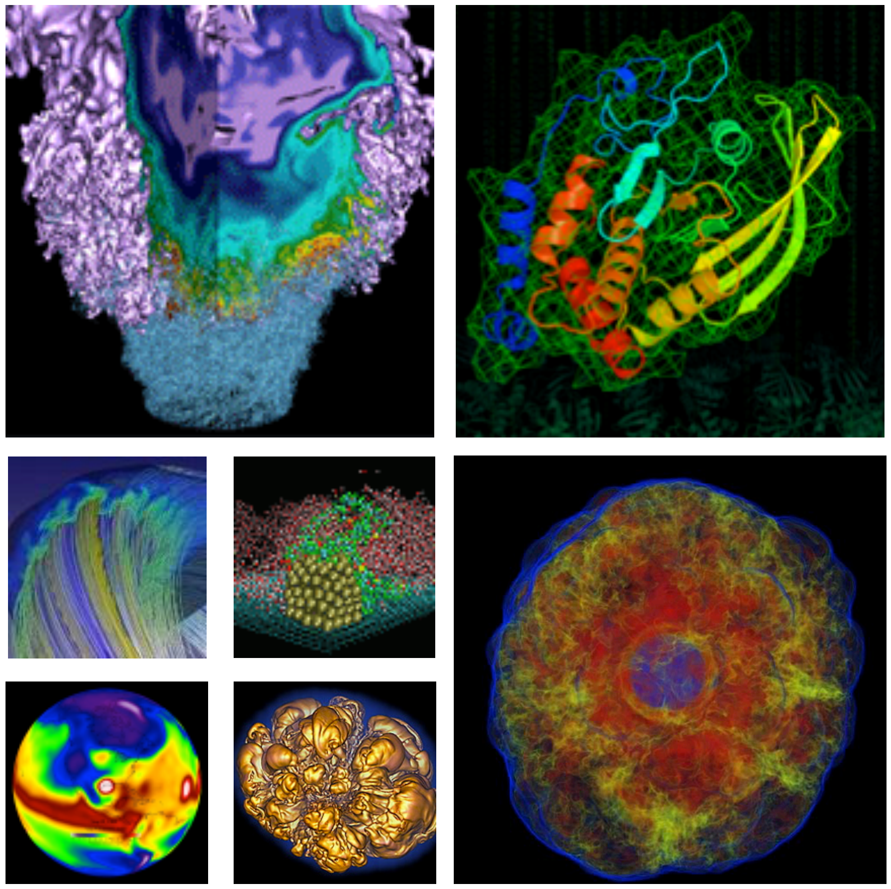
# Workflow Tools at NERSC



NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

**Shane Canon**
NERSC Data and Analytics Services

NERSC User Meeting
February 24, 2016

BERKELEY LAB
Lawrence Berkeley National Laboratory

## What Does Workflow Software Do?

- **Automate connection of applications**
  - Chain together different steps in a job pipeline.
  - Automate provenance tracking -> enable ability to reproduce results.
  - Assist with data movement.
  - Monitor running processes and handle errors.
  - Data processing of streaming experimental data (including near-realtime processing).
- **Workflows help work with (around?) batch scheduler and queue policies.**

## Workflows are Personal

- **Many tools exist in the workflow space**
  - Google: "Scientific Workflow Software"
- **It seems like each domain has its own workflow solution to handle domain-specific quirks**
- **No single tool solves every single problem**

| | |
|---|---|
| • **Fireworks** | • **BigPanda** |
| • **qdo** | • **Pegasus** |
| • **Tigres** | • **Taverna** |
| • **TaskFarmer** | • **Airavata** |
| • **Swift** | **…….** |

## Workflow Tools at NERSC

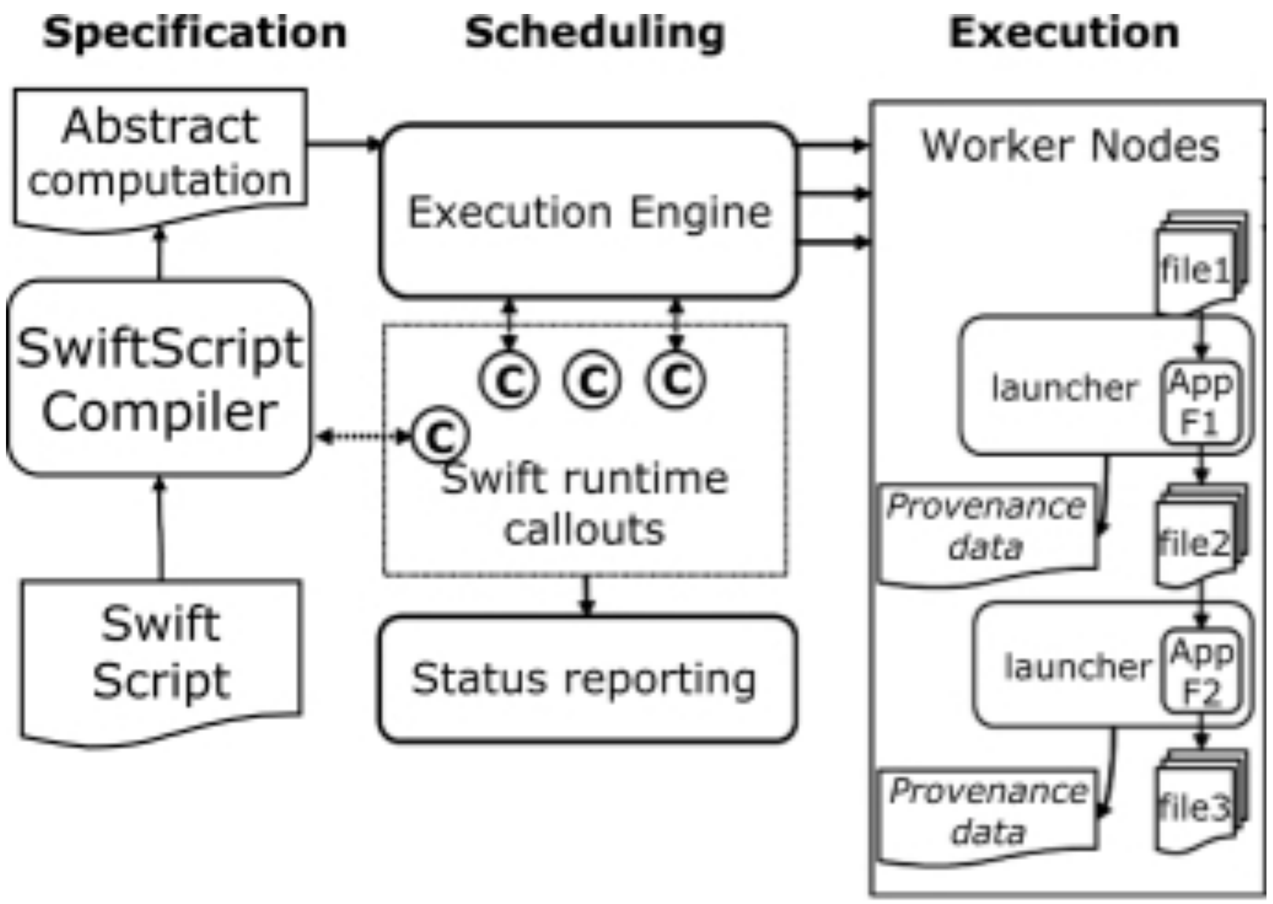
- **We support 3 workflow tools at NERSC**
  - **FireWorks**
  - **Swift**
  - **TaskFarmer**

