Running LSST sims with Fireworks and Docker at NERSC

Debbie Bard
NERSC

Overview

- Running sims at NERSC.
- Why is this an interesting workflow problem for us?
- Fireworks + Shifter = great.
- Implementation is easy.
- But what are you actually going to need?

NERSC

- We're moving to a new purpose-built building at LBL in the Fall.
- Shutting down old machines (Carver, Hopper).
- Next supercomputer: Cori
- **Cori Phase 1: September 2015**
  - ~1400 nodes, each with 2x16-core Haswell processors and 128GB.
  - "Burst Buffer" SSD sits between node and disk
  - SLURM job scheduler
  - External connectivity from every node
  - Containers/UDI support
  - *designed for data-intensive science*
- **Full Cori: summer 2016**
  - Knights Landing (Xeon Phi) processors
  - will retain Cori P1 as "data partition"

https://www.nersc.gov/users/computational-systems/cori/

Motivation

- **Sims motivation**:
  - how to run at scale, on heterogenous distributed architectures?
  - Already demoed running sims in a docker container, drawing run config from an external database.
  - Fireworks is a nice way of automating this.

- **Cori Phase 1 motivation**: want to exercise data features of Cori P1 - this is an ideal use case.
  - containers.
  - external access to the world from compute nodes.
  - improved disk IO.
  - workflow control.

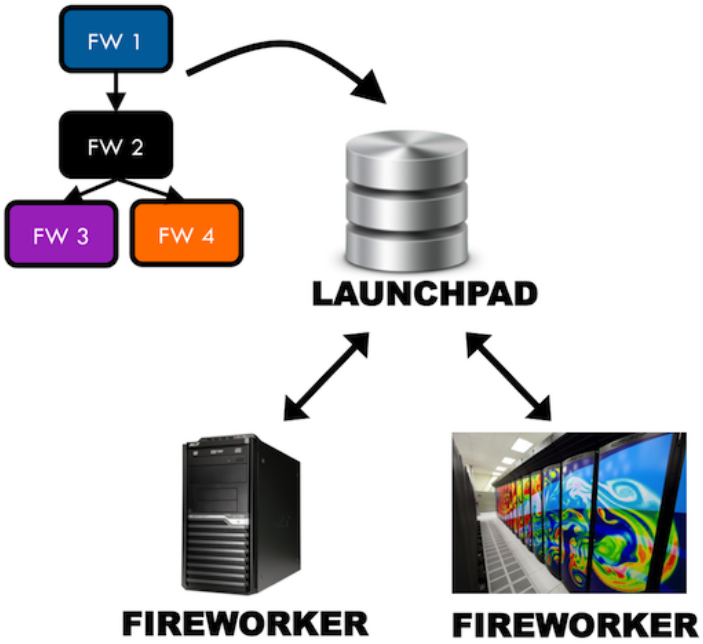
Containing docker: “Shifter”

- Docker has some issues in HPC
  - security: users have access to root in the image!
  - batch jobs need to run executable from image, not from user environment.
- Shifter is NERSC’s solution to these problems
  - wraps a docker image.
- Will run very easily with SLURM (Simple Linux Resource for Resource Management), the new NERSC job scheduling system.
- UDI is a key “Data feature” of the new NERSC Cori system.
- Open Source.
- ***This is how we’ll run containers on HPC architectures.***

https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2015/shifter-makes-container-based-hpc-a-breeze/

