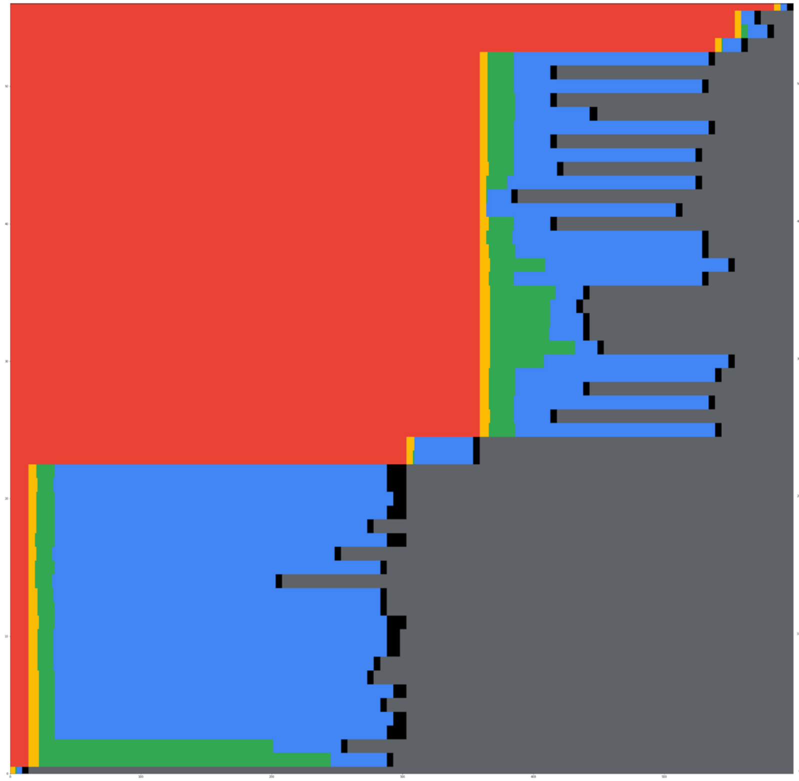
Fig. Anomaly Detection Framework

Identifying anomalies and their causes

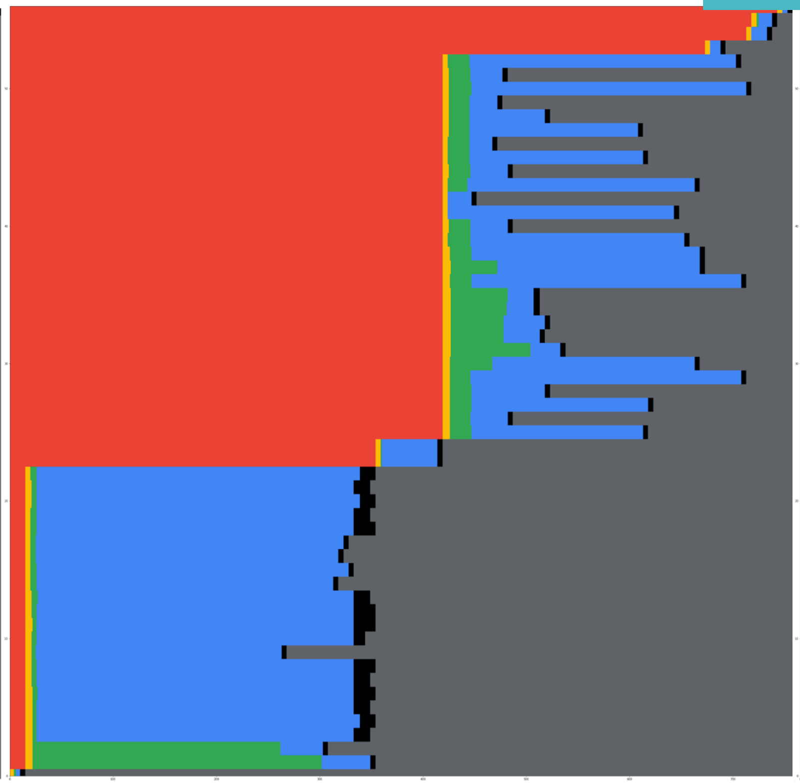
Gantt Charts: normal execution and different anomalies:
hard drive load, network packet loss

