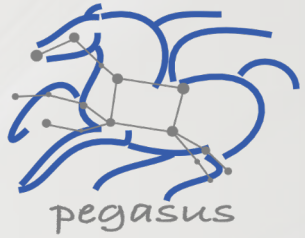


Pegasus, a Workflow Management Solution For Emerging Computing Systems

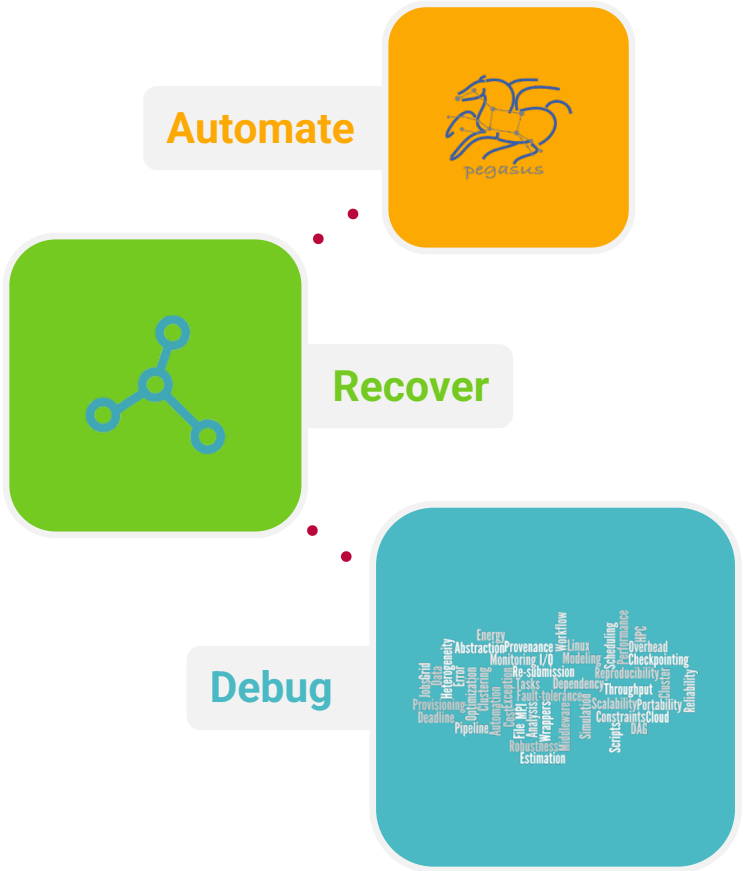
Ewa Deelman

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

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

USC Viterbi School of Engineering Information Sciences Institute

Pegasus Workflow Management System

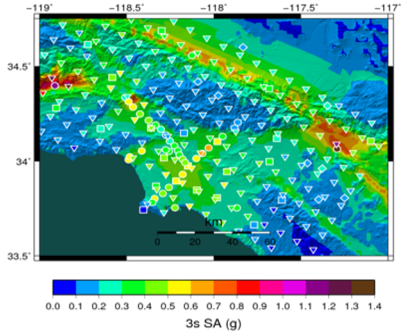


- **Automates Complex**, Multi-stage Processing Pipelines
- Enables Parallel, **Distributed Computations**
- **Automatically Executes** Data Transfers
- Reusable, Aids **Reproducibility**
- Records How Data was Produced **(Provenance)**
- Handles **Failures** to Provide Reliability
- Keeps Track of Data and **Files**

HTCondor High Throughput Computing

NSF funded project since 2001,
with close collaboration with HTCondor team

Southern California Earthquake Center's CyberShake



Mix of MPI and single-core jobs, mix of CPU, GPU codes.
Large data sets (10s of TBs), ~300 workflows with 420,000 tasks each
Supported since 2005: changing CI, x-platform execution

First Physics-Based "Shake map" of Southern California

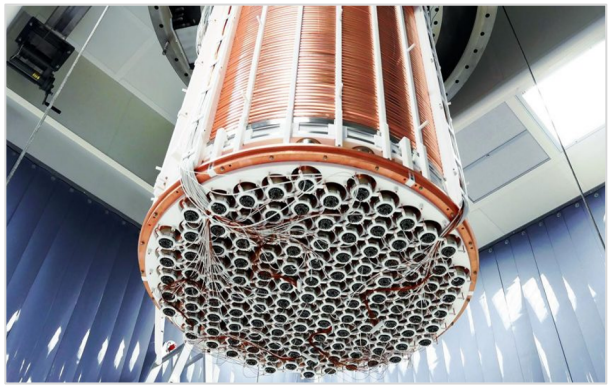
Laser Interferometer Gravitational-Wave Observatory (LIGO)

High-throughput computing workload, access to HPC resources, ~ 21K Pegasus workflows, ~ 107M tasks

Supported since 2001, distributed data, opportunistic computing resources

First direct detection of a gravitational wave (colliding black holes)

XENONnT - Dark Matter Search



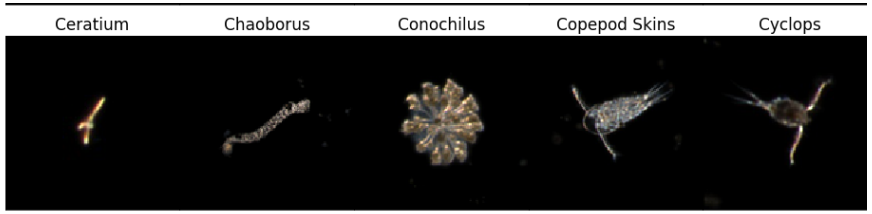
Custom data management
Rucio for data management
MongoDB instance to track science runs and data products.