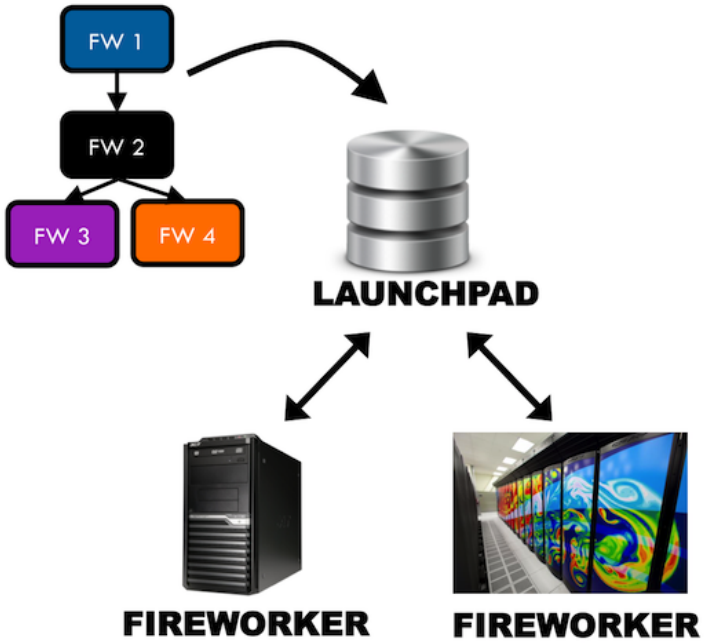
Fireworks

- Workflow engine designed for long-term distributed simulations
  - Materials Project, simulating properties of proposed compounds.
  - Used by groups at all DoE labs.
- Dynamic workflow
  - can modify behaviour of tasks down the line, based on output of previous jobs.
- Fault-detection routines
  - re-run failed tasks.
- Easy to write scripts!



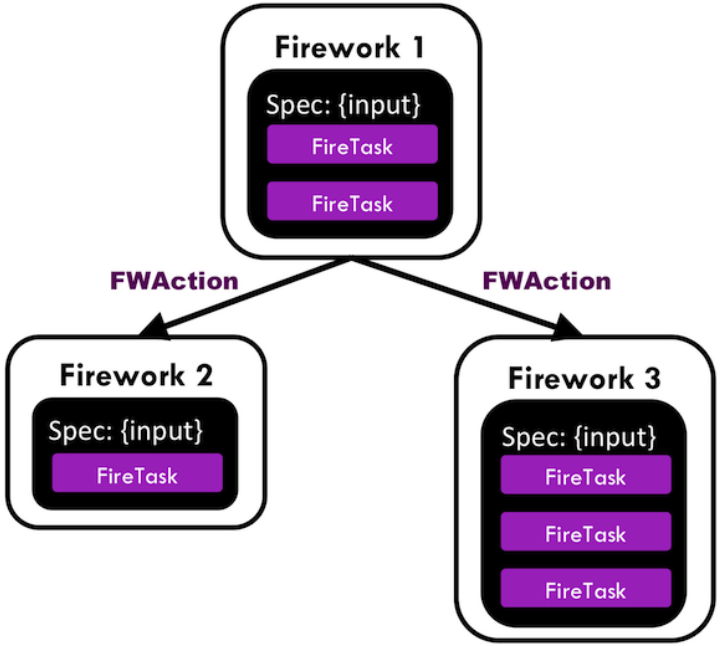
Fireworks

- Launchpad
  - MongoDB listing workflows (essentially just lists of compute ***tasks***).
  - Stores status of task, plus metadata.
  - Trivial to add more tasks in the middle of production without disruption (this is not the case for sql database).
- Your compute node (“Fireworker”) requests a task from the Launchpad and runs it.



Fireworks

- Complex workflows: can specify a sequence of tasks that are dependent on input from previous tasks.
  - e.g. could retain CatSim files at location they were generated, and only run PhoSim tasks on those files at same location (ditto for DM tasks).
  - e.g. if a CatSim task generated a very bright star in a field, could specify where/how to run that particular case (or to discard it).
- Basically, can specify task behaviour based on any MongoDB query.

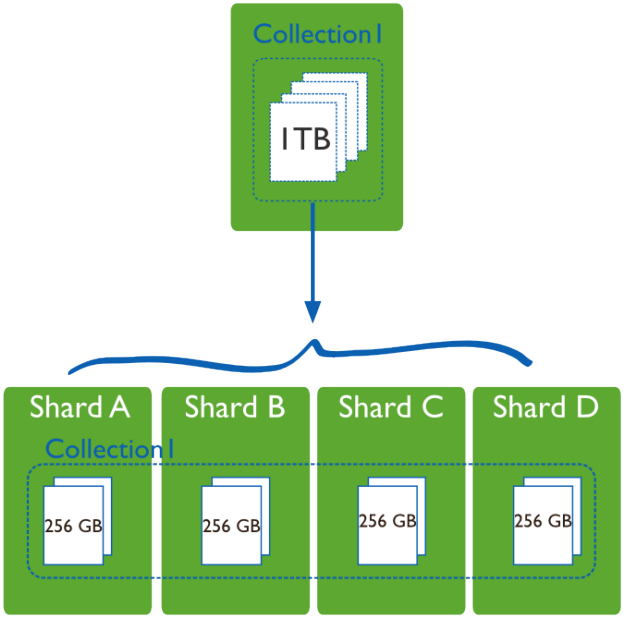


Fireworks, Shifter and PhoSim

- We want to run PhoSim inside a container:
  - Launch container on your compute node.
  - Container runs a FireWorks script that requests and runs a task from the Launchpad database.
  - Container closes upon completion of the task.
- In batch system at NERSC:
  - submit a shifter job (i.e. run the container on a batch node)
  - that's it!
    - job won't query the Launchpad DB for which PhoSim job to run until it starts execution.
- Cori will have a dedicated Serial queue for running non-MPI jobs (like phosim)
  - Currently: cannot run multiple executables per node on Edison.

