Mixing HTC and HPC Workloads With HTCondor and Slurm

Christopher Hollowell <hollowec@bnl.gov>
RHIC/ATLAS Computing Facility
Brookhaven National Laboratory

Co-authors: Joshua Barnett, Costin Caramarcu, William Strecker-Kellogg, Antonio Wong, Alexandr Zaytsev

Facility Background

The RHIC/ATLAS Computing Facility (RACF) at Brookhaven National Laboratory (BNL) has served the HEP/NP community for over 20 years

- Primary computing facility for the experiments at RHIC
- US Tier1 computing facility for the ATLAS experiment at the LHC

- Provide both processing and data storage services
  - Batch processor farm with over 50,000 logical cores (~2,200 servers)
  - Over 70 PB of data on tape (HPSS), and more than 45 PB of disk (GPFS, NFS, dCache, XROOTD)

Experimental HEP/NP processing workloads have traditionally fit a High Throughput Computing (HTC) model

- Embarrassingly parallel
  - Jobs fit on a single batch host, and are often single-threaded
    - Multi-node parallelization libraries (MPI, PVM) not used
- Runs well on commodity hardware/servers
  - No need for specialized low-latency interconnects between hosts
  - Limited, or no use of specialized processing hardware such as GPUs
- Often data intensive
  - Makes offloading HEP/NP processing (besides simulation) to commercial cloud providers expensive: still need local datacenters

Facility Background (Cont.)

Recently, scientific computing at BNL has undergone some consolidation as a result of BNL's Computational Science Initiative (CSI)

- RACF is now a part of BNL's Scientific Data & Computing Center (SDCC)

BNL's Computational Science Initiative First All-Hands Meeting

Facility Background (Cont.)

As a result of the recent SDCC consolidation, SDCC/RACF is maintaining BNL's HPC (High Performance Computing) systems, as well as our traditional HTC processing installations

- New equipment and staffing funds provided by CSI make this possible

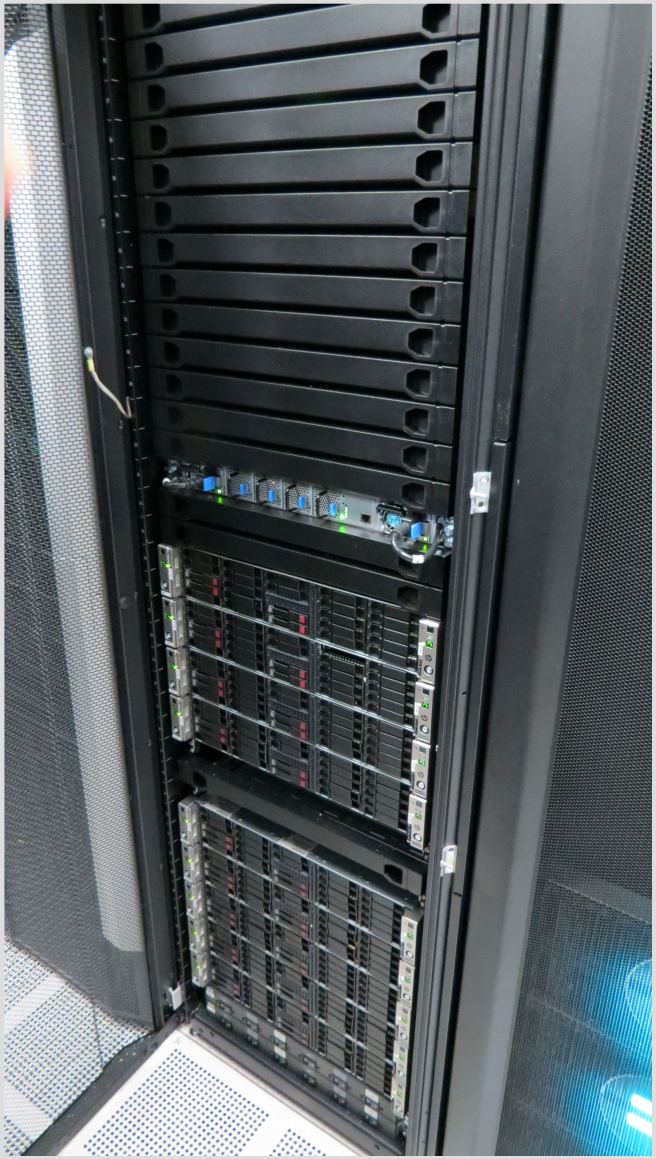
HPC resources being used by diverse user communities at BNL:

- CFN, LQCD Theory, Biology, NSLS-II, etc.