Monte Carlo simulations and the main processing pipeline.

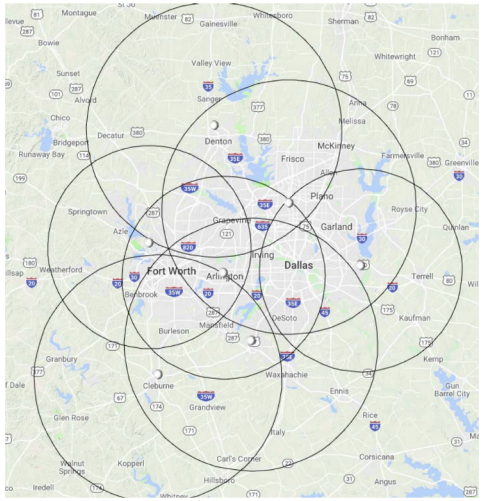
Edge Instruments and Sensors

Sensor data are increasingly used across science* domains

- Zooplankton classification
- Weather forecasting
- Study of the marine life



https://www.frontiersin.org/articles/10.3389/fmicb.2021.746297/full



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/

OOI: Ocean Observatories Initiative

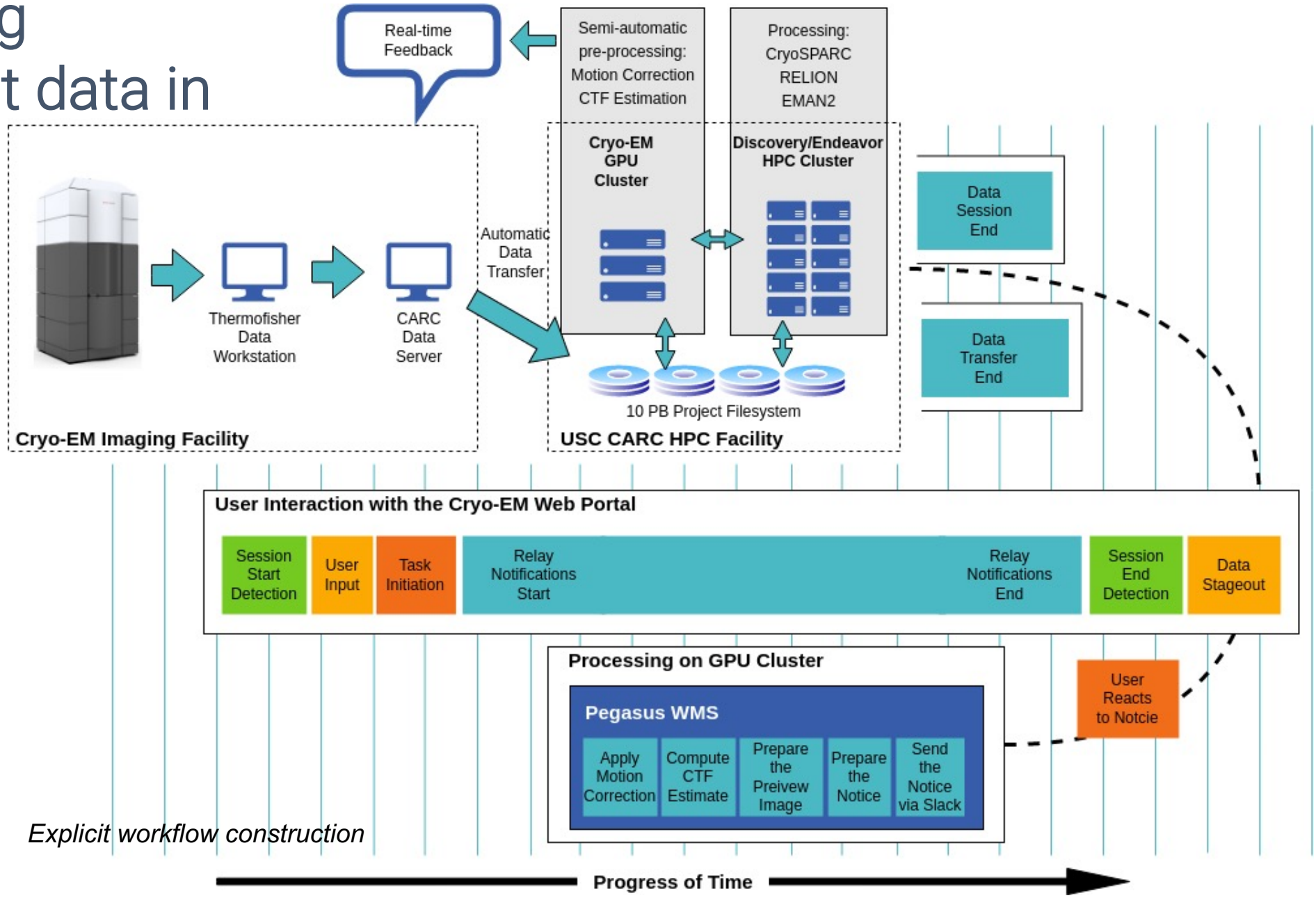
- Has deployed a variety of sensors in the Atlantic and the Pacific oceans to study the oceans and the marine life
- Hydrophone sensors have been deployed in three locations in the state of Washington
- The Orcasound community initiative is using them to study Orca whales in the Pacific Northwest region.

