Workflow Tools at NERSC

Debbie Bard
djbard@lbl.gov
NERSC Data and Analytics Services

NERSC User Meeting
March 21st, 2016
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
- qdo
- Tigres
- Galaxy
- Swift
- BigPanda
- Pegasus
- Taverna
- Airavata
- ........
• Last year Workflows working group investigated breadth of technologies
• We ‘support’ 2 tools at NERSC
  – FireWorks
  – Swift
  – this doesn’t mean other tools won’t be used/supported at NERSC, only that DAS has specific expertise in these.
• Create an ecosystem to enable self-supported WF tools
  – Databases, User defined software modules, AMQP services etc.

http://www.nersc.gov/users/data-analytics/workflow-tools/
Existing Workflow Ecosystem @ NERSC

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

Workflow tools exist in and interact with a rich environment of NERSC capabilities and services.
Use Case: Swift

- Enables execution over multiple compute resources
- Swift language “encapsulates” application, easing distribution, parallelisation and provenance capture
Use Case: Swift (Biosciences)

- Functional language: powerful parallel loops
- Example: protein simulation for drug screening

```swift
Sweep(Protein pSet[ ])
{
    int nSim = 1000;
    int maxRounds = 3;
    float startTemp[ ] = [ 100.0, 200.0 ];
    float delT[ ] = [ 1.0, 1.5, 2.0, 5.0, 10.0 ];
    foreach p, pn in pSet {
        foreach t in startTemp {
            foreach d in delT {
                IterativeFixing(p, nSim, maxRounds, t, d);
            }
        }
    }
}
```

10 proteins x 1000 simulations x 3 rounds x 2 temps x 5 deltas = 300K tasks

https://www.nersc.gov/users/data-analytics/workflow-tools/swift/
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.
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.
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)
Many-task frameworks

• Repeatedly perform tasks on a large dataset
• Map => perform an operation across a large set i.e. map a task across the dataset
• Reduce => collect and reduce the results from map operation
• Split the data across nodes and run task on each node
• Typically does not require much cross node communication

• Frameworks at NERSC
  – Spark
  – Hadoop
    – MySGE
    – Taskfarmer
Batch Queues

• NERSC has queues suited to jobs that need less than one compute core
  – Cori Shared 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.
Use Case: TaskFarmer

- Simple NERSC-developed utility that farms single-core tasks onto a multi-core compute node, tracks job success

1) Write a wrapper that defines one task to run

```
cd $SCRATCH/myDir
python myScript.py $1 $2 $3
```

2) Make list of tasks to run, with options to pass to your wrapper

```
wrapper.sh 0 0 1
wrapper.sh 0 1 0
wrapper.sh 0 1 1
```

3) Write a batch script that will run your tasks on compute nodes

```
#!/bin/sh
#SBATCH -N 2 -c 32
#SBATCH -p debug
#SBATCH -t 00:05:00
cd $SCRATCH/myDir
export THREADS=32
runcommands.sh tasks.txt
```

http://www.nersc.gov/users/data-analytics/workflow-tools/taskfarmer/
Putting it all together: Materials Project

- Simulate properties of all possible materials.

\[ E\psi(r) = -\frac{\hbar^2}{2m} \nabla^2 \psi(r) + V(r)\psi(r) \]

Basic laws of Physics

Generally applicable to any chemistry

Density functional theory (DFT) approximation

Material Properties

https://www.materialsproject.org/
Materials Project Workflow

input: A cool material !!

Submit!

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

- Fireworks used to organise simulation and data workflow

```json
spec:
  _tasks:
    - fw_name: ScriptTask
      script: echo "hello" >> hello.txt
```

MongoDB containing task info and metadata

https://www.nersc.gov/users/data-analytics/workflow-tools/fireworks/
Use Case: Materials Project

• Tasks submitted to Fireworks MongoDB via API/python script etc.
• MongoDB keeps track of all tasks
• 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
Fireworks: Error Handling and Dynamic Workflows

- Can change next step of workflow, based on outcome of previous step
- Can specify action based on soft failures, hard failures, human errors
  - “lpad rerun –s FIZZLED”
  - “lpad detect_lostruns –rerun”
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 for you.
Thank you.