  Supported: Installed, documented, and some staff expertise.
- **Create an ecosystem to enable self-supported of other Workflow tools**
  - Databases
  - User defined software modules
  - AMQP services (RabbitMq)

## Workflows and Data Intensive Science

- **Data intensive scientific computing may not always fit the traditional HPC paradigm**
  - Large numbers of tasks, low degree of parallelism.
  - Job dependencies and chaining.
  - Need to communicate with external datasources, DBs.
- **Workflow and work orchestration in this context can be thought of as sequences of compute and data-centric operations.**

## Visualising a workflow: swift



## Workflows as code: Swift

- **Swift is a workflow language (http://swift-lang.org)**

```
type file;

string curdir = java("java.lang.System","getProperty","user.dir");

app (file out, file err) mpi_hello (int time, int nproc)
{
    mpiwrap nproc mpiapp time stdout=@out stderr=@err;
}

int    nsim   = toInt(arg("nsim",    "10"));
int    time   = toInt(arg("time",    "1"));
int    nproc  = toInt(arg("nproc",   "56"));

global string mpiapp = arg("mpiapp", curdir+"/mpi_hello");

foreach i in [0:nsim-1] {
  file mpiout <single_file_mapper; file=strcat("output/mpi_",i,".out")>;
  file mpierr <single_file_mapper; file=strcat("output/mpi_",i,".err")>;
  (mpiout, mpierr) = mpi_hello(time, nproc);
}
```

## High Throughput "Bag of Tasks"

- **Often need to process large numbers of smallish tasks repeatedly.**
- **Typical queue policies work against you**
  - a lot of time lost waiting.
  - Batch system not set up for lots of little tasks.
- **Instead use a workflow system**
  - to queue up tasks.
  - to launch long running workers to consume these tasks.

- **Examples: taskfarmer, qdo and fireworks...**

## TaskFarmer Example

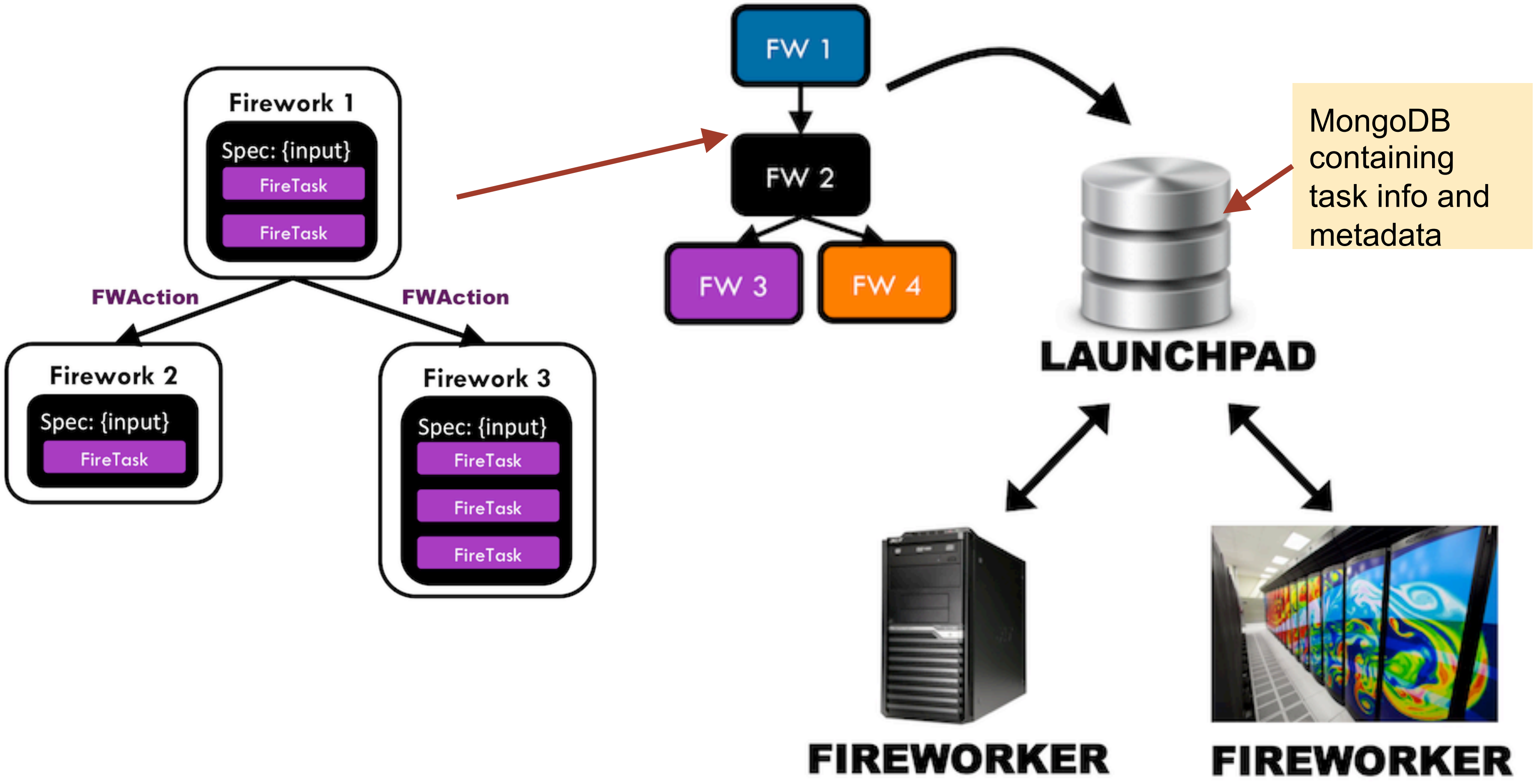
Define your tasks…

```
gzip $SCRATCH/file_1
gzip $SCRATCH/file_2
gzip $SCRATCH/file_3
gzip $SCRATCH/file_4
gzip $SCRATCH/file_5
gzip $SCRATCH/file_6
gzip $SCRATCH/file_7
gzip $SCRATCH/file_8
gzip $SCRATCH/file_9
gzip $SCRATCH/file_10
…
gzip $SCRATCH/file_200
```

