

# CI/CS WORKSHOP

THE COMMUNITY TOGETHER



Research**soc**



CI CoE PILOT

# End to End Workflow Monitoring and Execution

Ryan Tanaka  
Programmer Analyst  
USC Information Sciences Institute

# The Pegasus Workflow Management System

---

- Bridges the scientific domain and execution environment by **mapping high level workflow descriptions onto distributed resources**
- Enables scientists to:
  - **Automate** their work, as portable workflows
  - **Recover** from failures at runtime
  - **Debug** failures in their computations
- Built on top of HTCondor, a proven DHTC workhorse



Pegasus

---

**HTCondor**  
High Throughput Computing

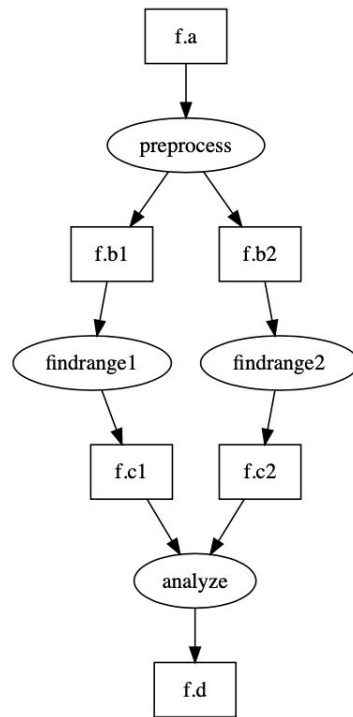
# Outline

---

- ~~Introducing the Pegasus WMS~~
- **Concepts**
- Features
- Production Use

# Workflows as DAGs

- Workflows are multi-step computational tasks organized as **directed acyclic graphs (DAG)**
- Define **abstract workflow** using one of our Python, Java, or R APIs
  - Abstract in the sense that users need not map jobs to resources or create file transfer jobs for input and output files
  - Pegasus will plan the abstract workflow into an executable workflow
- Example..



# Defining Workflow Inputs

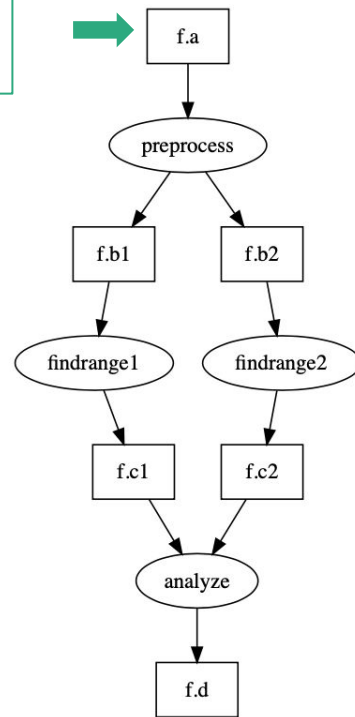
```
fa = File("f.a").add_metadata(creator="ryan")
rc = ReplicaCatalog().add_replica("local", fa, Path(".").resolve() / "f.a")
```

```
preprocess = Transformation(
    "preprocess",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)
```

```
findrange = Transformation(
    "findrange",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)
```

```
analyze = Transformation(
    "analyze",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)
```

```
tc = TransformationCatalog().add_transformations(preprocess, findrange, analyze)
```



# Defining Workflow Executables

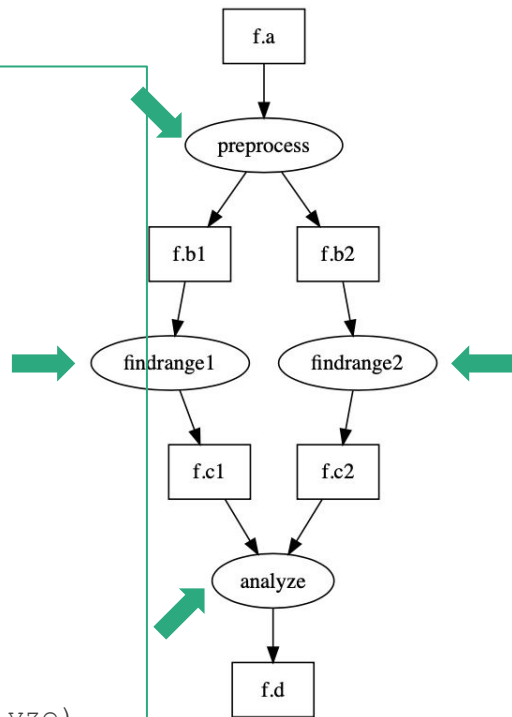
```
fa = File("f.a").add_metadata(creator="ryan")
rc = ReplicaCatalog().add_replica("local", fa, Path(".").resolve() / "f.a")

preprocess = Transformation(
    "preprocess",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)

findrange = Transformation(
    "findrange",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)

analyze = Transformation(
    "analyze",
    site="condorpool",
    pfn="/usr/bin/pegasus-keg",
)

tc = TransformationCatalog().add_transformations(preprocess, findrange, analyze)
```