https://www.orcasound.net/

These applications need workflow technologies to orchestrate computations edge-to-cloud edge-to-HPC

* Work in progress

Processing instrument data in real time



Hidden workflows

Explicit workflow construction

USC Viterbi School of Engineering Information Sciences Institute

Pegasus Workflow Management System

Workflow Challenges Across Domains

Describe complex workflows in a simple way

Access distributed, heterogeneous data and resources (heterogeneous interfaces)

Deal with resources/software that change over time

Ease of use. Ability to debug and monitor 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.

1. Resource-independent Specification

Input Workflow Specification YAML formatted

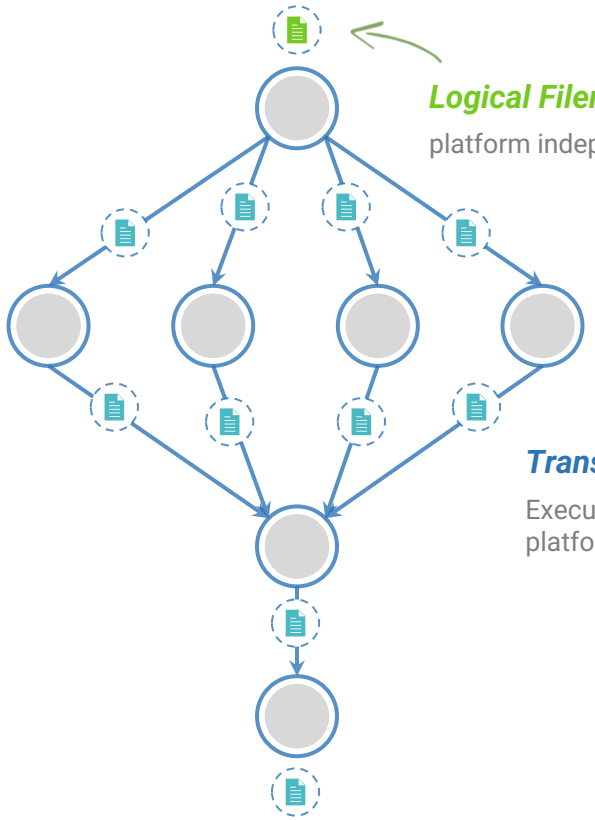
directed-acyclic graphs

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

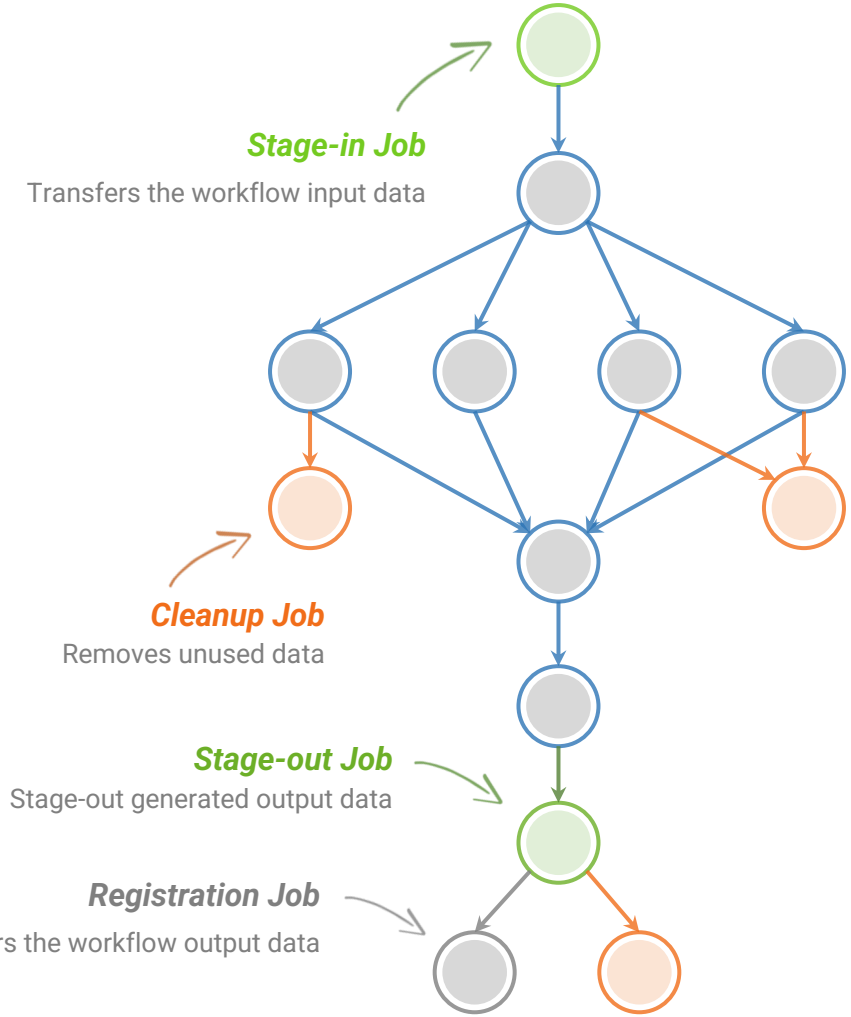


Logical Filename (LFN)

