# Exploring the use of containerized HTC workloads for running HL-LHC analysis on HPC centers

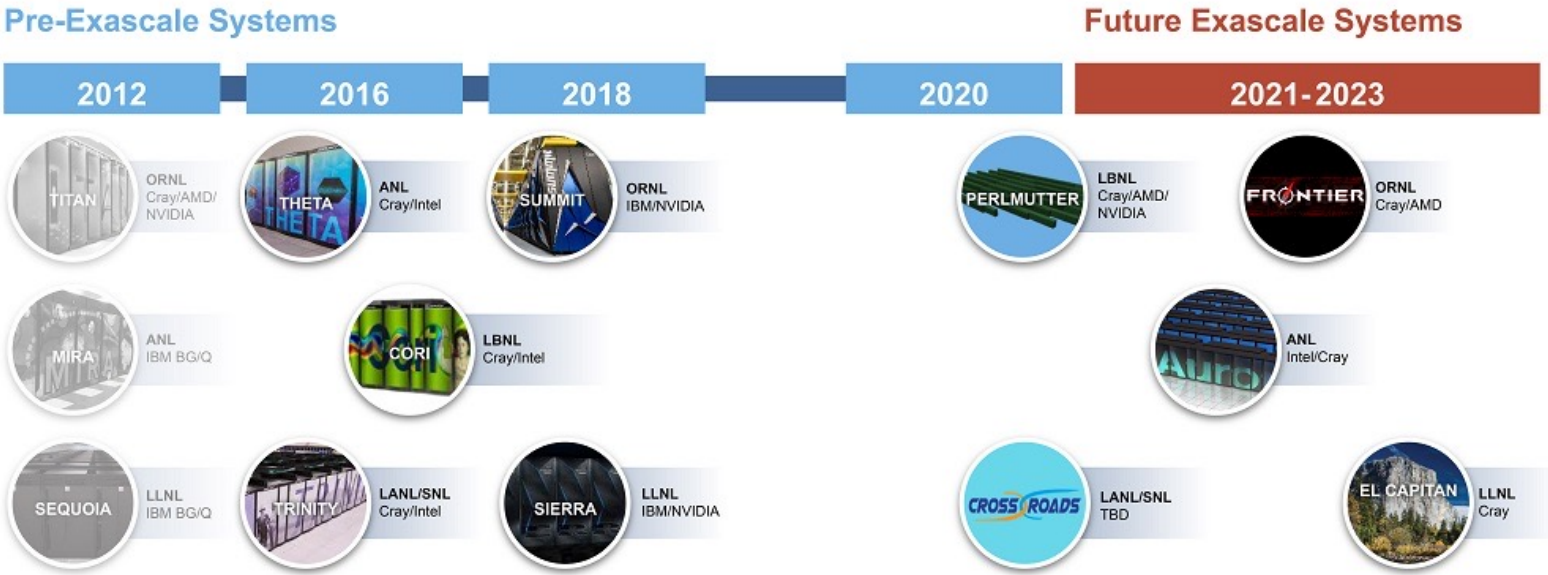
Maria. P. Acosta Flechas – Computing Services Specialist @ Fermilab

HTCondor Week 2022

May 24th 2021

# HPC panorama: going into the Exascale era

Within the federal government, the Department of Energy (DOE) leads the effort towards leading the world in high performance computing (HPC), with the nation's fastest and most capable supercomputers housed at the DOE national laboratories.



# How do we seamlessly run HTC workloads on HPC systems?

Depends … not one fits all

- Limited remote access for:
  - Job submission
  - Data transfers
  - Limited or no general network communication
- High-bandwidth shared filesystem
  - Both external and internal access (through Login/Edge machines)
  - We can move jobs in addition to their data