Scaling up: Database issues

- 1000s of jobs querying one database will cause problems
  - locking and concurrency control.
- MongoDB (or other noSQL) can deal well with this
  - sharding: distribute collection over multiple instances. Each instance is independant and uses own locks.
  - We're currently fixing a similar issue at NERSC for material science sims.



Scaling up: Data issues

- 1000s of jobs hitting a disk would be a problem - need to write to local scratch and transfer
  - PhoSim needs minimal modifications to do this.
  - could try to bundle jobs so that one node pulls multiple chips from same exposure… worth the complication?
- Where to transfer the data afterwards?
  - could scp fits somewhere at end of job
  - could use data portal at production sites
    - job-control DB would be given url, and a central job on data location could pull in files as they finish, or bundle them up into a regular Globus transfer (probably optimal)
- Cori P1 Burst Buffer might be useful...

Test setup

- Docker image containing fireworks and phosim
- Batch (or interactive) job on Edison at NERSC (or my laptop, or wherever) asks the launchpad for the next job to run, and runs it.
- Very simple example, but very easy to add complexity.

Add jobs to launchpad with python script

define Launchpad (mongoDB) to which you're going to push tasks/workflow.

command string for phosim job. This is run inside the container.

make FireWorks task and workflow from command string.

add script task to Launchpad (DB).

```
from fireworks import Firework, LaunchPad, ScriptTask

## set up the lpad and reset it.
launchpad = LaunchPad(host="mongodb03", name="debbie_fireworks_db", username="[redacted]
[redacted]", password="[redacted]")
launchpad.reset('', require_password=False)

## loop over runs I want to make.
## I'm only changing the output dirs each time, but I could certainly change this
ntasks = 100

for i in range(ntasks):
    name = "phosim-" ## index is added in the db.
    outdir = "/global/scratch2/sd/djbard/phosim/out-"+str(i)
    workdir = "/global/scratch2/sd/djbard/phosim/work-"+str(i)
    script = "python /home/phosim-3.4.2/phosim.py -c /home/phosim-3.4.2/examples/nobackgro
und -w "+workdir+" -o "+outdir+" /home/phosim-3.4.2/examples/star"

    print script

    firetask = ScriptTask.from_str(script)
    firework = Firework(firetask, name=name)
    launchpad.add_wf(firework)
```

Job submission script

- qsub -v SHIFTER=docker:djbard/fw-phosim:v0.4 qsub-shifter-single.sh

[needed to set these variables for running on Edison, not for Cori.]

“rocket launch” from a launchpad defined in “my_launchpad.yaml”

[aprun will become srun (new load manager for Cori).]

```
#PBS -q debug
#PBS -l mppwidth=1
#PBS -l walltime=00:30:00
#PBS -N jobby
#PBS -j oe

module unload python
module load fireworks python
export CRAY_ROOTFS=UDI
export SHIFTER_RUNTIME=1

cd $PBS_O_WORKDIR

aprun -n 1 -b /bin/bash -l  -c "rlaunch -l /global/homes/d/djbard/phosim-edison/my_launchpad.yaml singleshot"
```

Launchpad Definition

- my_launchpad.yaml defines DB interface:

```
!!python/unicode 'host': mongodb03
!!python/unicode 'logdir': null
!!python/unicode 'name': debbie_fireworks_db
!!python/unicode 'password': [redacted]
!!python/unicode 'port': 27017
!!python/unicode 'strm_lvl': !!python/unicode 'INFO'
!!python/unicode 'user_indices': []
!!python/unicode 'username': [redacted]
!!python/unicode 'wf_user_indices': []
```

Questions:

- Is there interest in this?
- What timescale will you want to run sim jobs?
- What volume of sims?
- CatSim + PhoSim, I presume?
  - note that it's easy to define a workflow of a CatSim task, followed by a PhoSim task.
  - + DM?

- I plan to run this myself to test out some Cori Phase 1 features: are there useful amounts of sims you'll need in the Fall/Winter?