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

2. Submit locally, run globally

- **Pegasus WMS ==** **Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker**

  Pegasus maps workflows to target infrastructure (1 or more resources)

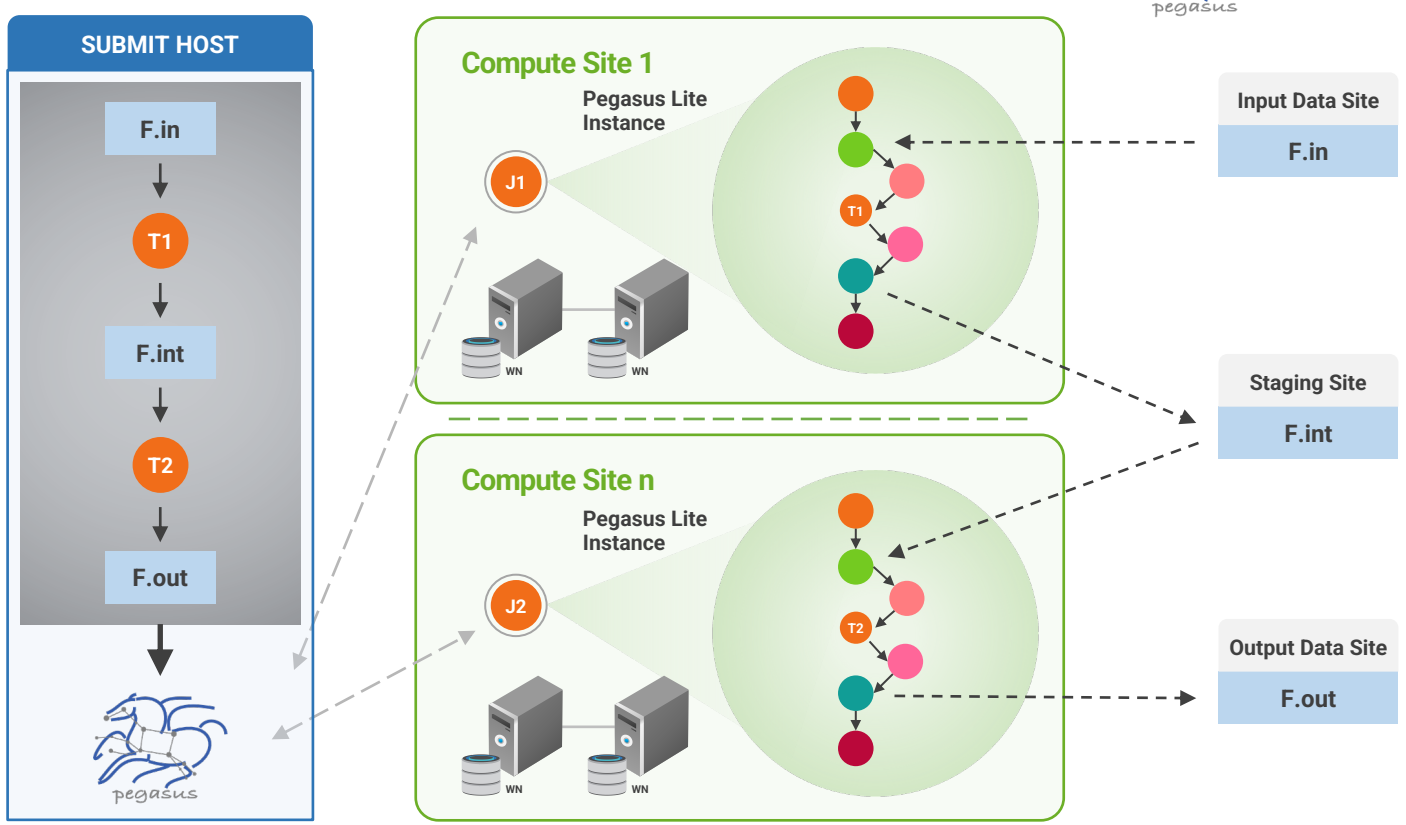
  DAGMan manages dependencies and reliability