Submit your taskfarmer job

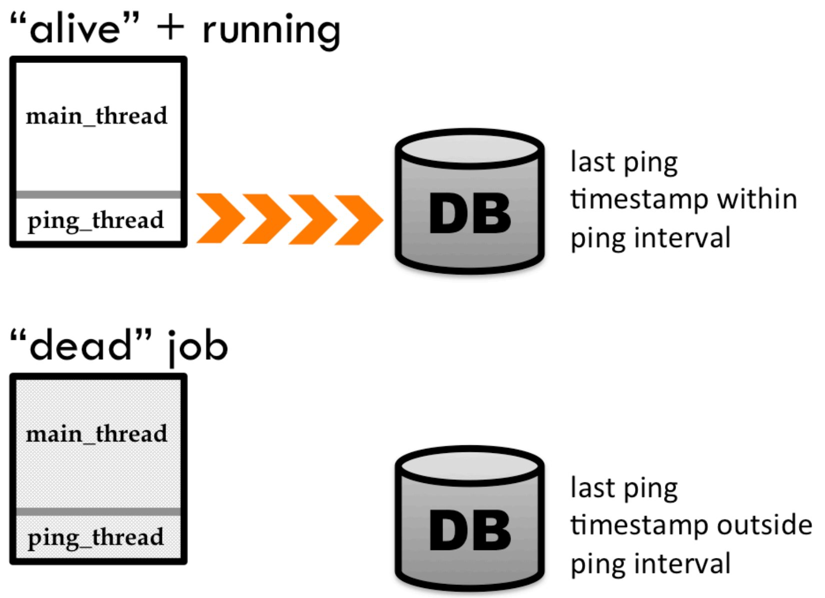
```
#!/bin/sh
#SBATCH -N 2 -c 64
#SBATCH -p debug
#SBATCH -t 00:05:00
#SBATCH -C haswell

module load taskfarmer
runcommands.sh tasks.txt
```

## Use case: Fireworks (material science)



## Fireworks: Error Handling and Dynamic Workflows

- **Can specify action based on soft failures, hard failures, human errors**
  - "lpad rerun –s FIZZLED"
  - "lpad detect_unreserved –rerun" OR
  - "lpad detect_lostruns –rerun" OR


"alive" + running
main_thread
ping_thread
DB
last ping timestamp within ping interval
"dead" job
main_thread
ping_thread
DB
last ping timestamp outside ping interval

I DON'T ALWAYS MAKE A CATACLYSMIC MISTAKE
BUT WHEN I DO, I JUST GO BACK IN TIME AND FIX IT
Signs of the Timms

## Batch Queues

- **NERSC has support for serial and high throughput queues that are well suited to jobs that need many task computing**
  - Cori Serial queue designed specifically for these use cases.
- **Reservations available for special needs.**
- **Consider using job packing options in various workflow tools to optimize for HPC queue infrastructure**
  - also for packing single-core jobs into a multi-core node.