Architected and administrating two new HPC clusters:

- Institutional Cluster (IC)
  - 108 HP Proliant XL190r nodes (in 54 Apollo r2600 chassis)
    - Adding another 92 systems soon
  - 2 Intel Xeon E5-2690v4 CPUs per system
  - 256 GB RAM
  - 2 K80 GPUs per system
  - EDR Infiniband interconnect
- KNL Cluster
  - 144 Intel S7200AP-based nodes (in 36 Intel H2312XXL2 chassis)
  - 192 GB RAM
  - 1 Knights Landing Xeon Phi CPU per system
  - Dual-rail Omni-path interconnect

BNL's Institutional Cluster



Adaptation to Support HPC Cluster Infrastructure

Significant portions of the knowledge/skill needed to maintain and architect HTC systems also applies to HPC cluster administration

However, staff had to become familiar with some technologies we haven't used before

- Software
  - HPC applications generally make use of the MPI message passing library
    - Further complicated by multiple MPI implementations
  - Batch system
    - HTCondor's “parallel universe” only functional for small clusters with dedicated ownership
      - Adopted a different batch system than on the HTC side of the facility: Slurm
- Hardware
  - Processing
    - GPUs, Xeon Phi
  - Low latency interconnects
    - Inifiniband, Omni-path
    - Typically available on HPC clusters to greatly improve inter-node message passing efficiency

HTCondor

Developed at the University of Wisconsin

Introduced as an LSF replacement at our facility over 10 years ago
- BNL an early adopter of HTCondor in the HEP/NP community (2004)
- HTCondor is freely available, open source software - saved us significant LSF licensing costs

Excellent for HTC use
- HT - “High Throughput” is in the name

  Goal of the software is to fill all available batch slots as quickly as possible
  - Fairness of resource assignment (user priorities) is a secondary consideration, balanced over a large integral of time

  Highly scalable in handling large numbers of disparate jobs
  - In our environment, a single central manager/negotiator pool easily manages more than:
    - 25,000 batch slots and running jobs
    - 200,000 idle jobs

Features such as group quotas and partitionable slots have allowed us to implement advanced scheduling policies

HTCondor – Limitations in an HPC Environment

HTCondor has support for multi-node job scheduling – “parallel universe”

Parallel universe limitations – suitable for small dedicated clusters

- Supports FIFO or best fit scheduling only – can't easily be used in an environment where different users/groups pay for shares of a cluster and opportunistic usage is permitted
- All job submissions limited to a single submit node running condor_schedd
  - “Dedicated scheduler”
  - Reduces scalability
    - Our HTC systems utilize many submit hosts to improve scaling, and limit single points of total failure
- Using with MPI requires the user to utilize custom “mpiscripts” that call condor_ssh to start up processes on the different nodes
  - Different workflow than with more HPC-centric batch systems
    - Can confuse users
  - ssh = slow job spawning times

Slurm

Most HPC clusters being maintained by other groups at BNL were using the Moab/TORQUE batch system
- Significant cost
- Decided to investigate Slurm as an alternative

Primarily developed by SchedMD, with a number of outside contributors
- Freely available, open source

In use at many large HPC centers, including NERSC

Supports many useful/required features for our HPC clusters
- Advanced fairshare scheduling policies for parallel/multi-node jobs
  - Whole-node scheduling available
  - Scheduling of GPU resources supported
  - Allows users to request particular NUMA/MCDRAM KNL configurations
    - Includes scheduling of node reboots to implement NUMA/MCDRAM configuration changes
  - Partitioning of resources, and advanced reservation
- PMI(2/x) based process spawning – fast job startup times
- Network topology aware
- Highly available

Slurm – Limitations in an HTC Environment

Anecdotal reports from other HEP/NP sites that Slurm is not capable of handling large numbers (many thousands) of running/queued jobs
- Required for HTC use with many independent jobs
- Slurm recently published an HTC optimization document, however:
  - http://slurm.schedmd.com/high_throughput.html

Many of Slurm's features not required in an HTC environment
- Can complicate administration

Slurm accounting requires adding users to Slurm's accounting DB
- An extra administrative step not necessary with HTCondor

Our HTC users are used to working with HTCondor
- It's working well, and efficiently scheduling resources – not desirable to change

Possible to Allow for HTC ↔ HPC Opportunistic Use?

Due to different requirements and capabilities, we're using different batch systems for our HTC and HPC installations

Would it be possible to allow our HTCondor users to make use of our Slurm-managed HPC resources, and vice-versa?

