Shared Physics Analysis Facility @ BNL (Tier-3)

Overview and Plans

70 YEARS OF DISCOVERY

A CENTURY OF SERVICE

William Strecker-Kellogg
<willsk@bnl.gov>

USATLAS Workshop at ANL
Dec. 2018

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Timeline / Overview

2015 - consideration of what would required to support a shared Tier-3

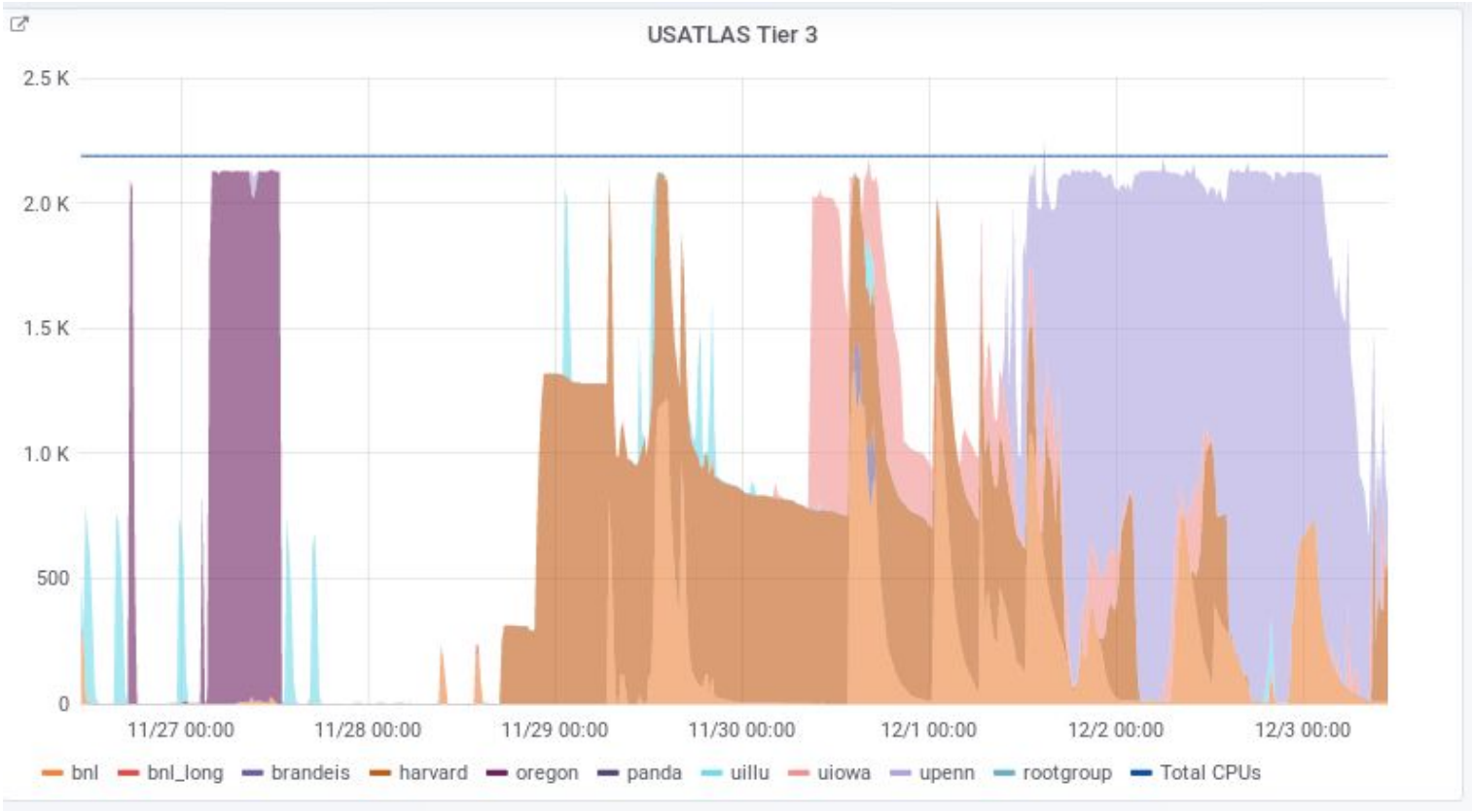
Initial design mirrored our condor pool at the Tier-1, using group quotas to map people's home institutions instead of panda queues.