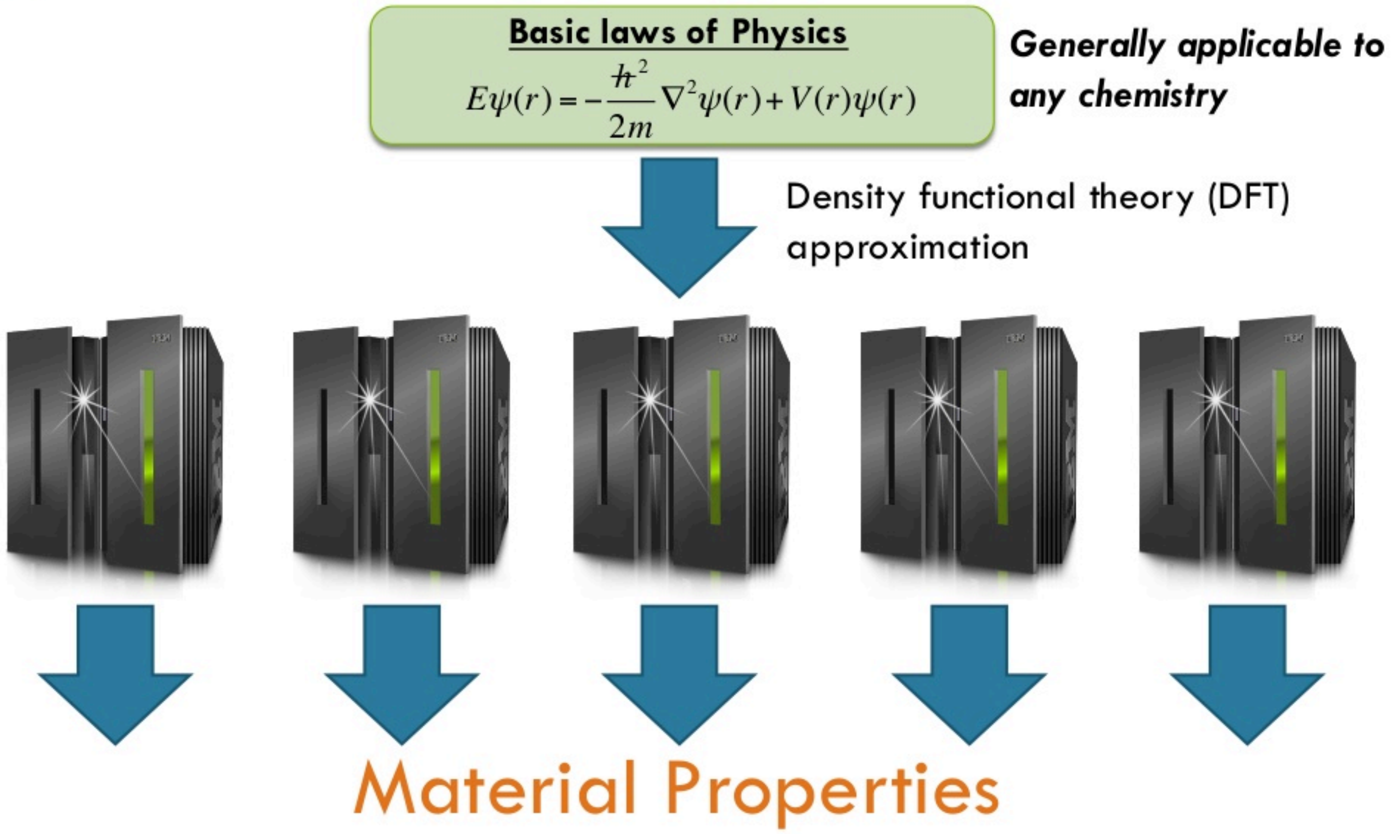
# Workflow Ecosystem @ NERSC

- **Science Gateways**
- **Databases**
  - Mongo, Postgres, MySQL, SQLite, SciDB
- **Workflow tools (self-supported)**
  - Fireworks, Swift, Tigres, qdo
- **High throughput batch queues**
- **NEWT REST API**
- **Globus / Data Transfer Nodes**
- **Task frameworks**
  - Taskfarmer
- **Other web based tools for interactive use cases**
  - iPython, R Studio, NX
- **MapReduce frameworks**
  - Spark

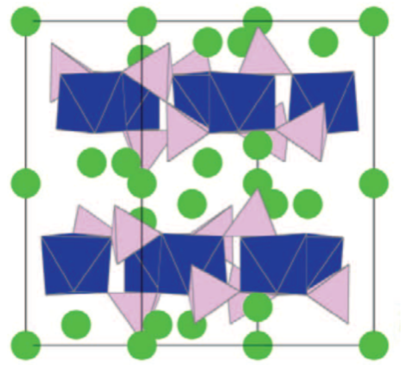
Workflow tools exist in and interact with a rich environment of NERSC capabilities and services.

## Use Case: Materials Project

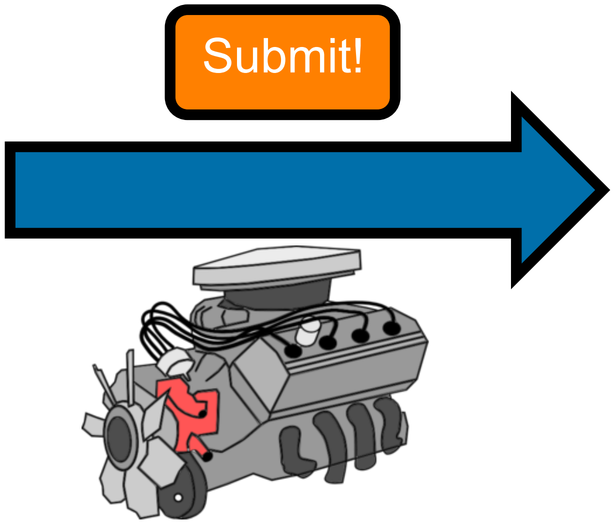
- **Simulate properties of all possible materials.**

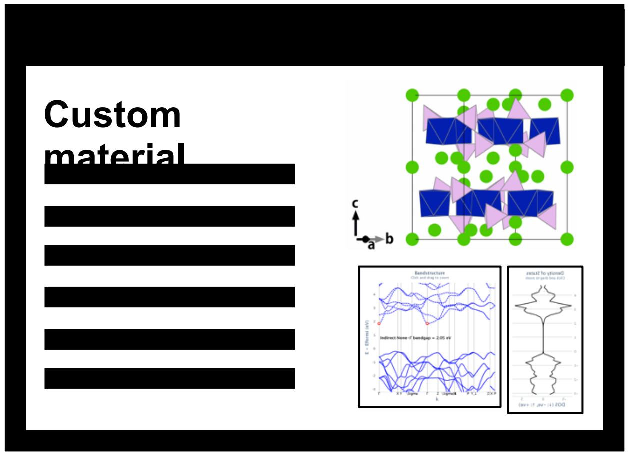

**Basic laws of Physics**
$E\psi(r) = -\frac{\hbar^2}{2m}\nabla^2\psi(r) + V(r)\psi(r)$
*Generally applicable to any chemistry*
Density functional theory (DFT) approximation
Material Properties

# Materials Project Workflow

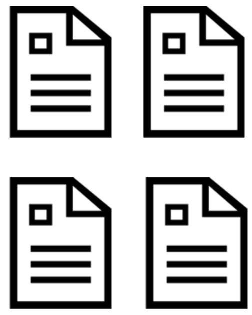


input: A cool material !!

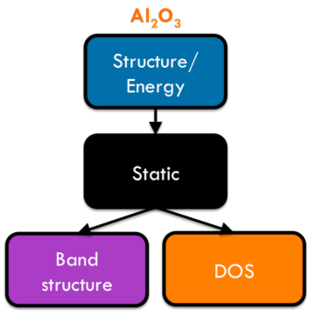




output: Lots of information about cool material !!