- Purely a technical investigation
- No official resource sharing relationship defined between our HTC and HPC installations

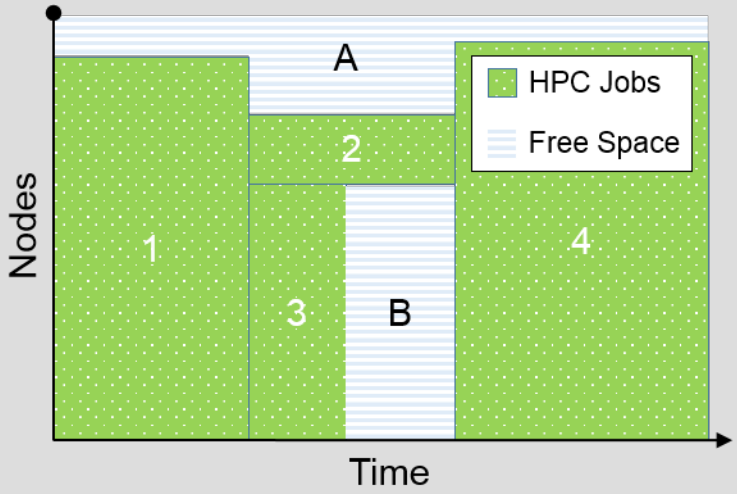
A mechanism for automatically transforming “vanilla universe” jobs to the “grid universe” has been included in HTCondor since version 7.1.0:
the condor_job_router daemon

Recently, HTCondor added support for a new “slurm” batch type to the grid universe

- Allows HTCondor to interface with Slurm
- Currently only available in the development branch
- Initially included in the 8.5.3 release in Mar 2016
- Investigated this functionality for HTC → HPC opportunistic use

Using HTCondor's Job Router with Slurm

Thanks to Joshua Barnett, an RACF/BNL summer student, for his work implementing this project



Availability of unused processing resources in HPC clusters is likely due to the inherent packing inefficiencies of multi-node jobs

- Can easily fill these unused resources with single-core HTC jobs

In a test environment, we were able to demonstrate opportunistic usage of Slurm-managed HPC resources via HTCondor, nearly transparently to users

- To opt-in, users simply add the following line to a job's JDF:

```
+WantJobRouter = LastRejMatchTime =!= UNDEFINED
```

- Created a separate low-priority, preemptible “condor” partition in Slurm for opportunistic use. Example configuration from slurm.conf:

```
PartitionName=condor Shared=FORCE OverSubscribe=YES
State=UP Nodes=test000[1-6] MaxTime=1-00:00:00
DefaultTime=05:00 qos=condor Priority=1
PreemptMode=cancel Default=no
```

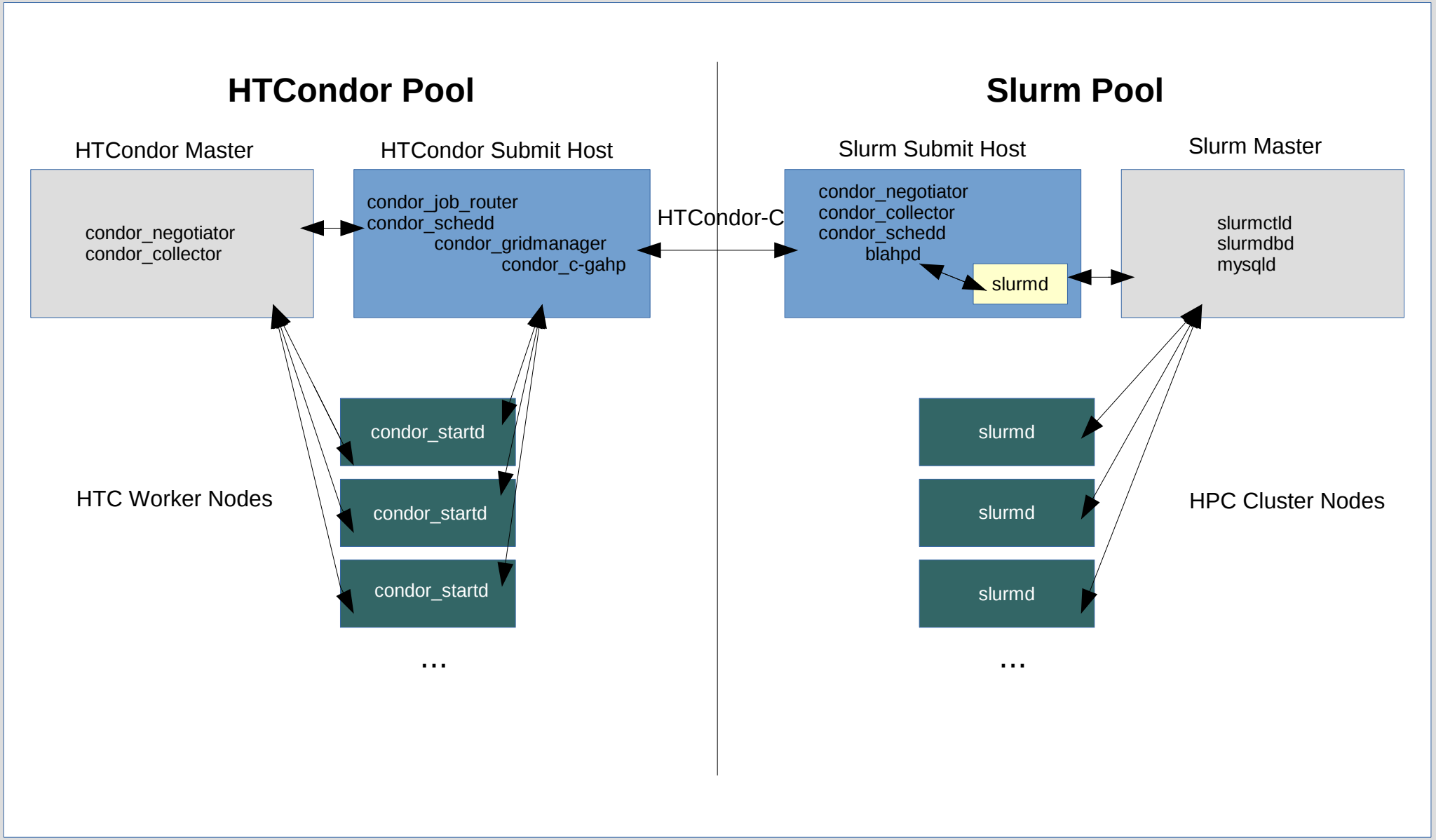
Using HTCondor's Job Router with Slurm (Cont.)