  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**



3. Flexible Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, …)

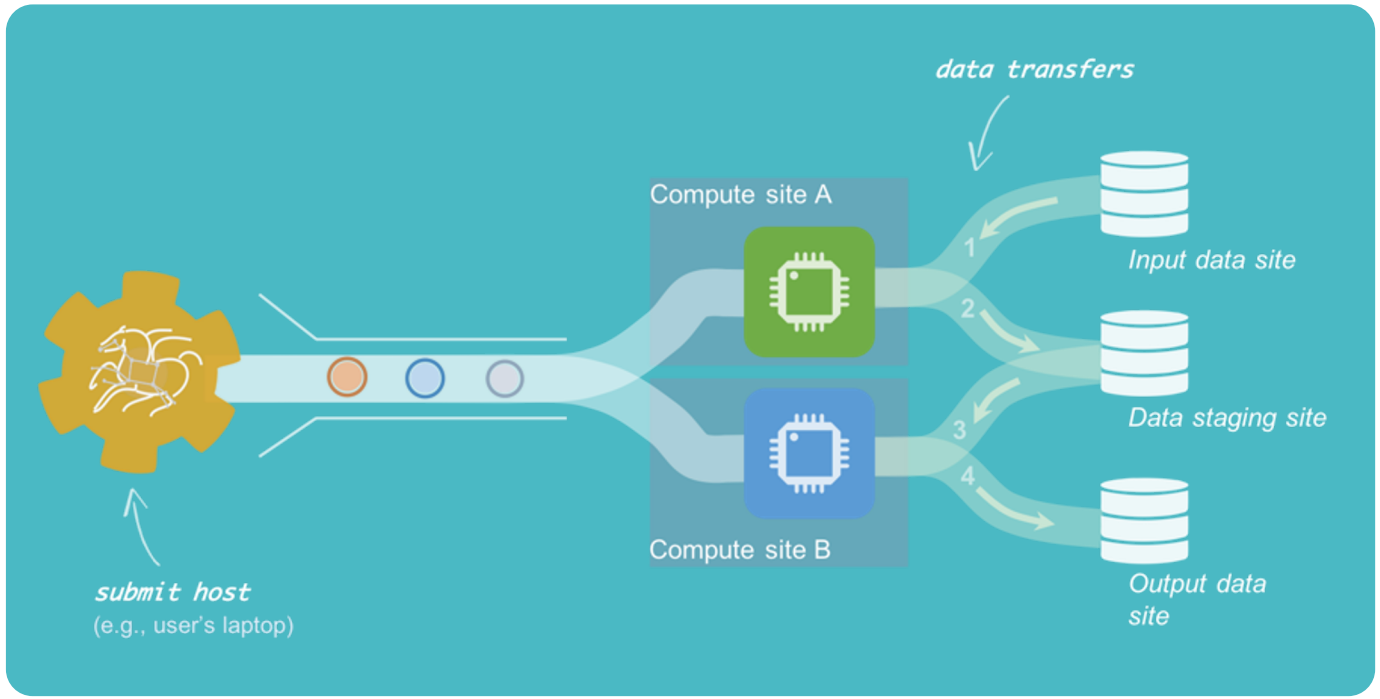
Worker nodes do not share a file system
Data is pulled from / pushed to the submit host via HTCondor file transfers
Staging site is the submit host

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

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

I/O is directly against the shared file system



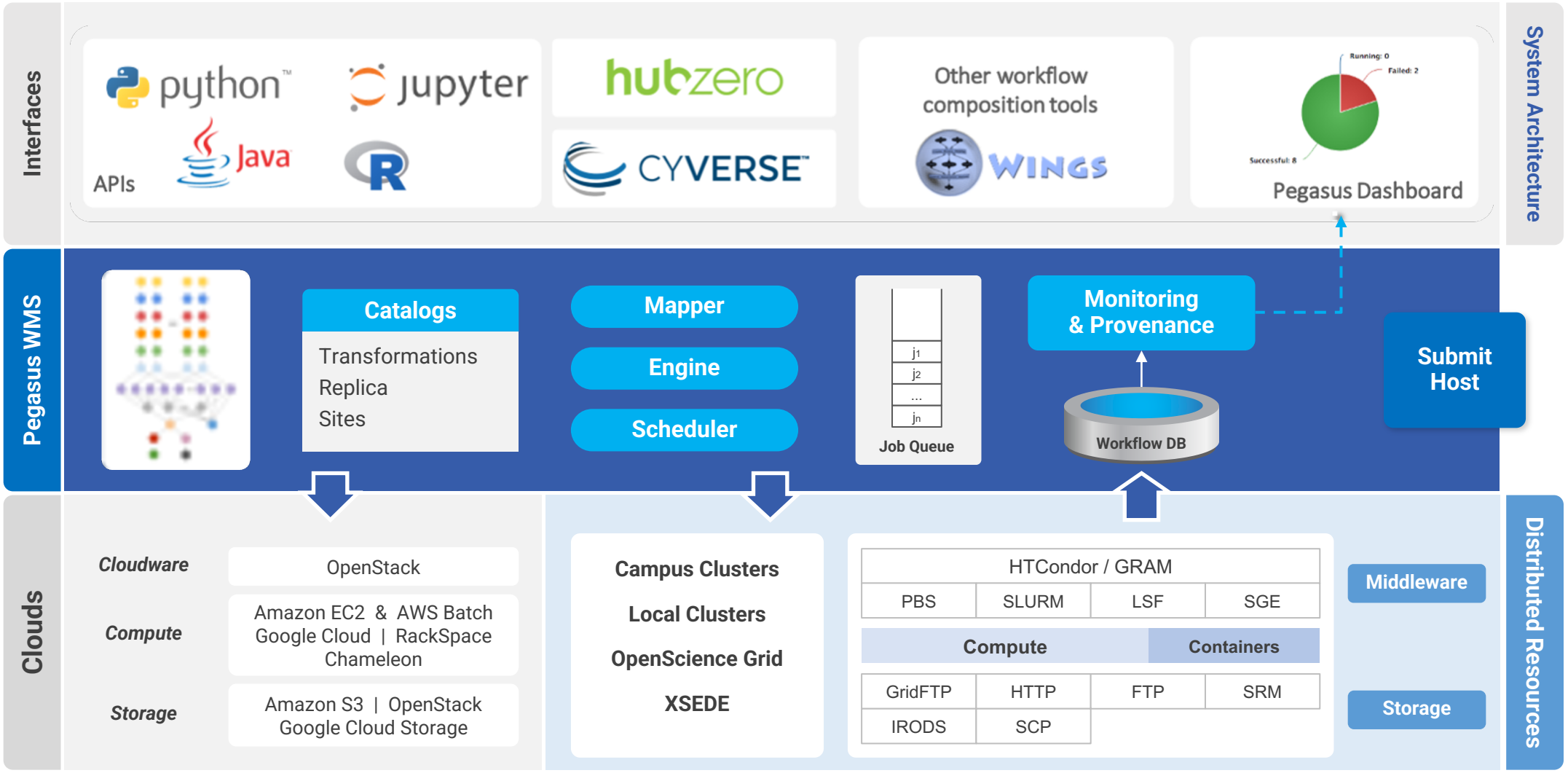
4. Flexible Data movement Pegasus-transfer