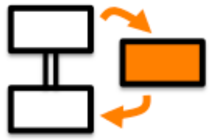


Input generation (parameter choice)

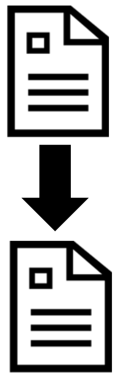


Workflow mapping

Supercomputer submission / monitoring



Error handling

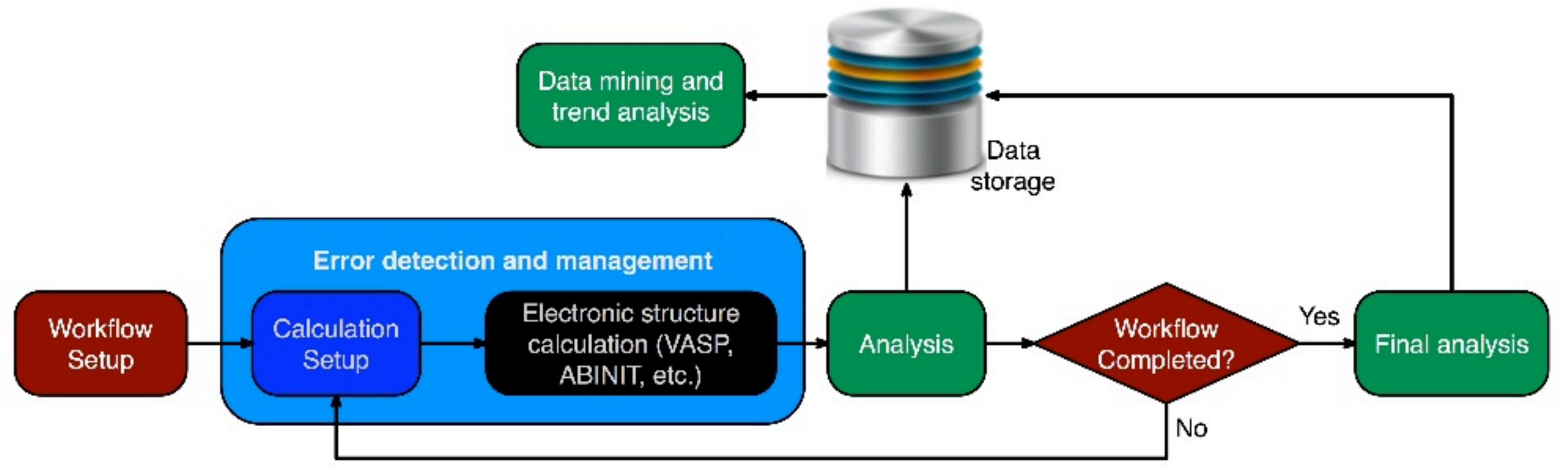


File Transfer

File Parsing / DB insertion

## Use Case: Materials Project

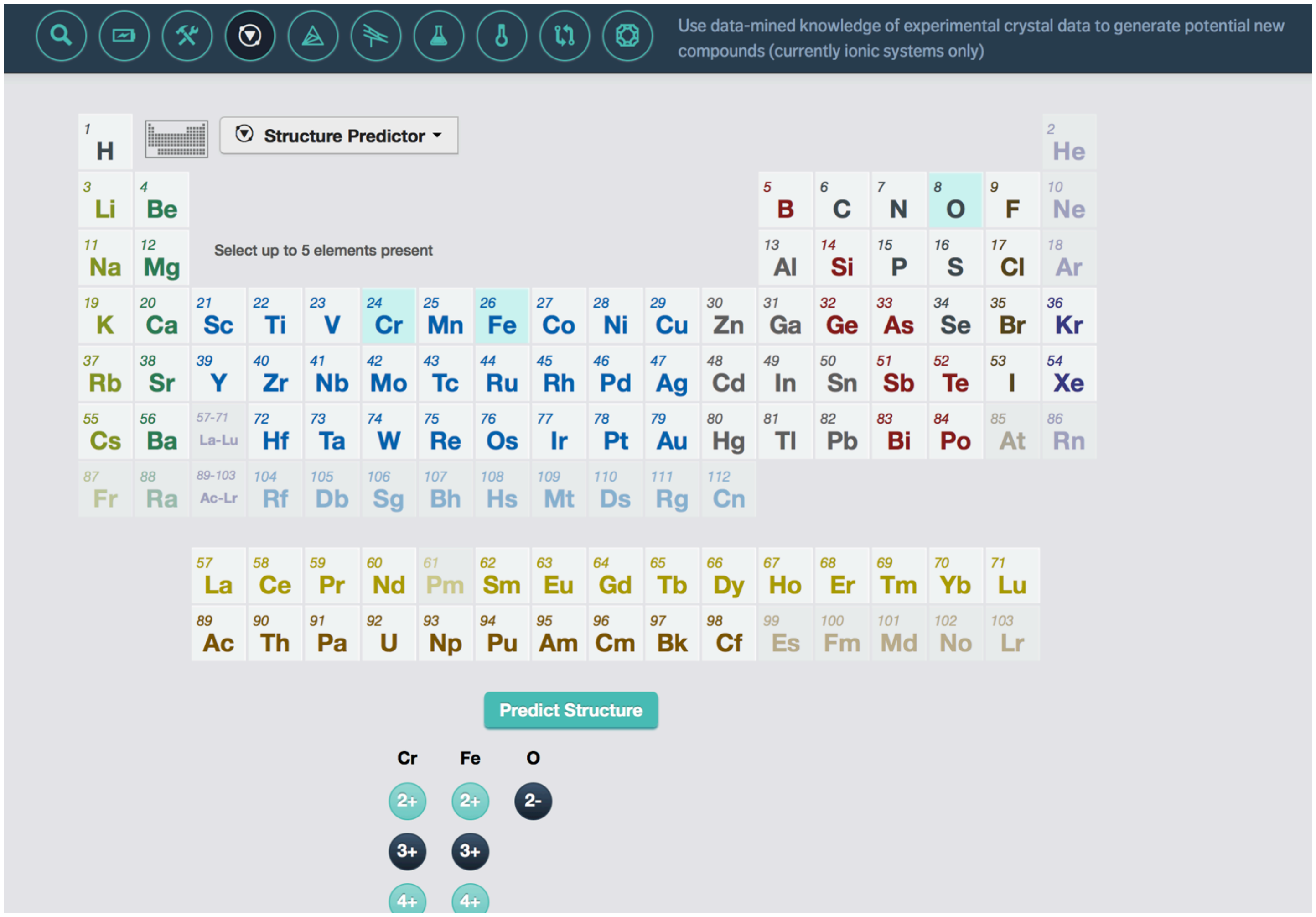
- **Simulate properties of all possible materials.**


Data mining and trend analysis
Data storage
Error detection and management
Workflow Setup
Calculation Setup
Electronic structure calculation (VASP, ABINIT, etc.)
Analysis
Workflow Completed?
Yes
No
Final analysis