# Defining the Workflow

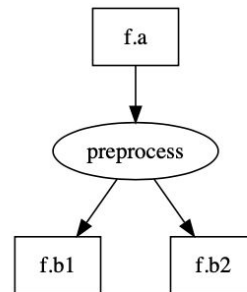
---

```
wf = Workflow("blackdiamond")
```

# Defining the Workflow

```
wf = Workflow("blackdiamond")

fb1 = File("f.b1")
fb2 = File("f.b2")
job_preprocess = Job(preprocess)\
    .add_args("-a", "preprocess", "-T", "3", "-i", fa, "-o", fb1, fb2)\
    .add_inputs(fa)\
    .add_outputs(fb1, fb2)
```

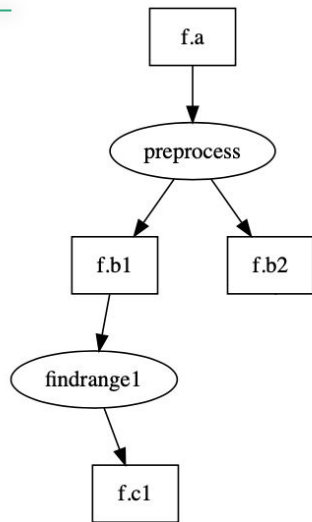


# Defining the Workflow

```
wf = Workflow("blackdiamond")

fb1 = File("f.b1")
fb2 = File("f.b2")
job_preprocess = Job(preprocess)\
    .add_args("-a", "preprocess", "-T", "3", "-i", fa, "-o", fb1, fb2)\
    .add_inputs(fa)\
    .add_outputs(fb1, fb2)

fc1 = File("f.c1")
job_findrange_1 = Job(findrange)\
    .add_args("-a", "findrange", "-T", "3", "-i", fb1, "-o", fc1)\
    .add_inputs(fb1)\
    .add_outputs(fc1)
```



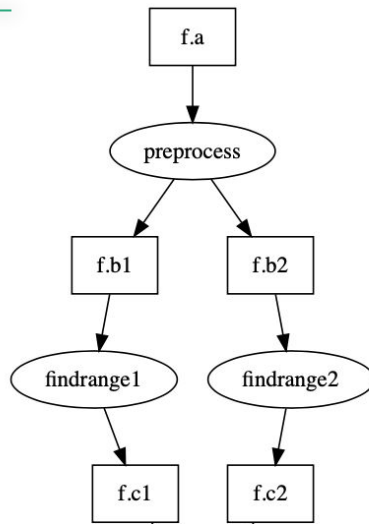
# Defining the Workflow

```
wf = Workflow("blackdiamond")

fb1 = File("f.b1")
fb2 = File("f.b2")
job_preprocess = Job(preprocess)\
    .add_args("-a", "preprocess", "-T", "3", "-i", fa, "-o", fb1, fb2)\
    .add_inputs(fa)\
    .add_outputs(fb1, fb2)

fc1 = File("f.c1")
job_findrange_1 = Job(findrange)\
    .add_args("-a", "findrange", "-T", "3", "-i", fb1, "-o", fc1)\
    .add_inputs(fb1)\
    .add_outputs(fc1)

fc2 = File("f.c2")
job_findrange_2 = Job(findrange)\
    .add_args("-a", "findrange", "-T", "3", "-i", fb2, "-o", fc2)\
    .add_inputs(fb2)\
    .add_outputs(fc2)
```