## HPC Batch system integration




COBALT

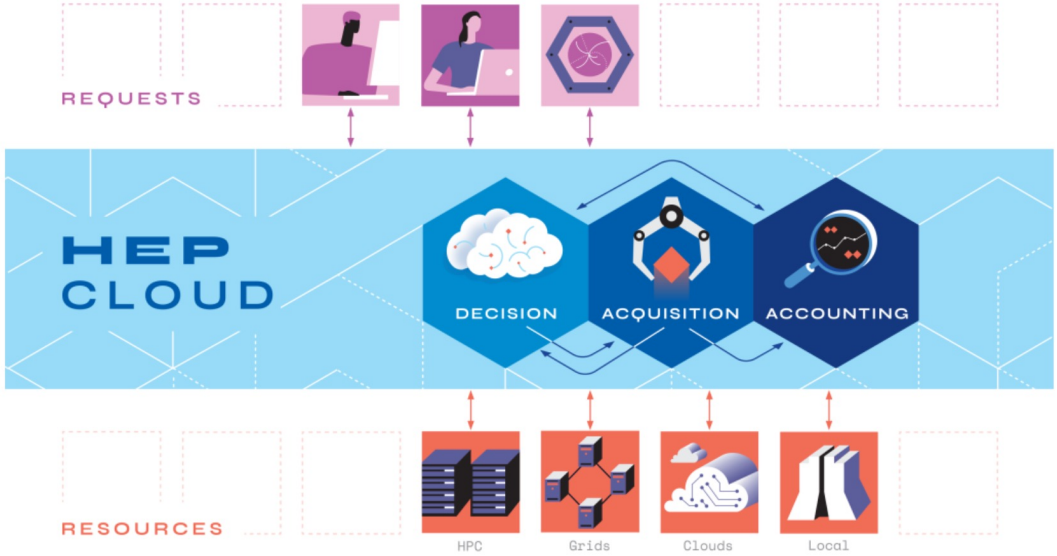
(Component Based Lightweight Toolkit)

## HTCondor binaries availability




# HEPCloud @ Fermilab

"A scientific gateway to resources beyond local worker nodes and grids, expanding into high performance computing (HPC) centers and the cloud."




https://hepcloud.fnal.gov/

# HEPCloud: Exploring diverse integration paths

## Opportunities

- Modular design of the Decision Engine
- Already capable of handling commercial cloud bursts and HPC submission to: NERSC Cori, TACC Frontera, Expanse, Purdue-Anvil, Fermilab's Wilson Cluster and more.
- Underlying GlideinWMS functionality provides the DE with powerful resource management and allocation.

## Challenges

- Project does not own or operate experiment submit points (CMS WMAgent, Jobusub for FermiGrid)
- Jobs need to be moved to HEPCloud-capable Schedds in order to make use of these resources.
- Several monitoring attributes and traceability are key and need to be delivered to the experiment.
- Data transfer systems are tightly coupled to each experiment central computing.

# Theta supercomputer at Argonne National Laboratory

Theta is an 11.7-petaflops supercomputer based on Intel processors and interconnect technology, an advanced memory architecture, and a Lustre-based parallel file system, all integrated by Cray's HPC software stack.

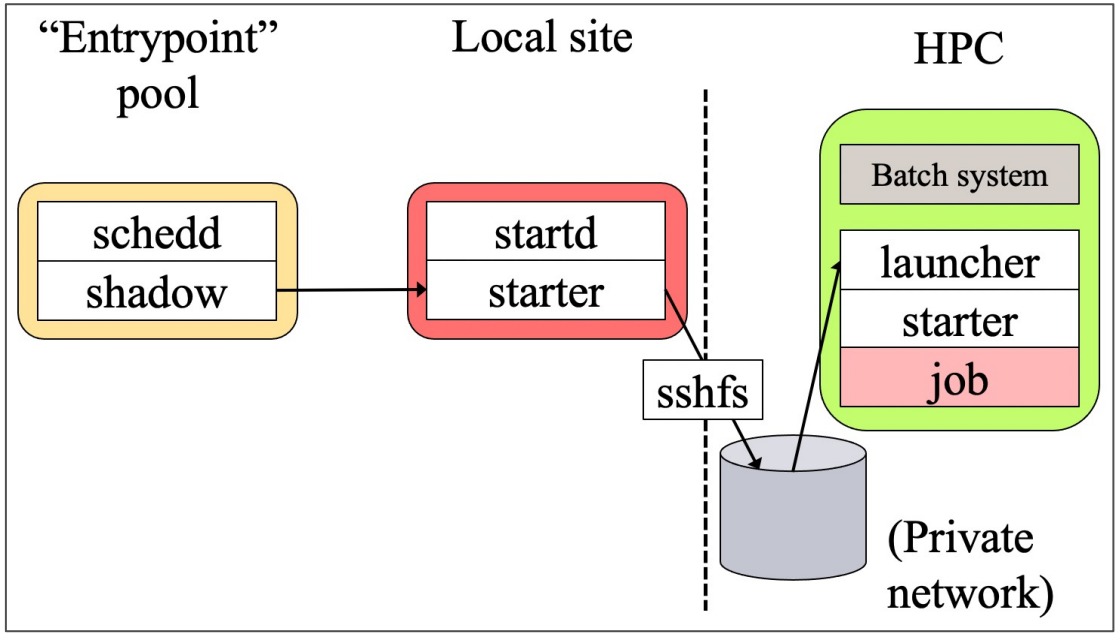
| | Per Node | Aggregate |
|---|---|---|
| Compute Nodes | Intel KNL 7230 | 4,392 |
| Compute Cores | 64 | 281,088 |
| Compute Memory - DDR4 | 192 GiB | 843,264 GiB |
| Compute Memory - MCDRAM | 16 GiB | 70,272 GiB |
| Compute SSD | 128 GiB | 561,176 GiB |