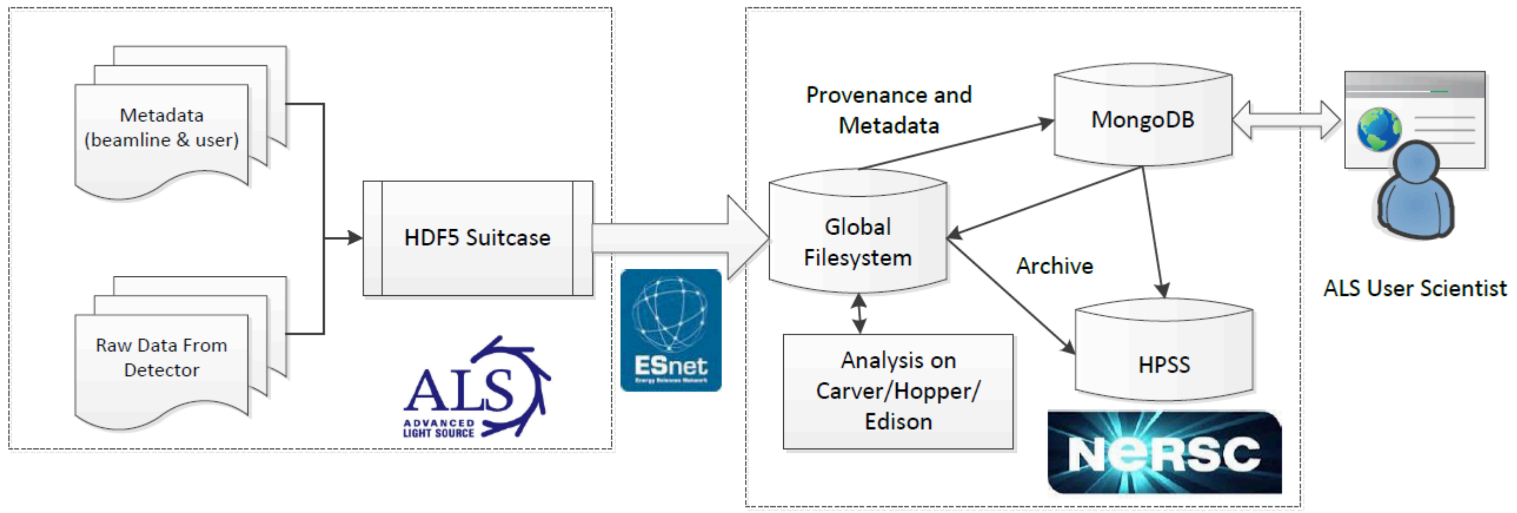
## Use Case: Materials Project

- **Tasks submitted to Fireworks MongoDB via API/ python script etc.**
- **MongoDB keeps a list of tasks to be run.**
- **Fireworks submits workers to NERSC queues.**
- **Workers pull jobs from MongoDB.**
- **Fireworks manages job orchestration**
  – Retry on failure
  – File transfer
  – Job Dependencies
  – Flow control for subsequent jobs
  – Duplicate management

# Materials Project Gateway

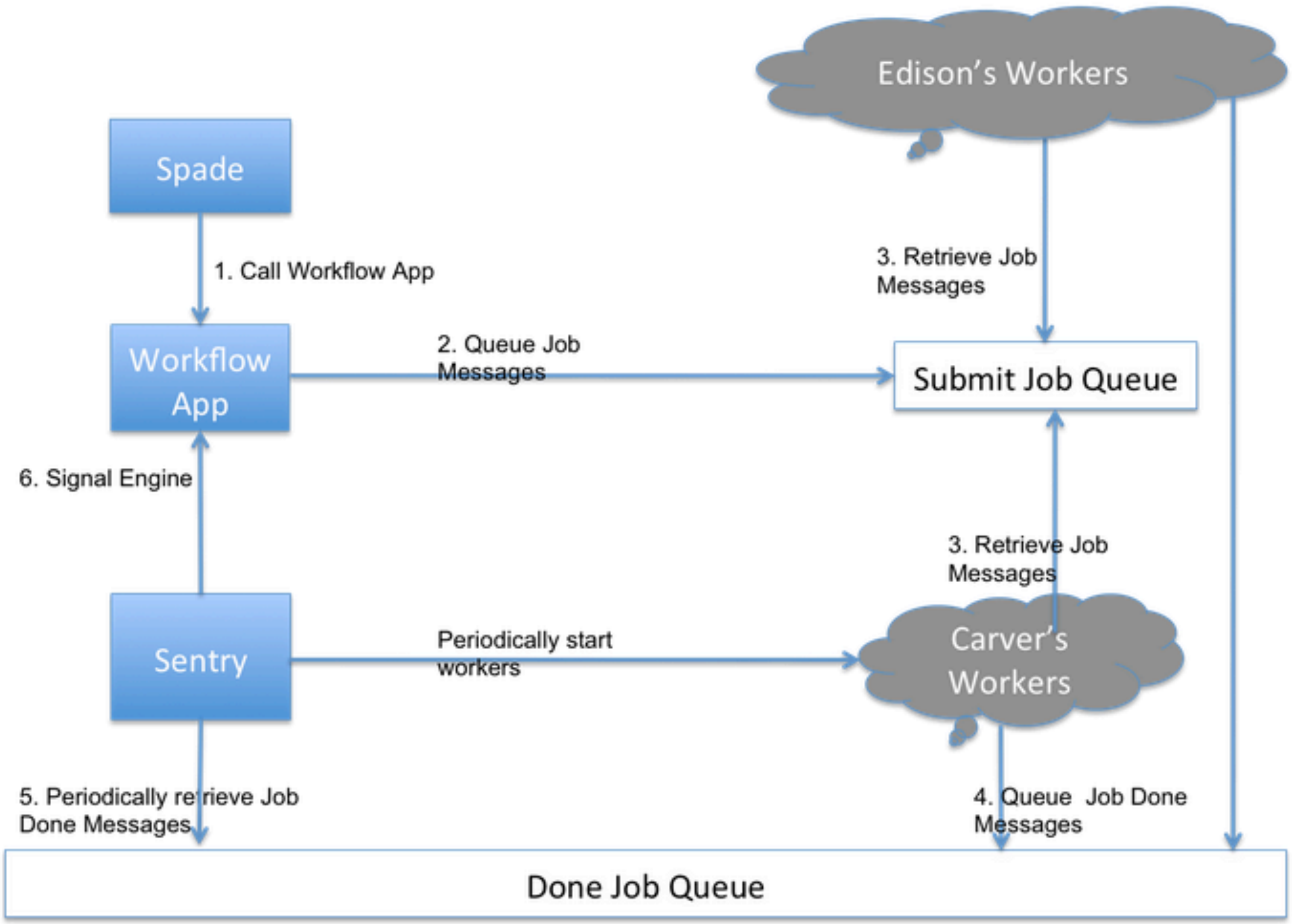

Use data-mined knowledge of experimental crystal data to generate potential new compounds (currently ionic systems only)