# Defining the Workflow

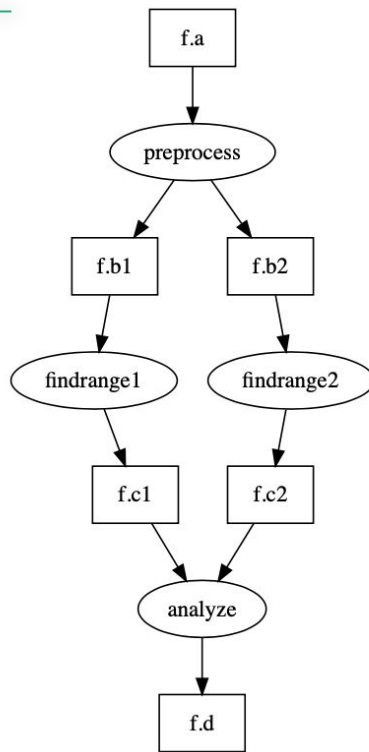
```
wf = Workflow("blackdiamond")

fb1 = File("f.b1")
fb2 = File("f.b2")
job_preprocess = Job(preprocess)\
    .add_args("-a", "preprocess", "-T", "3", "-i", fa, "-o", fb1, fb2)\
    .add_inputs(fa)\
    .add_outputs(fb1, fb2)

fc1 = File("f.c1")
job_findrange_1 = Job(findrange)\
    .add_args("-a", "findrange", "-T", "3", "-i", fb1, "-o", fc1)\
    .add_inputs(fb1)\
    .add_outputs(fc1)

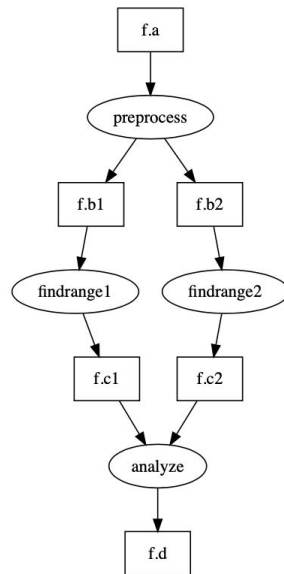
fc2 = File("f.c2")
job_findrange_2 = Job(findrange)\
    .add_args("-a", "findrange", "-T", "3", "-i", fb2, "-o", fc2)\
    .add_inputs(fb2)\
    .add_outputs(fc2)

fd = File("f.d")
job_analyze = Job(analyze)\
    .add_args("-a", "analyze", "-T", "3", "-i", fc1, fc2, "-o", fd)\
    .add_inputs(fc1, fc2)\
    .add_outputs(fd)
```



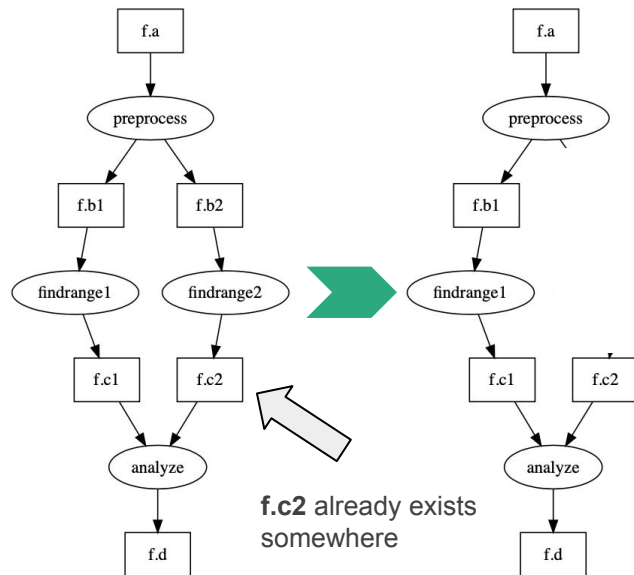
# Workflow Planning

- Workflow planning process
  - Data reuse module will optionally prune jobs for which output files already exist
  - Task clustering optimizations may be performed for small independent jobs
  - Mapping jobs onto physical compute resources
  - Add auxiliary jobs for data staging, cleanup, file registration, etc.
- Generated executable workflow submitted to through HTCondor to be run



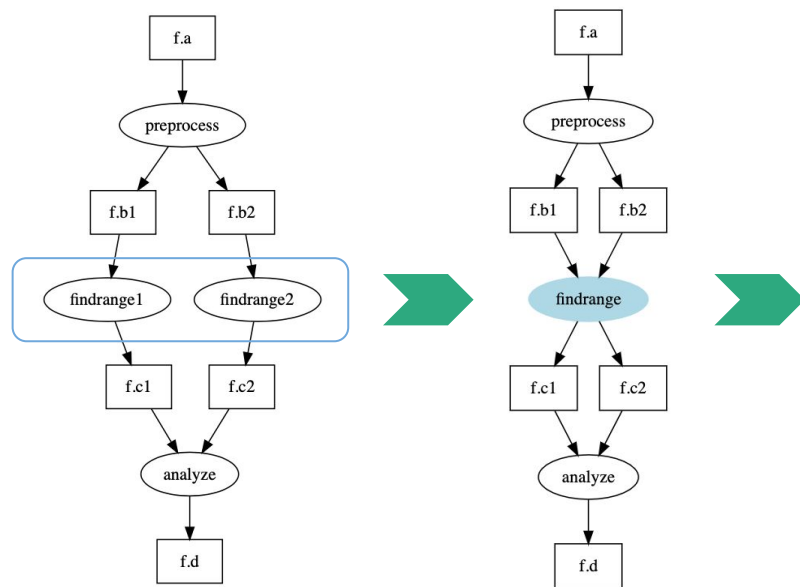
# Workflow Planning

- Workflow planning process
  - Data reuse module will optionally prune jobs for which output files already exist
  - Task clustering optimizations may be performed for small independent jobs
  - Mapping jobs onto physical compute resources
  - Add auxiliary jobs for data staging, cleanup, file registration, etc.
- Generated executable workflow submitted to through HTCondor to be run