# Theta and HEPCloud


- ALCC allocation for FY21 enabled R&D effort for implementing the split/starter on Theta.
- Extensive testing and multiple code refactors later, we successfully ran HL-LHC production workflows via the split/starter method.
- Smaller allocation was granted for FY22 enabling a second phase of R&D for Lumberjack.

# HTCondor split-starter

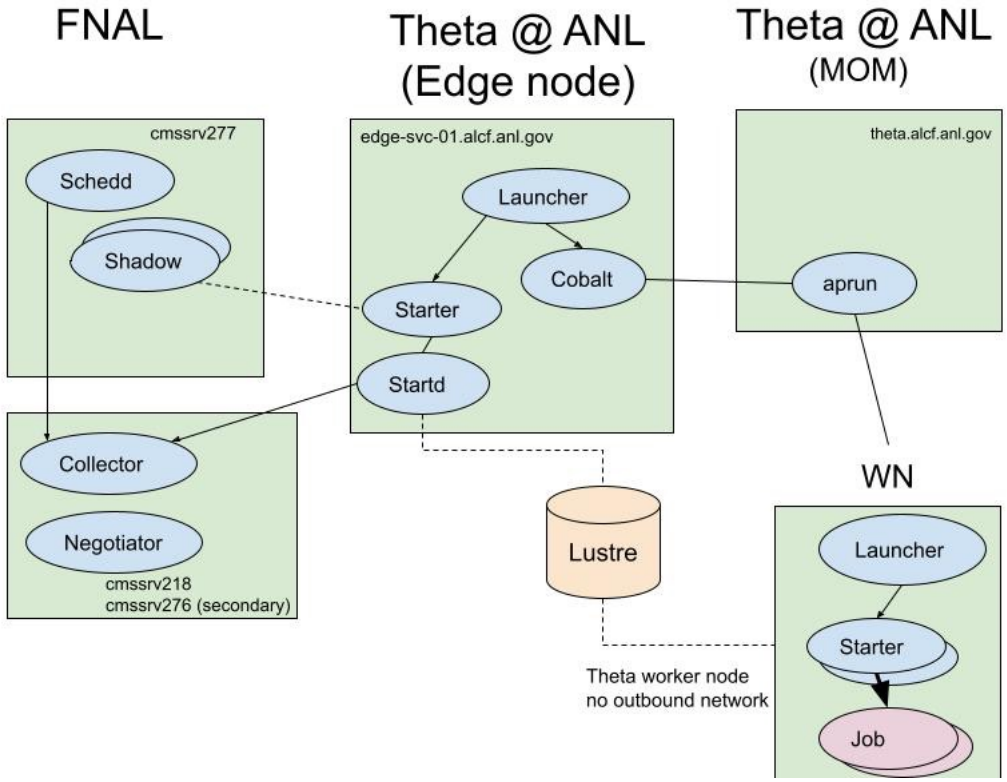
- Proxy startd on public network
  - Accepts jobs
  - Starter writes job + data to filesystem
- On execute node
  - Job + data copied to local disk
  - Standalone starter runs job
  - Results written back to filesystem




Based on diagram from: https://indico.cern.ch/event/936993/contributions/4022104/