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

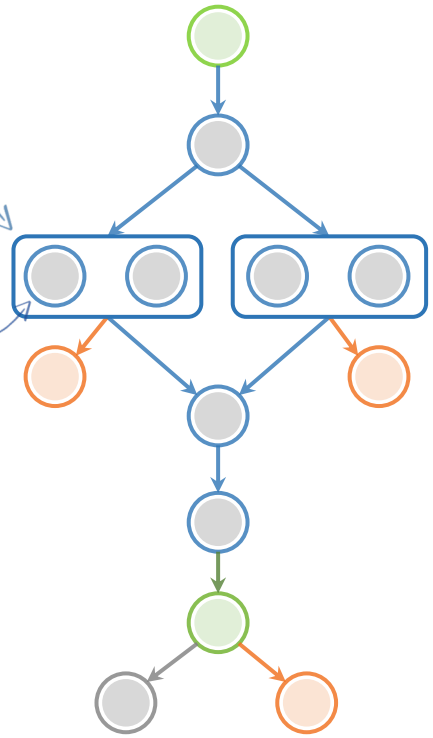
- **Directory creation, file removal**
  If protocol can support it, also used for cleanup
- **Two stage transfers**
  e.g., GridFTP to S3 = GridFTP to local file, local file to S3
- **Parallel transfers**
- **Automatic retries**
- **Credential management**
  Uses the appropriate credential for each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