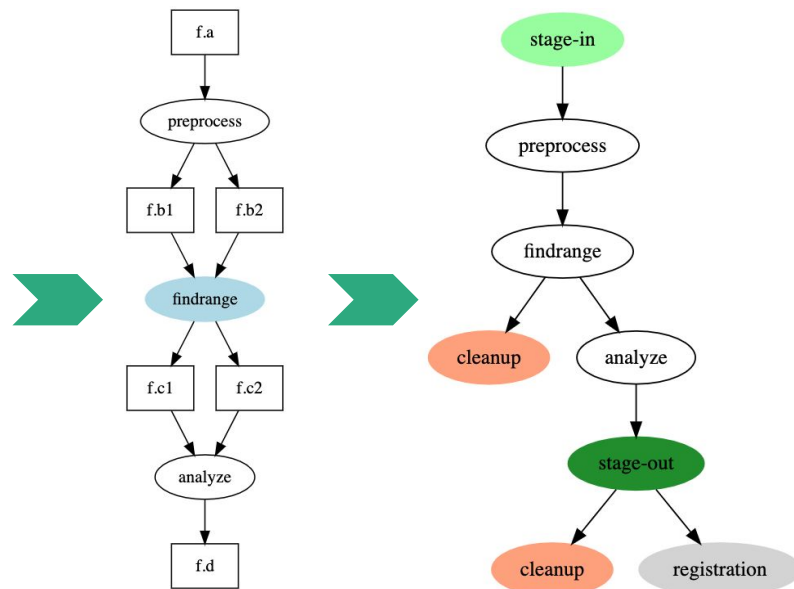
# Workflow Planning

- Workflow planning process
  - Data reuse module will optionally prune jobs for which output files already exist
  - Task clustering optimizations may be performed for small independent jobs
  - Mapping jobs onto physical compute resources
  - Add auxiliary jobs for data staging, cleanup, file registration, etc.
- Generated executable workflow submitted to through HTCondor to be run



# Workflow Planning

- Workflow planning process
  - Data reuse module will optionally prune jobs for which output files already exist
  - Task clustering optimizations may be performed for small independent jobs
  - Mapping jobs onto physical compute resources
  - Add auxiliary jobs for data staging, cleanup, file registration, etc.
- Generated executable workflow submitted to through HTCondor to be run



# Outline

- ~~Introducing the Pegasus WMS~~
- ~~Concepts~~
- **Features**
- Production Use

# 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
  - Submit host used as staging site
- **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 directly against the shared file system

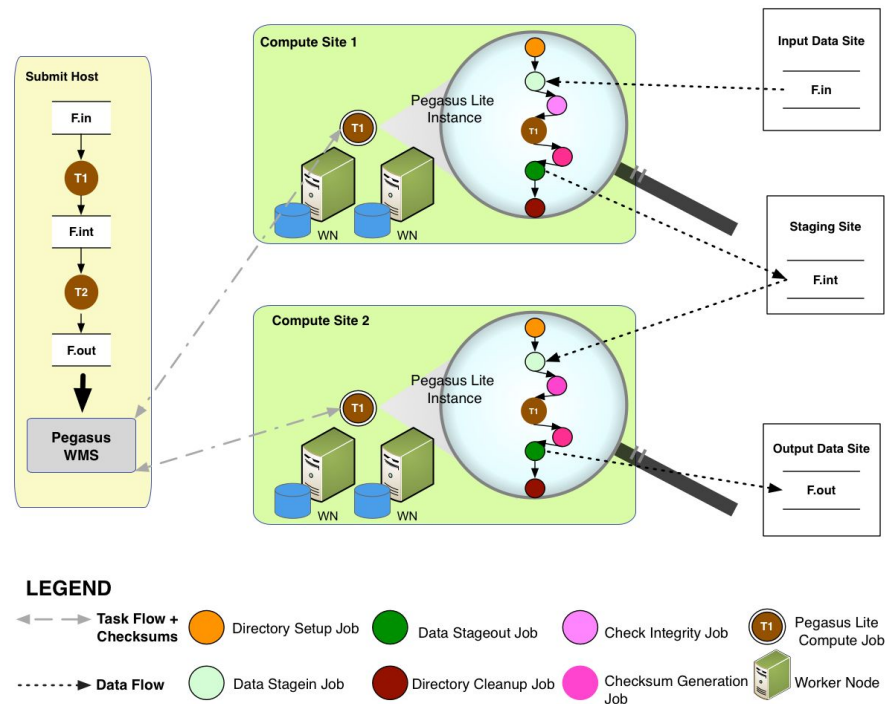
# pegasus-transfer internal file transfer utility

- Directory creation, file removal
- Two stage transfers between incompatible protocols
  - E.g., GridFTP to S3 is executed as: GridFTP to local file, local file to S3
- Parallel transfers
- Automatic retries
- Credential management

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

# Automatic Integrity Checking

- Pegasus automatically performs integrity checksums on input files right before jobs begin
  - Checksums can be specified for inputs coming from external sources
  - All intermediate and output files have checksums which are generated and tracked within the system
- Checksum validation failure results in job failure



# Monitoring and Debugging Tools CLI

- **pegasus-status**

- View current status of running workflow
- View summary of jobs and sub workflows

- **pegasus-analyzer**

- View errors from any failed jobs

- **pegasus-statistics**

- View summary of workflow statistics
- Succeeded jobs, failed jobs, retries, workflow  
walltime, etc.