Old interactive analysis hosts for BNL and some soon-to-be-retired batch nodes that we rolled into a new pilot Tier-3.

Present - improvements and growth (130+ users, 29 institutions)

Compute+Storage Resources

- SSH Gateways and NX servers available
- 12 interactive login nodes for job submission
- Batch: ~2200 CPUs providing 22.5k HS06
  - 2-3Gb/core
- 1Gb network non-blocking for each node
- Various Storage Technologies; per-user...
  - 5Tb Storage in dCache
    - Access through pNFS interactively, xroot on batch
  - 500Gb GPFS space
  - 20Gb NFS home-directory area



- Similar picture for the last 6 months as well
  - Occupancy lower than could be wished
  - Could benefit from more aggressive backfill* (more on this later)
  - Shared pool migration will address occupancy issues (next slide)

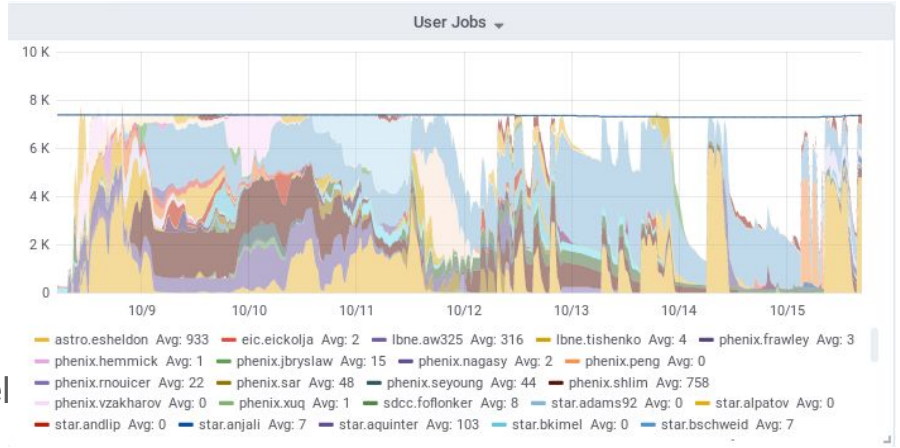
Last week's usage

Shared Condor Pool Migration

- See first half of this talk from HTCondor Week for more technical details
- General Idea: Move from several large pools to 1 (STAR, PHENIX, ATLAS, T3)
  - Current situation: Star + Phenix: 34kCore, Tier1: 20kCore, Several Smaller Pools
  - Goal: One single pool utilizing group-quota model for share (quota=contribution)
    - Without namespace conflicts, allows "grafting" of groups into each other
  - Reasoning: better sharing of resources between stakeholders, improved utilization
  - Need agreement on common policy parameters (time / size limits)
- Secondary Goal: partitionable slots everywhere
  - Tier-3 will get this so no need for dedicated high-memory or multicore slots
  - Preemption Issues:
    - HTCondor currently has major issues with starvation of multicore jobs by lower-priority single-core queues
      - No reservations, stateless matchmaking
    - Solution: no preemption, everyone agrees on runtime parameters
- Tertiary Goal: easier administration

Shared Pool Migration Status

- Initial shared pool with static slots and no quotas (excessed ATLAS hardware)
  - 7kCores
  - Good usage with "flocked" RHIC jobs
  - Hardware will be repurposed into new model



- Submit nodes will push jobs into shared pool first, then flock back to existing experiment hardware (if they retain any)
- Transition ongoing, will begin in earnest next week
- SPAR timeline TBD...

Backfill

- Lack of preemption is an issue for backfilling slots
  - How to kick out backfill work
  - What are reasonable expectations of latency to access one's own resources

- Conversation with Jarka + other CERN folks + Condor team about this last week

- Move to shared pool exacerbates this, May need an external component that intelligently drains or evicts susceptible jobs
  - Condor 8.8 will have a new "backfill" job-type that can fill in defragmenting nodes