# Use Case: SPOT Suite



- Collect Data from Beamline
- SPADE/Globus to move data to NERSC
- Trigger Analysis at NERSC via AMQP
- View Jobs and Results on Science Gateway
- Track Provenance and Metadata via MongoDB

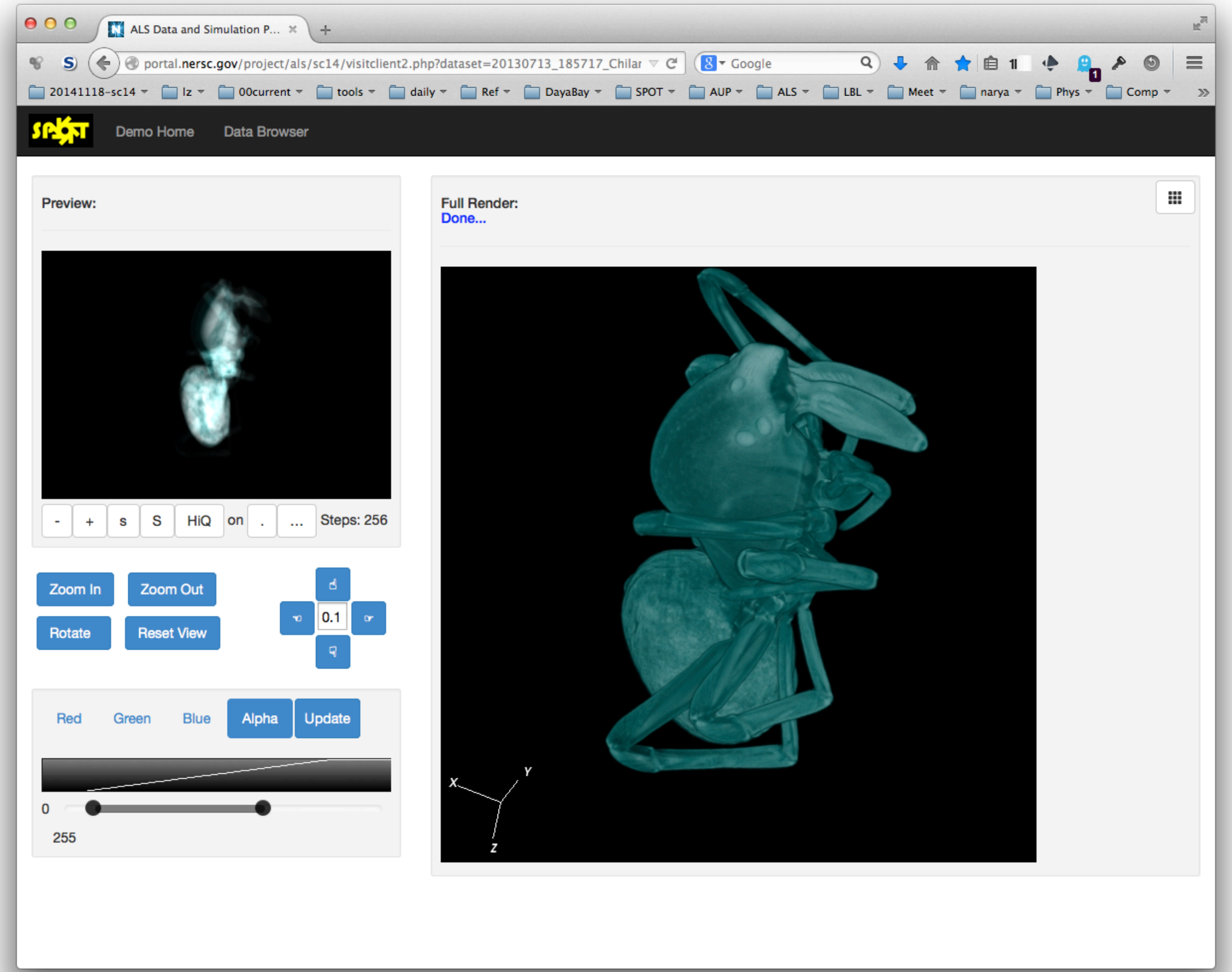
## Use Case: SPOT Suite Workflow



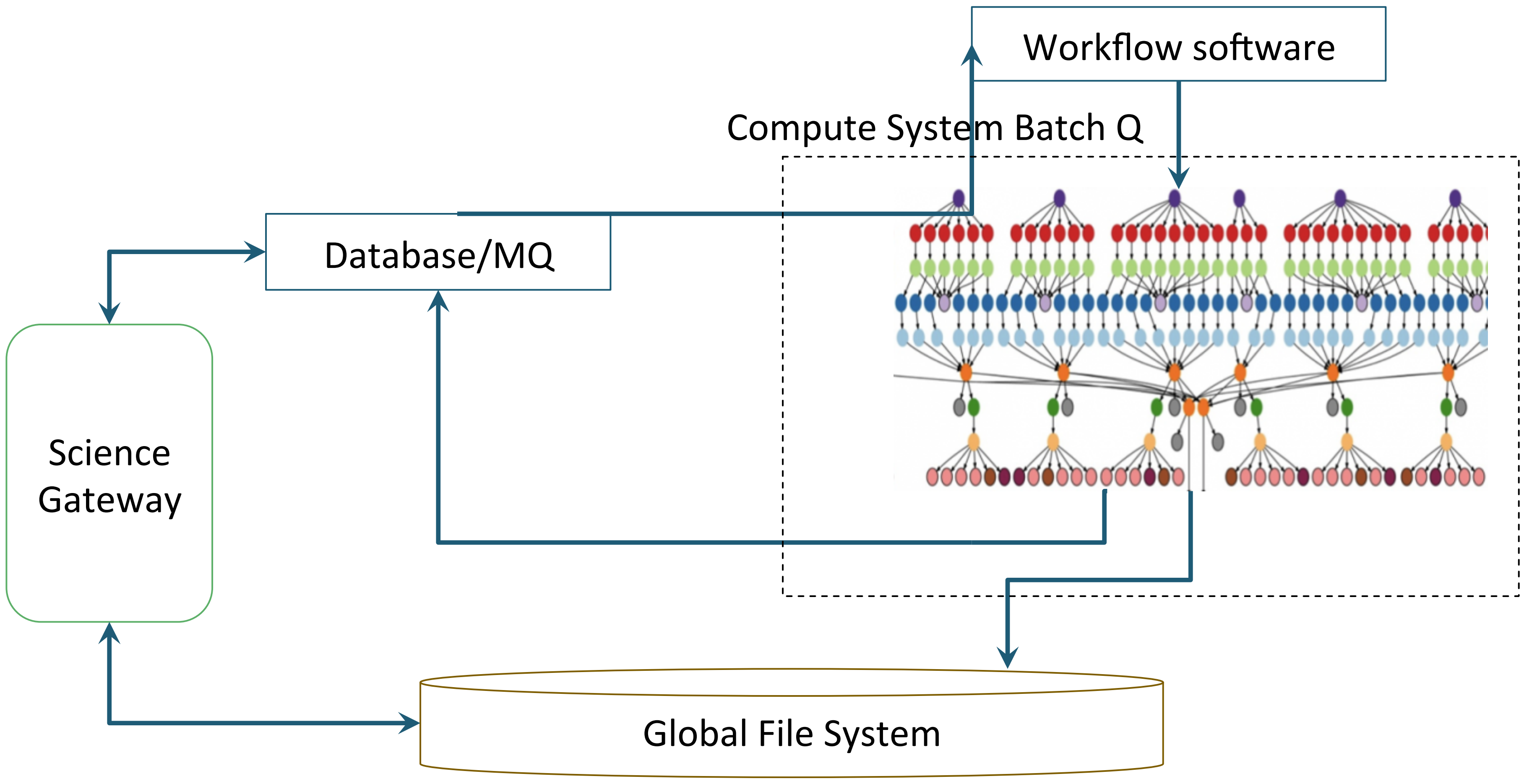
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

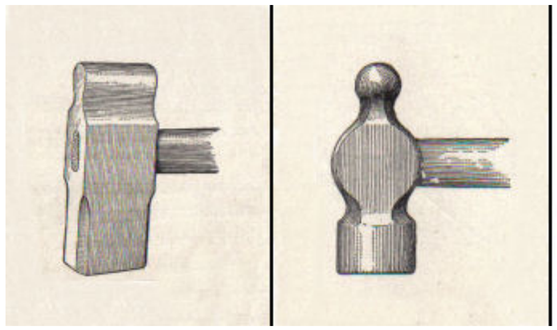
# SPOT Suite Gateway



# Tying it all together



## Finding the Right Hammer

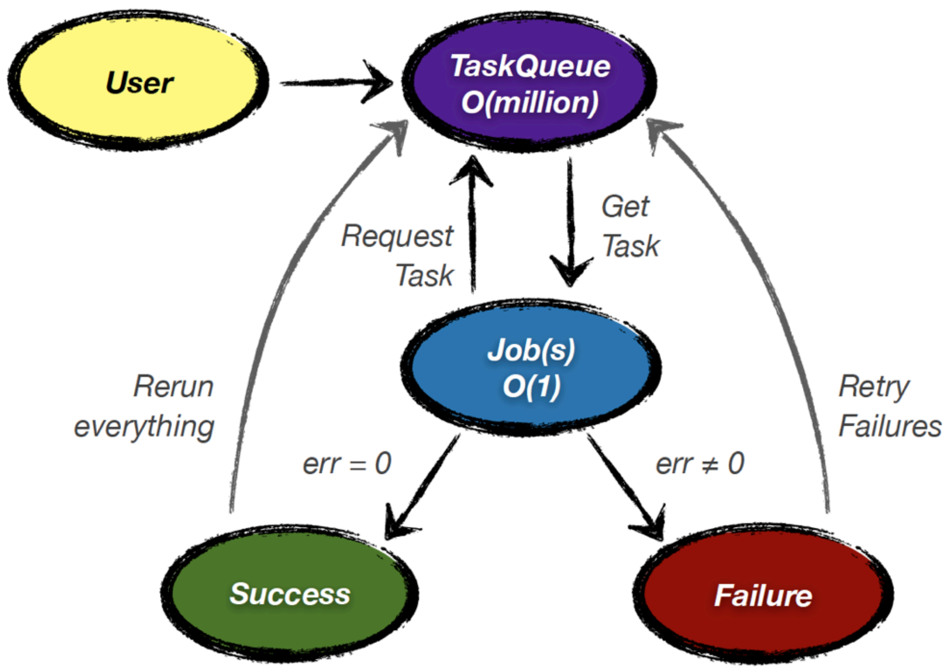


- Workflow tools have lots of features but there is no one size-fits-all
- NERSC is building expertise in classes of workflow tools and will help guide you towards the right tool for your job
- Consider stitching together a couple of different tools to make it all work

# Thank you.

## Use Case: qdo (cosmology)

qdo Model



- qdo is specifically designed to package up multiple small tasks into one batch job.

## qdo examples

```
#- Command line
qdo load Blat commands.txt     #- loads file with commands
qdo launch Blat 24 --pack      #- 1 batch job; 24 mpi workers

#- Python
import qdo
q = qdo.create("Blat")
for i in range(1000):
   q.add("analyze blat{}.dat".format(i))

q.launch(24, pack=True)

#- Python load 1M tasks
commands = list()
for x in range(1000):
    for y in range(1000):
        commands.append("analyze -x {} -y {}".format(x, y))

q.add_multiple(commands)   #- takes ~2 minutes
q.launch(1024, pack=True)
```

## Engagement

- **Enabling science in a scalable manner**
  – Build re-usable workflow components that can be used across domains.
  – Support a 2 to 4 classes of workflow tools
  – Create an ecosystem of services to enable new tools
  – Engage with domain specific science to address specific needs. Each project will have its own requirements. Bring those requirements to the table and we can evolve our ecosystem to meet your needs.