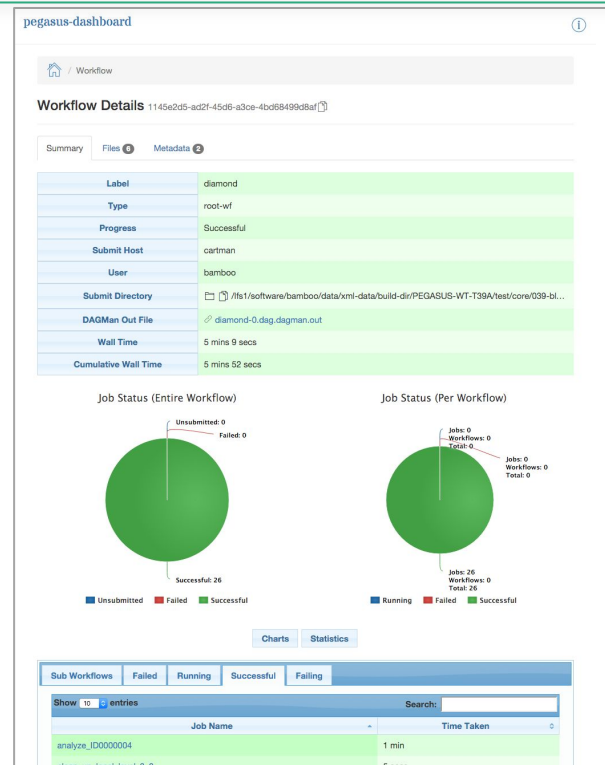
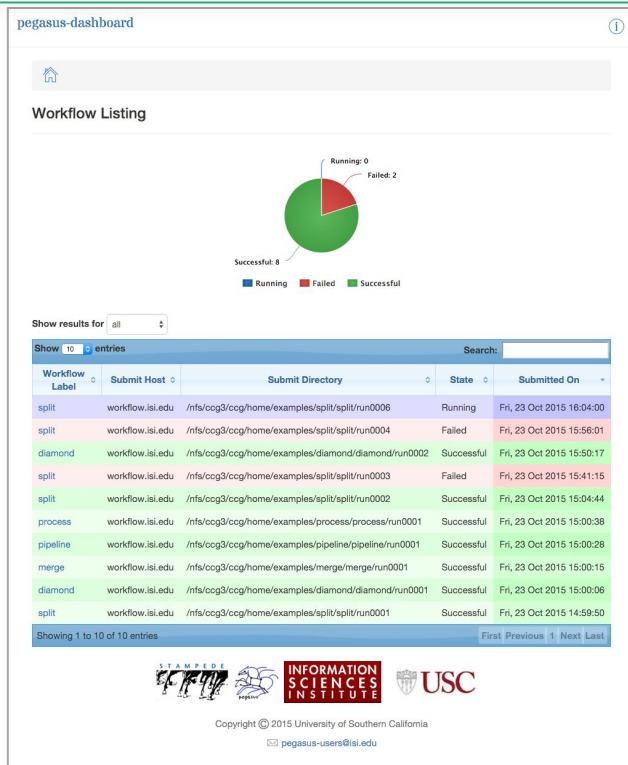
```
$ pegasus-status -l /Workflow/dags/directory
```

```
STAT  IN_STATE  JOB
Run    07:01   level-3-0
Run    06:25   |-sleep_ID000005
Run    06:20   \_subdax_level-2_ID000004
Run    05:44   |-sleep_ID000003
Run    05:39   \_subdax_level-1_ID000002
Run    05:03   \_sleep_ID000001
Summary: 6 Condor jobs total (R:6)
```

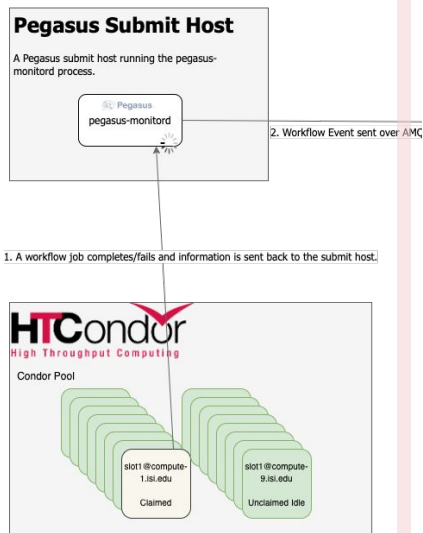
UNRDY	READY	PRE	IN_Q	POST	DONE	FAIL	%DONE	STATE	DAGNAME
0	0	0	1	0	1	0	50.0	Running	level-2_ID
0	0	0	2	0	1	0	33.3	Running	level-2_ID
0	0	0	3	0	1	0	25.0	Running	*level-3-0
0	0	0	6	0	3	0	33.3	TOTALS	(9