Work by Patrycja Krawczuk
and George Papadimitriou

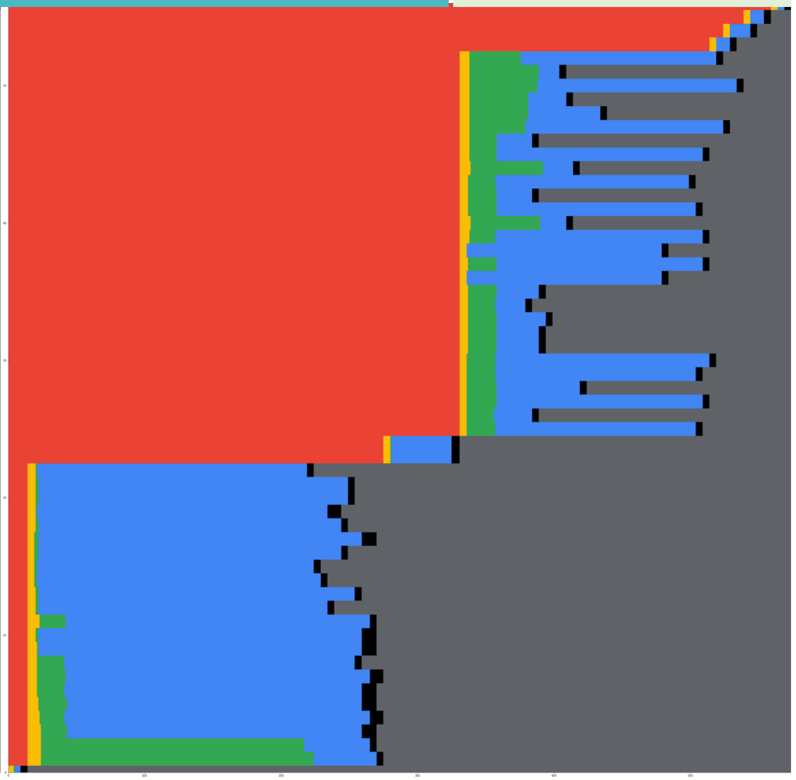




normal_1000genome-20200616T174351Z-run0044.png



hdd_50_1000genome-20200610T041238Z-run0006.png



loss_0.5_1000genome-20200520T031010Z-run0017.png

ready_delay | wms_delay | queue_delay | runtime | post_script_delay | finished

90% accuracy on the workflows we trained on

Robust Execution: Anomaly Detection and Classification using Graph Neural Networks (GNN)

PoSeiDon

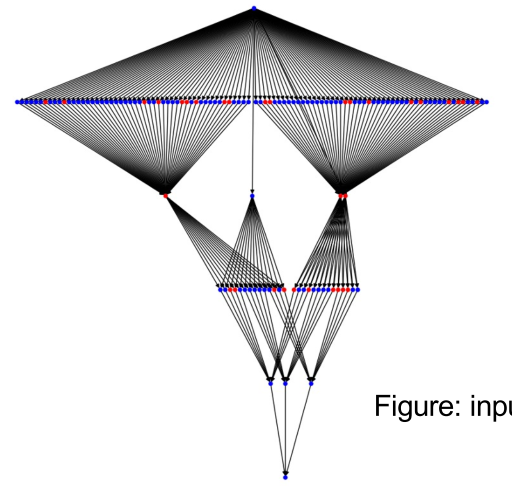


Figure: input data

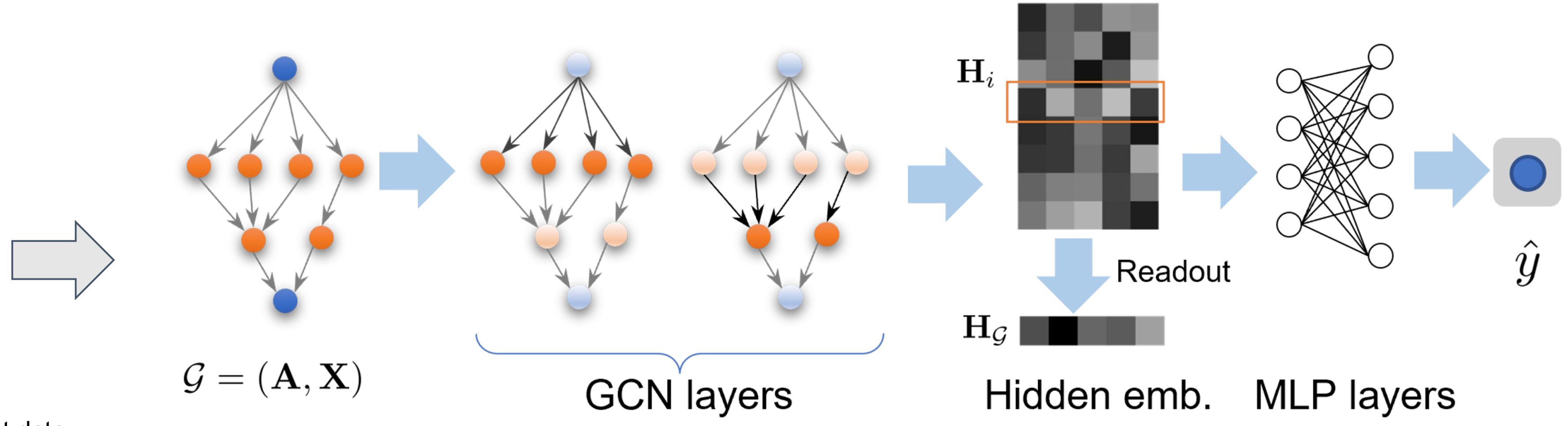


Figure: Graph neural networks architecture

Input: directed acyclic graphs (DAGs) represent normal and anomaly workflows

Output: the normal/anomaly labels for workflow-level (entire graph) and job-level (single node)

- Our GNN models achieved 25% improvement accuracy over conventional methods for anomaly detection.
- We achieve 2-4 times faster training time when compared with conventional machine learning models.
- Developing explainable AI methods to explain anomalies in the workflow performance.