Interactive Analysis

- Existing paradigm:
  - Split work between interactive and batch resources
  - Batch 100x-1000x size of interactive
    - Users sufficiently motivated to (learn to) use batch systems
    - Intuitively understand workflow:
      - Develop, compile, test, small-scale run, data movement, all on interactive nodes
      - Workflow processing done on batch
- New paradigm: Jupyter
  - Expanding interactive toolset
  - Lower barrier of entry—both for learning curve and user-base
    - Learning curve
      - SSH, Shell, Batch Systems, etc... "steep" for some newer users
      - More a problem in domains without such a long tradition of large-scale computing
        - Life Sci, Photon Sci, etc...
    - Userbase: supporting external users
      - From other domains, growing need to support external users
        - More a problem for institutions with a strict security posture

Jupyter at BNL

- I will be leading this effort in the new calendar year
  - Project is far wider than just ATLAS, however SPAR will be a "ideal" user
- Common solution for HTC and HPC resource access
  - Significant challenges, will touch on the HTC part for now (and common parts)

  Very early plans below!

- How to grant access:
  - Front-end authenticating proxy landing page
    - Choose running model (HTC / HPC, Local, Batch, etc...)
    - Get running instance appropriate to your security zone
      - Internal users get full access to our environment
      - External? Isolated in container, no POSIX FS access, storage by API/Token only
        - Scratch area? Home?
      - This is a radical departure from cyber-security norms!
  - Batch system (next slide)

Jupyter Batch Spawner

- Let's apply the same paradigm here!
- Small cluster of directly-launched jupyter instances
  - Load-balanced from a frontend proxy?
- Containers at this level? Why?
  - Better to support choice of user-environment via containers, no need for scheduling with container orchestration
  - (biased statement) Many, many attempts at "scheduling" as an afterthought in a project, almost always done poorly! Just use a batch system...
- Batch System!
  - HTCondor and Slurm support running a jupyterlab session as a batch job!
  - Containers can enter at the batch level to isolate external people
  - Or can be based on the choice of environment
  - Open questions:
    - Latency, cleanup, starvation

Experiment Environments / Containers

- (opinion) Containers solve the problem of different user environments, not the problem of scheduling / deploying units of compute
- When you get a jupyter session, what environment are you in?
- Create a "default" env by cloning our native farm-image
  - The one on the current farm nodes
- User choice at portal for which environment to start?
  - Local jupyter: spawn in container, access software in shared area
  - Batch spawner: batch system container layer to spawn
- No orchestration needed, but, whose problem is setting up the environments?
  - Collaborative between admins and experiment software folks

Integrating Jupyter with Compute

- How to make it easier to use compute from Jupyter?
  - See second half of talk referenced on slide 5
  - Abstract away using a batch system
  - Experimental code I wrote
- Goal: abstract away the fact that you are using a batch system at all
  - Either through trivial substitutes
    - map()→condormap()
  - Or (better) through cell "magics"
    - %slurm or equivalent

```
from condormap import condormap
import collections
import numpy


# Sample function
def logistic(r, len=10):
    d = collections.deque(maxlen=len)
    x = 0.4
    for _ in xrange(5 * 10**7):
        x = x * r * (1.0 - x)
        d.append(x)
    return list(d)


for k, d in condormap(logistic, numpy.arange(3.5, 3.6, 0.01), withdata=True):
    print sorted(d)
    t = set(round(x, 5) for x in d)
    print k, "Mode ", len(t)
```

Integrating Jupyter with Compute

- Collaborating with Swan folks at CERN
  - Had a Google SoC student make a really nice JS UI for job tracking
  - Used Ganga integration
    - We are looking into how to integrate HTCondor with this or do something similar
    - Ganga "api" make some assumptions that are not good for HTC
      - Have a student intern part-time who may work on this

Conclusions

1. Consolidating condor pools == a good thing for efficiency
   a. SPAR is a good candidate to participate in this
2. Jupyter is attractive for users; much work to be done integrating "new" paradigm into traditional HEP environments

Things I'd Like to Discuss

1. Jupyter deployment: pip, conda, containers?
2. Security: getting sign-off on mingling of web and interactive domains?
   a. External users (federation)?
3. New users: what technologies can be used to effectively isolate the vulnerable parts of your environment from untrusted users

SPAR-Visible Next Steps

1. Set up user-accessible local Jupyter platform
   a. Users can start playing here
   b. Enable batch-spawner for scaling load "depth"
   c. Scale load "width" via cluster of jupyter-enabled submit nodes
2. Develop front-end enhancements for selecting environment / type
   a. Will need to be able to select ATLAS environment among choices here
3. Work on "glue" tooling between Jupyter and ATLAS Environment