Summary: 3 DAGs total (Running:3)

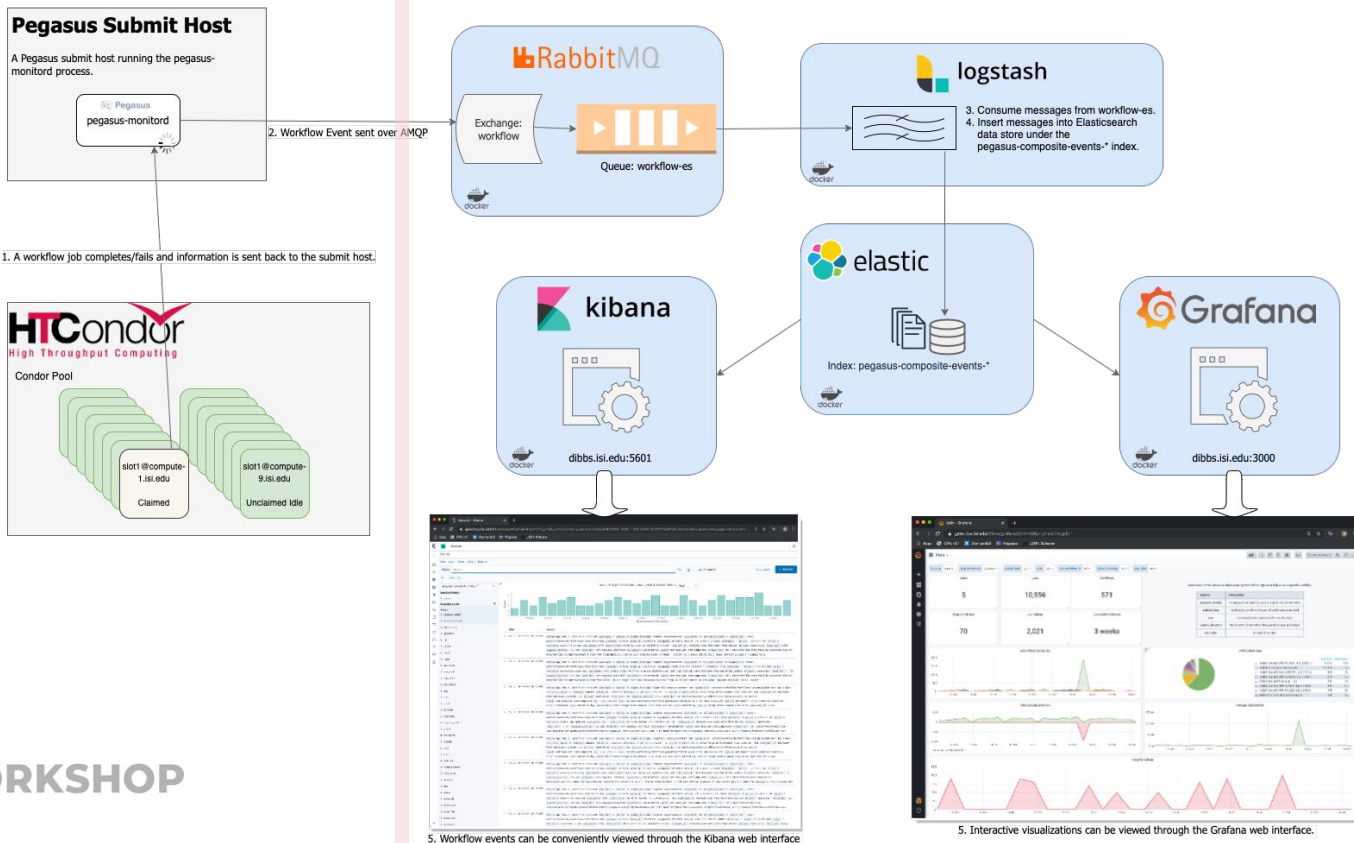
# Monitoring and Debugging Tools dashboard