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



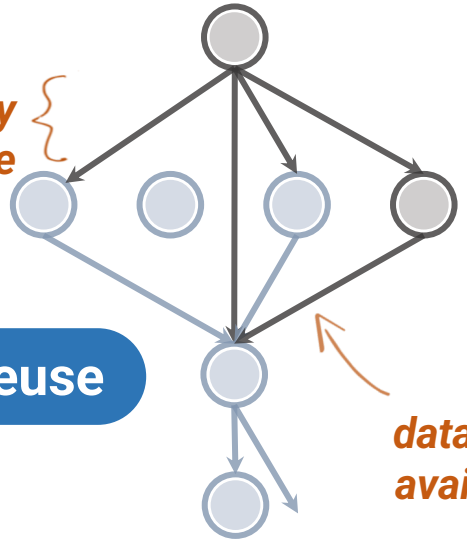
Optimizing for performance



Clustered Job
Groups small jobs together to improve performance

Task
Small granularity

Task Clustering

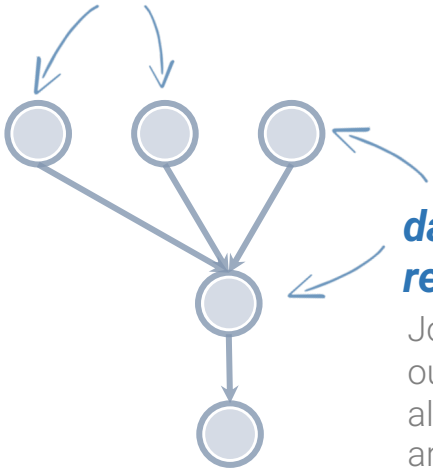


data already available

Data reuse

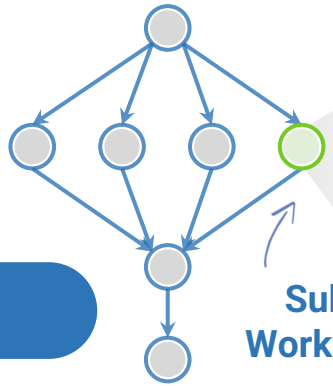
data also available

workflow reduction



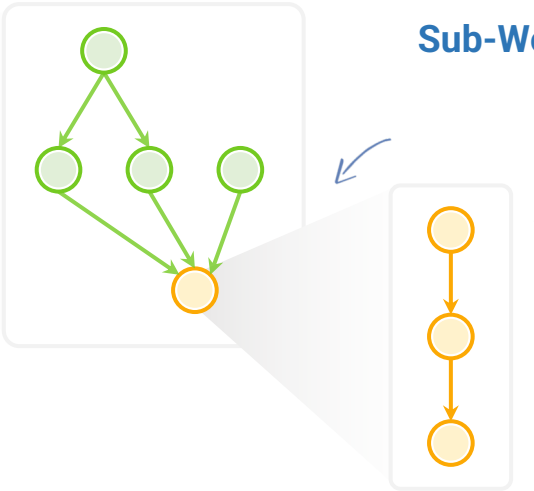
data reuse

data reuse
Jobs which output data is already available are pruned from the DAG



Scale

Sub-Workflow



Sub-Workflow

Recursion ends When abstract workflow with only compute jobs is encountered

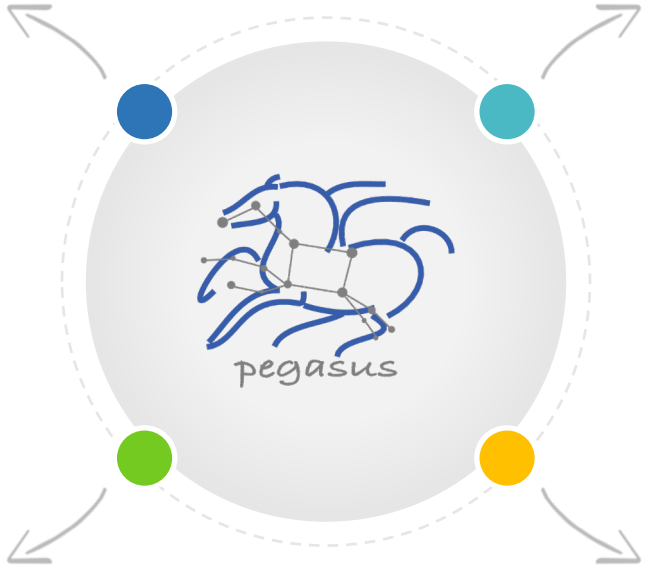
Handling failures

Postscript

detects non-zero exit code output parsing for success or failure message exceeded timeout do not produced expected output files

Job Retry

helps with transient failures set number of retries per job and run



Checkpoint Files

job generates checkpoint files staging of checkpoint files is automatic on restarts

Rescue DAGs

workflow can be restarted from checkpoint file recover from failures with minimal loss

Challenges of Workflow Management on Heterogeneous Systems and what we can do better

Applications:

- Tasks can have different resource needs
- The needs may not be fully known ahead of the execution
- There may be a number of inter-related workflows running at the same time (workflow ensembles)

HPC Systems:

- Resource discovery is based on static information about general availability
- Limited dynamic information from the scheduler
- Limited resource provisioning capabilities, but there are some good examples
  - Can reserve burst buffers as part of the workflow
  - Can share burst buffers across workflows to support data reuse

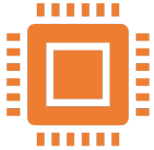
Main issue: who controls the resources?

- Traditional schedulers do fine-grained resource assignments, may not be scalable or desirable for extreme, highly heterogeneous systems
- In an alternative view WMS could request a set of resources from the scheduler and manage them on behalf of the workflow or workflow ensemble
  - Doing it indirectly today with pilot jobs

Resource control

Knowledge

WMS knows the current and potentially future workload needs at a coarse level so it can potentially make better decisions



Resource discovery for WMS scheduling

Number of available cores, CPUs, GPUs, memory

Available of various architectural features, such as burst buffers and their capacity, interfaces

Discovery of data location (if you pre-staged data into a BB or node-local disk for example)

Resource provisioning

Not just focused on compute resources

Could be asking for:

- Compute resources "close to data sets"
- The same compute resources as job *j-1* had

Questions?

What information to expose about the target system(s)?

How does this information need to be aggregated/what's the right level of abstraction?

Can we use ML techniques to learn to schedule better in the absence of complete information?

Job scheduling

Need task performance models that take into account the heterogeneous system features