USC Viterbi School of Engineering · renci · Argonne National Laboratory · Berkeley Lab

This work is funded by DOE under grant #DE-SC0022328

Graph Neural Networks - performance

Available workflows

Single model for multi-workflows

Workflow	Binary				Multi-label
	Accuracy	F1	Recall	Precision	Accuracy
1000 Genome	$0.917 \pm .014$	$0.915 \pm .019$	$0.921 \pm .009$	$0.938 \pm .010$	$0.882 \pm .006$
Nowcast w/ clustering 8	$0.768 \pm .009$	$0.715 \pm .017$	$0.778 \pm .023$	$0.768 \pm .15$	$0.792 \pm .009$
Nowcast w/ clustering 16	$0.837 \pm .012$	$0.675 \pm .020$	$0.815 \pm .012$	$0.837 \pm .011$	$0.830 \pm .007$
Wind w/ clustering casa	$0.776 \pm .002$	$0.652 \pm .032$	$0.769 \pm .021$	$0.776 \pm .017$	$0.764 \pm .19$
Wind w/o clustering casa	$0.781 \pm .02$	$0.853 \pm .013$	$0.800 \pm .012$	$0.781 \pm .008$	$0.886 \pm .007$
1000 Genome (partial anomaly)	$1.000 \pm .0$	$1.000 \pm .0$	$1.000 \pm .0$	$1.000 \pm .0$	$1.000 \pm .0$
ALL	$0.836 \pm .006$	$0.878 \pm .013$	$0.886 \pm .011$	$0.856 \pm .009$	$0.877 \pm .008$

Figure: Graph-level classification

Model	Acc.	Recall	Prec.	F1
SVM	0.622	0.622	0.667	0.550
MLP	0.874	0.874	0.875	0.874
RF	0.898	0.898	0.908	0.887
AlexNet	0.910	0.914	0.910	0.910
VGG-16	0.900	0.900	0.900	0.900
ResNet-18	0.910	0.916	0.910	0.910
Our GNN	**0.917**	**0.921**	**0.939**	**0.915**

Gantt Chart

Figure: Model comparison

SVM: Support vector machines (SVMs)
MLP: Multilayer perceptron with hidden layers (128, 128, 128)
RF: Random forest with maximum depth set to 3.
(AlexNet,…) Gantt Chart: computer vision inspired DNN by generating Gantt charts from node features.

Means and Methods are changing

- More AI-based methods are being used to analyze and synthesize data, chose solutions, etc..
- LLM used to generate codes: ChatGPT, Google Copilot
- More black boxes in the methods—not only AI, lots of software available for reuse and repurposing

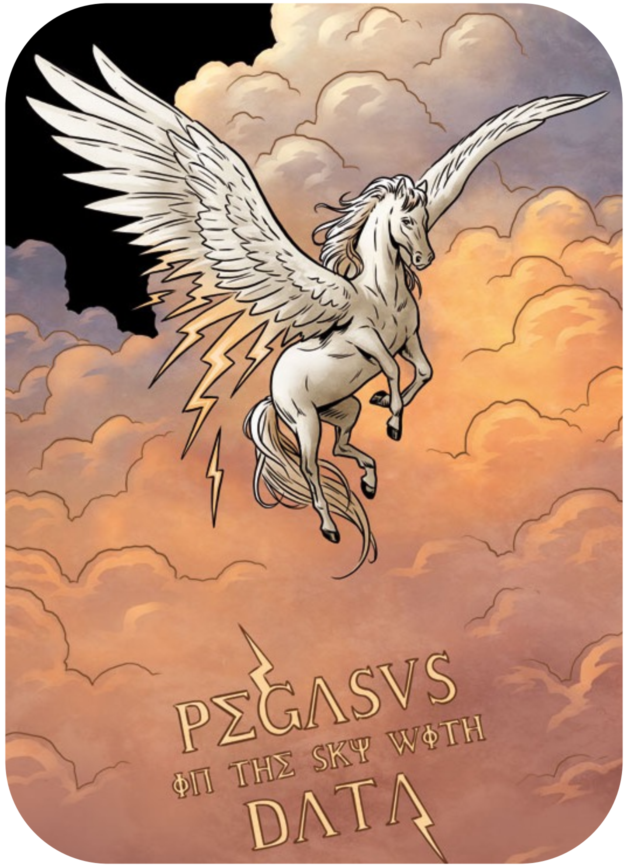
Growing emphasis on validation

Growing need for Automation

- Meta-analysis: Need be be able to easily re-run the same analysis with different data and parameters (potentially a large number of times)
- Easily replicate our own work
- Easily reproduce others' finding

Conclusions

- The world is changing around us (quickly)
  - Users' experiences and expectations
  - Means and methods are growing more complex and less tractable
- We need to expand more effort to support accessible, robust and open science (reusability, scalability, reproducibility, trustworthiness)
- Workflow and resource management systems and other CI should continue to increase the level of automation, component reuse, and ease of use
- We need to explore how we can better systematize system development to support CI component reuse and development and improve user experience.



http://pegasus.isi.edu