# Monitoring and Debugging Tools AMQP endpoint

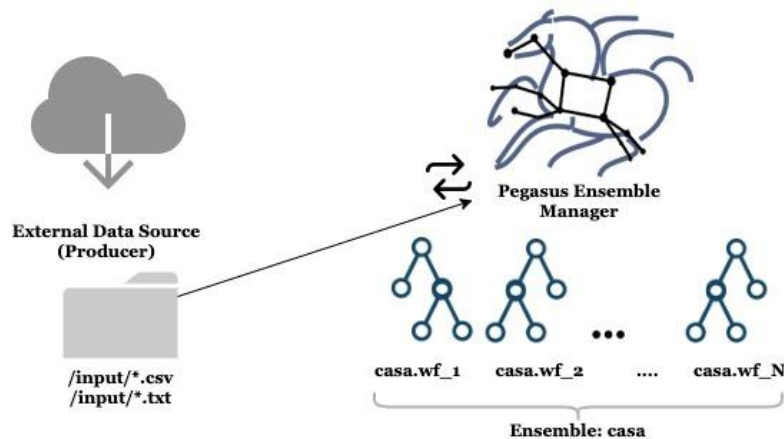


# Monitoring and Debugging Tools AMQP endpoint



# Ensemble Manager workflow management & dynamic triggering

- Service for managing collections of workflows called **ensembles**
- Allows for **throttling of concurrent planning and running workflows**
- Support for **triggering of new workflow runs based on arrival of new input files** which match one or more given patterns



# Outline

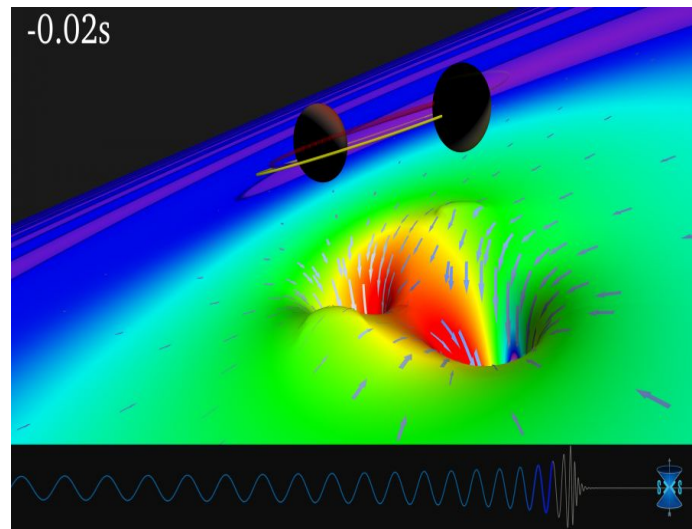
- ~~Introducing the Pegasus WMS~~
- ~~Concepts~~
- ~~Features~~
- **Production Use**

# LIGO PyCBC Workflows for Gravitational Wave Science

- Laser Interferometer Gravitational Wave Observatory

- Facility for gravitational wave research
- Methods:
  - PyCBC software package
  - Pegasus WMS workflows
  - Compute using OSG, XSEDE, etc.

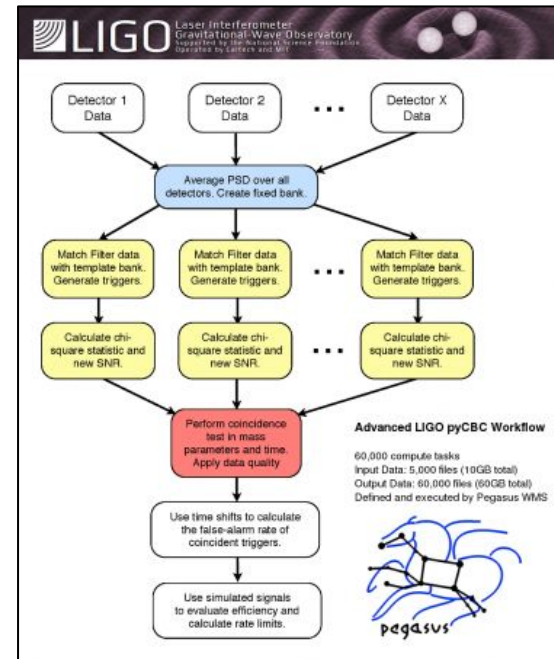
What do these workflows look like..