Chose to use HTCondor-C communication for the routed grid universe job
- Would have required SSH (BOSCO) access to the Slurm submit host to directly specify “slurm” as the grid resource
- Required instantiating a separate HTCondor pool consisting of the collector, negotiator, and schedd on the Slurm submit host instead

Enabling the condor_job_router daemon in the HTCondor configuration (/etc/condor/config.d/20_router):

```
# These settings become the default settings for all routes
JOB_ROUTER_DEFAULTS = \
[ \
      requirements=target.WantJobRouter is True; \
      MaxIdleJobs = 20; \
      MaxJobs = 100; \
      /* now modify routed job attributes */ \
      /* remove routed job if it goes on hold or stays idle for over 6 hours */ \
      set_PeriodicRemove = JobStatus == 5 || \
                  (JobStatus == 1 && (CurrentTime - QDate) > 3600*6); \
      set_requirements = true; \
]
JOB_ROUTER_ENTRIES = \
[ GridResource = "condor slurmsub01.sdcc.bnl.gov slurmsub01.sdcc.bnl.gov"; \
      name = "Slurm_Pool"; \
      Set_remote_JobUniverse = 9; \
      set_remote_GridResource = "batch slurm"; \
      set_remote_Requirements = False; \
]
JOB_ROUTER_SCHEDD2_NAME = condsub01.rcf.bnl.gov
JOB_ROUTER_SCHEDD2_POOL = condmaster01.rcf.bnl.gov
```

HTCondor's Job Router with Slurm – Architecture



HTCondor's Job Router with Slurm – Architecture (Cont.)

Job Router Architecture in Action

User submits a “vanilla” universe JDF to the local condor_schedd

```
condsub01% condor_submit myjob.jdf
```

If no local job slots are available, condor_job_router daemon submits a new copy of the job to condor_schedd with the universe transformed to “grid”

```
condsub01% condor_q
-- Schedd: condsub01.rcf.bnl.gov :
ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
5216.0   testuser        9/21 16:07   0+00:00:00 I  0   122. testloop.sh
5217.0   testuser        9/21 16:07   0+00:00:00 I  0   122. testloop.sh

condsub01% condor_q -format '%s:' ClusterID -format 'Universe = %s\n' JobUniverse
5216:Universe = 5
5217:Universe = 9

condsub01% condor_q 5217 -autoformat:n RoutedFromJobID GridResource
    remote_GridResource
5216.0
condor slurmsub01.sdcc.bnl.gov slurmsub01.sdcc.bnl.gov
batch slurm
```

condor_schedd spawns a condor-c_gahp server to submit the job to the remote HTCondor pool running on the Slurm submit host

HTCondor's Job Router with Slurm – Architecture (Cont.)