# HTCondor split-starter – HEPCloud extras

- Pull jobs from CMS WMAgents
  - Job router grabs 'T3_US_ANL' jobs and submits them to our local schedd
  - Job is duplicated in the local queue.
  - Original job stays Idle, copy reflects state of routed job
- "Edge node" provisioning
  - Runs all FNAL side starters
  - Initiates interactions with Theta
  - Keeps track of jobs
  - Talks to local pool central managers
- Extra services available to the startd
  - Squid service for CVMFS
  - Non-FUSE, unprivileged CVMFS through cvmfsexec
  - Singularity



# HTCondor split-starter – Results

- An excellent alternative for heavily restricted inbound networks as most communication happens through files via SSHFS or shared storage (if available).
- Ephemeral and effective, will report to the remote pool as regular Startds.
- An ideal scenario where we benefited from lightweight credentials in the form of IDTOKENS.
- Snowmass Gen Campaign 2021 on US HPC assigned exclusively to ANL-Theta ran on production split/starters resulting in ~14.4M CMS HL-LHC events
- ALCF support facilitated R&D efforts, provisioning machines for supporting proxy starter workloads outside of shared login nodes.

# The idea of Lumberjack (credits to Jaime Frey)

A set of jobs in the local schedd are flagged as managed by an external scheduler and they end up in some remote schedd for scheduling and execution

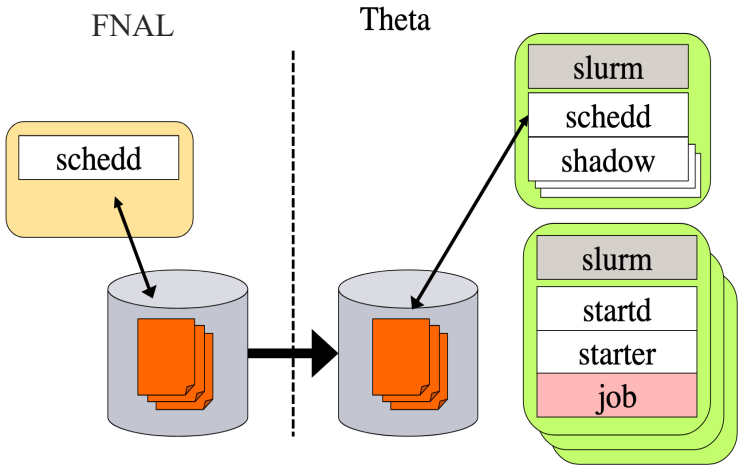


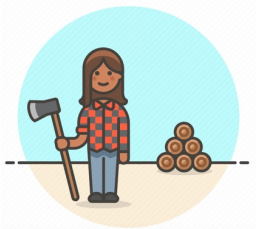

Diagram from: https://indico.cern.ch/event/936993/contributions/4022104/

- "Local" or jobs to be exported are regular vanilla jobs.
- Explicit "export" and "import" client commands prepare the jobs for movement to another schedd and return the results.
- The user is responsible for moving all of the job-related files to/from the remote schedd location.
- No intermediate updates of job status are made to the local schedd.
- The remote schedd is a newly-created schedd intended to just run these jobs.

# Lumberjack – HEPCloud workflow

A set of jobs in the local schedd are flagged as managed by an external scheduler and they end up in some remote schedd for scheduling and execution

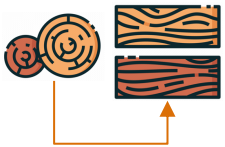
HTCondor Schedd: Forest of jobs waiting to run



1. Export metadata, input and SPOOL files of a subset of jobs defined by a constraint expression

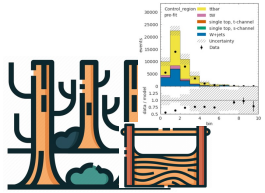
2. Estimate, adjust and prepare for remote processing

3. Containerized remote processing (sawmill)



4. Gather results and outputs

5. Import metadata, SPOOL and output files of processed jobs previously exported



6. Schedd marks imported jobs as finished, results and outputs are staged out transparently for the user


job_queue.log

# 1. Export job metadata and inputs

- In order to facilitate issuing import/export commands to the Schedd, I wrote a python wrapper for Lumberjack methods which allows us to do this via the command line:

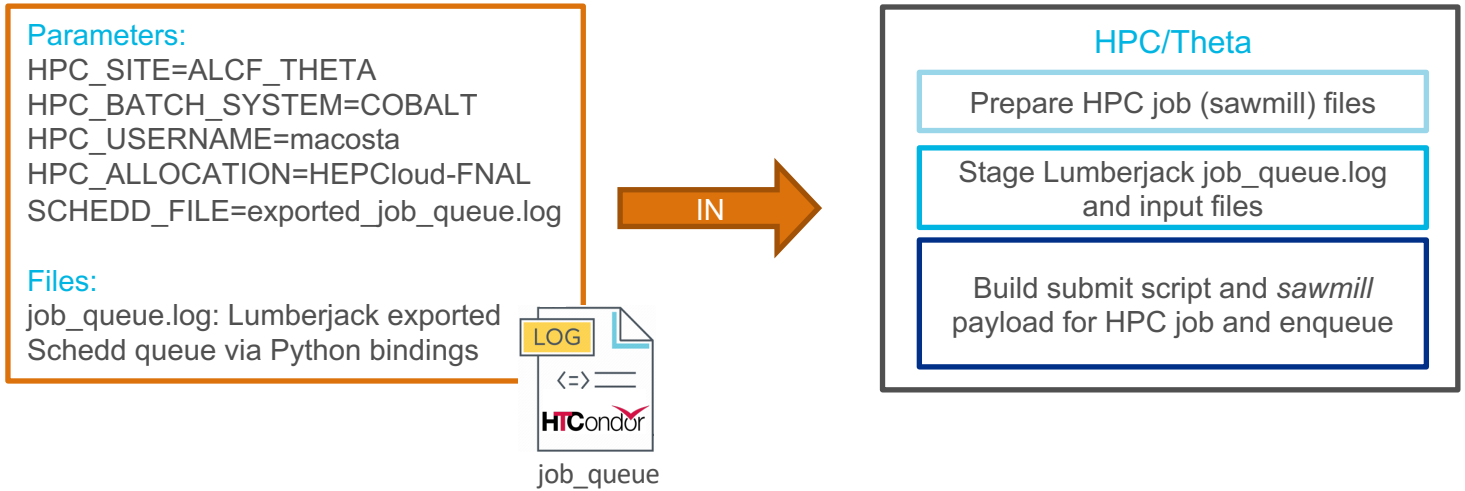
```
usage: condor_lumberjack.py [-h] [--export] [--import]
                            [-jobconstraint JOBCONSTRAINT]
                            [-ids IDS [IDS ...]] [-out OUT]
                            [-remotespool REMOTESPOOL] [-in INPATH]

Queries an HTCondor Collector for Schedd objects.
 If the "--export" flag is used, exports a group of jobs matching a constraint OR a list of ClusterIDs to the output directory specified by the -out argument.
 If the "--import" flag is used, imports an -already- exported job queue file specified by the -in argument to the local Schedd

optional arguments:
  -h, --help            show this help message and exit
  --export
  --import
  -jobconstraint JOBCONSTRAINT
                        Constraint expression (String) for selecting jobs to export
  -ids IDS [IDS ...]    Space separated list of ClusterIDs (Int) to export
  -out OUT              Output directory for the exported job file
  -remotespool REMOTESPOOL
                        Path of the SPOOL directory on the remote Schedd
  -in INPATH            Path to the exported job queue file to import
```

- This script needs to be run from the Original Schedd, it passes a job constraint, or a set of job IDs to HTCondor and places outputs into a defined output directory
- https://github.com/HEPCloud/fnalhpc_startd/blob/master/lumberjack/condor_lumberjack.py

# 2. Staging and resource estimation

What is happening?

1. File staging for the Schedd process within the container is done automatically.
2. SPOOL and input file paths are 'translated' for emulating original Schedd paths within container and stored in a key,value hash file.
3. A 'sawmill' job is submitted to the HPC

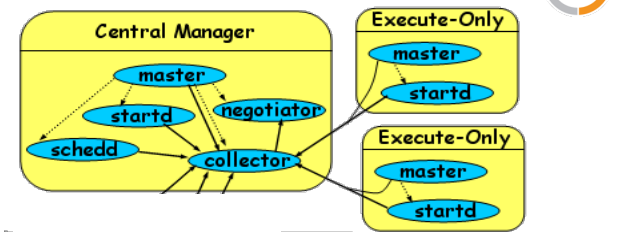


# 3. Containerized remote queue processing

aprun sawmill_payload_start.sh

```
configure_sawmill --file /scratch/job_queue.log
frontier_squid start
cvmfsexec -- singularity run docker://hepcloud-sawmill
```