0.2 Second before the black holes collide.  
Image credit: SXS/LIGO

# LIGO PyCBC Workflows for Gravitational Wave Science

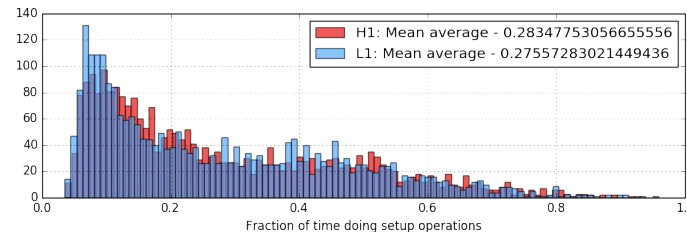
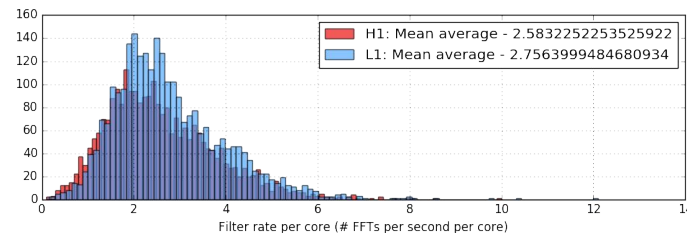
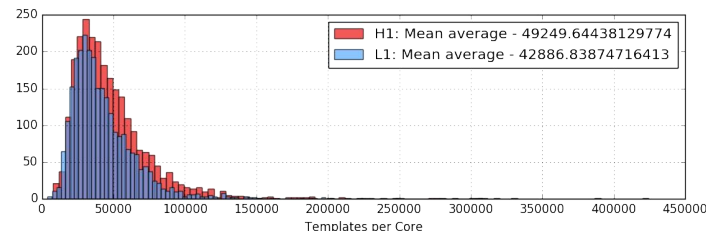
- Advanced PyCBC Workflows
  - 40,000 compute tasks
  - 1,100 input files
  - 63 output files
  - 725 GB processed data
  - Compute: LIGO Data Grid, OSG, EGI, XSEDE



Advanced LIGO pyCBC Workflow. Image Credit: Samantha Usman, Duncan Brown et al

# LIGO PyCBC Workflows for Gravitational Wave Science

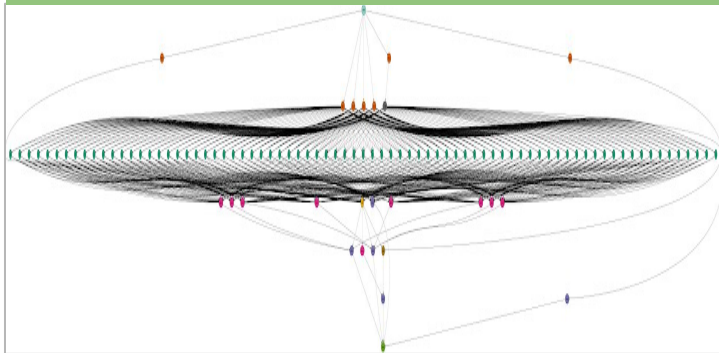
- Plots typically generated as part of post run analysis
- Using the AMQP data collection setup, these charts are able to be updated live as jobs complete, affording LIGO researchers better monitoring capabilities of the PyCBC workflow runs



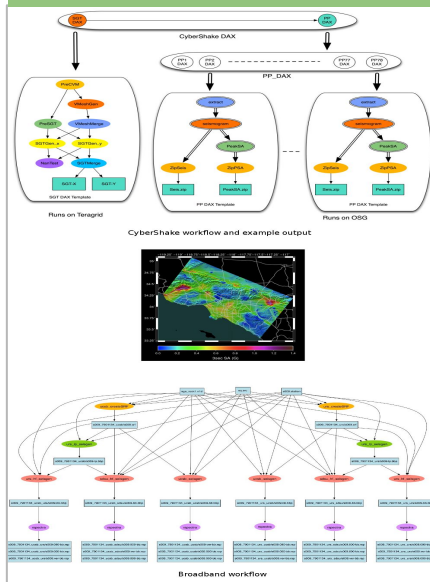
# Other Production Use

In the last 12 months, Pegasus users ran **240K workflows**, **145M jobs**

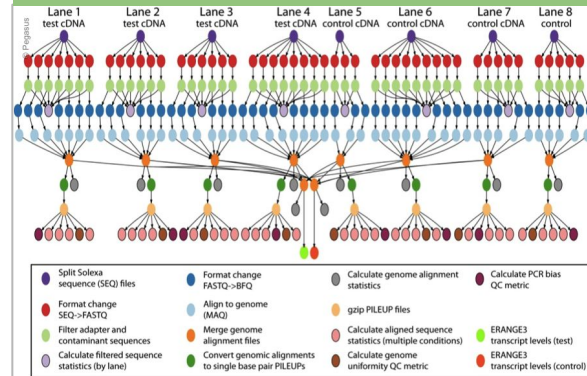
## XENONnT - Dark Matter Search



## SCEC CyberShake



## Epigenomics (USC)





# Pegasus est. 2001

Automate, recover, and debug scientific computations

## Get Started

---

### Pegasus Online Office Hours

<https://pegasus.isi.edu/blog/online-pegasus-office-hours>

Bi-monthly basis on the second Friday of the month, where we address user questions and also apprise the community of new developments.

### Pegasus Website

<https://pegasus.isi.edu/>

### Users Mailing List

[pegasus-users@isi.edu](mailto:pegasus-users@isi.edu)

### Pegasus Website

[pegasus-support@isi.edu](mailto:pegasus-support@isi.edu)

Pegasus 5.0 coming soon! Beta1 is out.

# Questions?

# Thank you!

tanaka@isi.edu