Job Router Architecture in Action (Cont.)

Job is queued in remote condor_schedd

```
slurmsub01% condor_q -nobatch
-- Schedd: slurmsub01.sdcc.bnl.gov :
ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
3793.0  testuser         9/21 16:09   0+00:00:00 I  0   122. testloop.sh

slurmsub01% condor_q 3793 -autoformat:n JobUniverse GridResource
    remote_GridResource
9
batch slurm
undefined
```

condor_schedd spawns blahpd to submit the job to Slurm, and manage it

Calls /usr/libexec/blahp/slurm_submit.sh, which in turns call sbatch

Slurm runs the job on an available HPC cluster node

```
slurmsub01% squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5483     condor bl_76b98 testuser R       00:30      1 hpcnode010
```

All copies of the job updated to reflect the “running” status

HTCondor's Job Router with Slurm – Potential Issues

Can take up to ~5 minutes for the local copy of the job in the queue to reflect remote state

- When job completes, the local accounting statistics are updated to accurate final values

Assumes user exists in both the HTC and HPC systems with the same username

Slurm's accounting DB doesn't know about HTCondor/HTC users

- Prevents them from being able to run
- Created a cronjob that periodically (1/min) checks if the owner of a job in the HTCondor queue on the Slurm submit host exists in the DB
  - Calls sacctmgr to add the user if not present - uses a cachefile list of previously seen users to avoid unnecessary sacctmgr calls

HPC HTCondor pool spool directory needs to be in a shared filesystem (i.e. NFS) in order for Slurm nodes to run the user's executable – hoping this requirement will be removed as HTCondor Slurm grid support is improved

- HTCondor copies user's executable to the spool directory, so it needs to be visible on the Slurm compute hosts
  - One-line change on the Slurm submit host's HTCondor configuration:

```
SPOOL = /nfs/spool/condor/
```

HPC → HTC Opportunistic Usage

Allowing our HPC users to opportunistically run HPC jobs on our HTC systems isn't currently possible

- Don't want to run a single HTCondor “dedicated scheduler” to support parallel universe
- HPC applications generally require low-latency node interconnects to run efficiently
  - Most of our HTC systems only have 1 Gpbs ethernet connectivity
- Many of our HPC users require GPUs or Xeon Phi processors, which aren't available on our HTC resources

However, some of our HPC users have been able to reduce their jobs to running on a single CPU, or single host, making opportunistic usage a possibility using HTCondor Flocking

- No way to transparently route jobs in the Slurm queue to HTCondor
  - Users need to submit HTC jobs to the HTCondor pool on the Slurm submit host directly

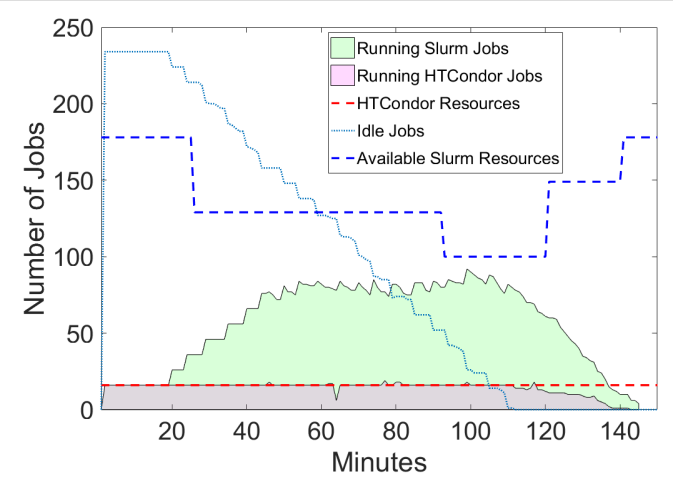
Conclusions

After many years of maintaining HTC equipment for HEP/NP, RACF has now also been tasked with administrating HPC systems for other user groups at BNL

Due to differing requirements and capabilities it is was decided to administrate two separate batch systems for our HTC and HPC users: HTCondor and Slurm, respectively

Using a configuration we've developed which utilizes the condor_job_router daemon, it's technically possible to allow our HTC/HTCondor users to opportunistically utilize our HPC/Slurm resources

- Can be done in a way which is fairly transparent to users
- HTC users benefit from the “commodity-centric” nature of their hardware requirements



Possible to allow our HPC users to opportunistically utilize our HTC installations as well via HTCondor flocking, but currently only if they plan on running HTC-like jobs

- Users need to manually submit such jobs to HTCondor