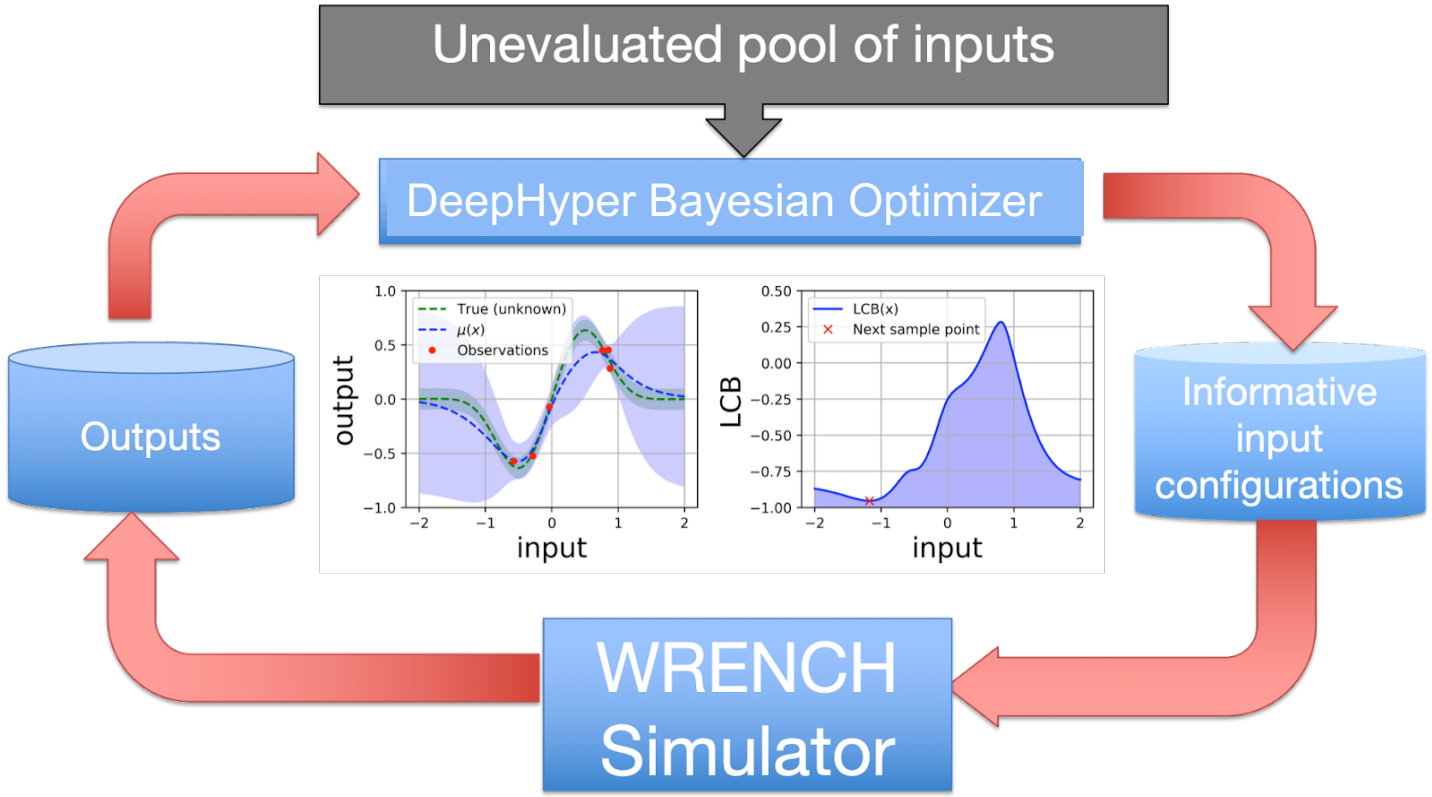
Mix of analytical models, simulation, machine learning

Need models that take into account data movement within the storage hierarchy: memory, burst buffers, file system, via LAN, via WAN

Focus on data-aware scheduling for an ensemble, workflow, job, within a job

Need more functionality from schedulers to support changing application demands over time (de-allocate/allocate)

PoSeiDon
Platform for Explainable Distributed Infrastructure



Task Performance Models

41 simulator parameters

Developing accurate simulators for performance prediction

Prasanna Balaprakash, ANL and Loïc Pottier, USC

U.S. DEPARTMENT OF ENERGY · USC Viterbi School of Engineering · renci · Argonne NATIONAL LABORATORY · BERKELEY LAB

Online Application and System Monitoring

Besides information needed for making resource provisioning and scheduling decisions, need information during execution to:

- Capture performance and faults
- Feed ML models that learn about performance and detect anomalies
- Any other information that can help WMS make decisions about releasing/acquiring resources
  - New resources available?

WMS needs to be self-aware, assure its own execution does not affect the workflows it managed

- Assuming the WMS or a specialized workflow engine is running in-situ along side the simulation and analytics

High Resolution Gantt Charts & CNNs

- **Ready time:** Timestamp since the beginning of the workflow, where all dependencies have been met and job can be dispatched.
- **Pre script delay:** Time spent on a script that is executed before job submission (if exists).
- **WMS delay:** Time spent by the workflow management system to prepare and submit the job.
- **Queue delay:** Time spent in the queue waiting for resources.
- **Stage in delay:** Time spent transferring input data.
- **Runtime:** Time spent during computation.
- **Stage out delay:** Time spent transferring data to the intermediate scratch directory or final output directory.
- **Post script delay:** Time spent on a script executed after job exits (e.g., wms parses stdout and exit code).
- **Completion time:** Timestamp marking job completion, since beginning of workflow.

Model	Acc.	Recall	Prec.	F-score	Time (s)
Without Data Augmentation					
AlexNet	**0.910**	**0.910**	**0.918**	**0.910**	222.75
VGG-16	**0.910**	**0.910**	**0.916**	**0.910**	283.42
ResNet-18	0.880	0.880	0.881	0.879	320.43

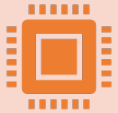
Experimental Results: Transfer Learning Approach

Need more sophistication from WMS to make scheduling and adaptation decisions

USC Viterbi School of Engineering Information Sciences Institute

Conclusions

Keep the separation between the workflow description and the executable workflow



Need new workflow technologies that are more versatile

- Explore various data communication modalities within workflows, taking full advantage of system architectural features. (memory, burst buffer, file system)
- Develop better resource provisioning and job scheduling capabilities

There are opportunities to define new schedulers and new scheduler/WMS interactions

- Need more information and control flow
- Need abstractions for information flow
- Need to combine resource provisioning and job scheduling

Layering is important: relationship to runtime systems