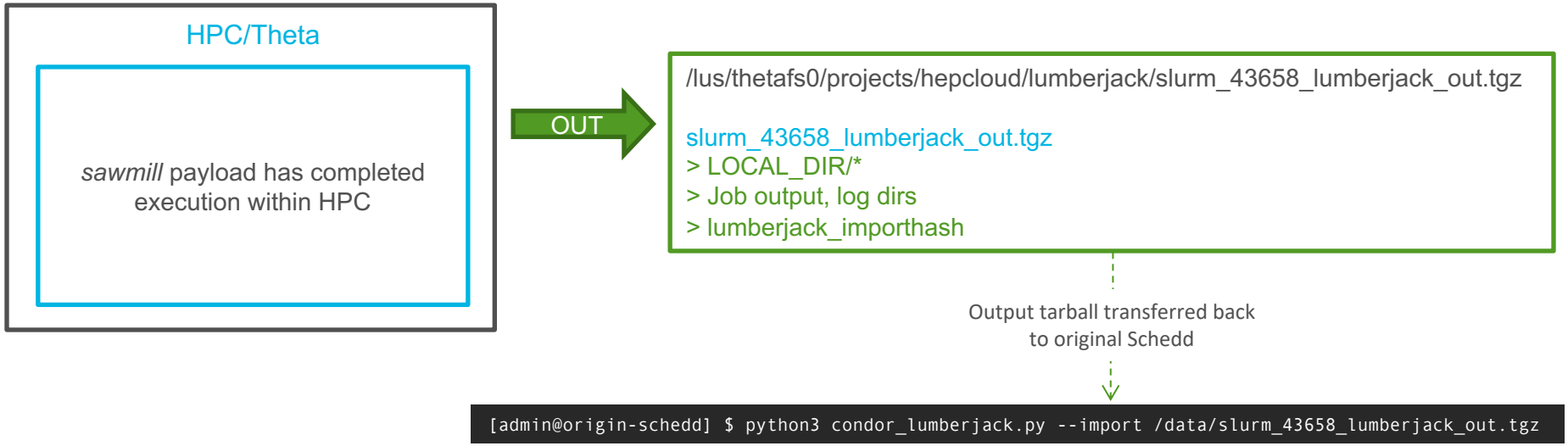
Singularity container (HTCondor)




’Sawmill’: submitting HTCondor pools as HPC jobs:

- Custom HPC job runs containerized HTCondor central manager (COLLECTOR, NEGOTIATOR, SCHEDD) and *n* startd instances filling resources allocated to the job.
- Startd instances are tailored specifically for the job(s) type and requirements as known from exported queue file.
- Provisions ancillary services (CVMFS, frontier squid, monitoring) for jobs.
- Custom entrypoint script monitors for internal queue status, once all jobs have completed, HPC job exits.
- Performs stage out.

# 4-6. Output collection, importing results into original schedd




Final steps for stageout:

- 'sawmill' performs stage out into a tarball containing job logs, HTCondor container logs, job output files and a 'translation' hash table for path matching.
- Hash table automatically interpreted by condor_lumberjack.py import command
- Places output files and dirs back, exported jobs marked C, done!

# Lumberjack - Results

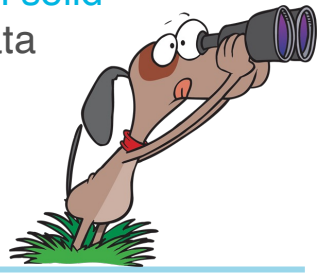
- Provides a unique opportunity for on-the-fly partial or total Schedd queue job processing on any remote system capable of running containerized payloads. From HPCs to public clouds and Kubernetes, possibilities are endless!
- Integrating with Workload Management Systems is complex as many moving parts in a submit point increase the work required to replicate file paths and job states in a containerized environment.
- An ideal solution for many use cases that can benefit from getting ‘chunks’ out of their HTCondor queues and running them instantly on an external system that supports containers.

# Final summary

| Split-starter | | | | Lumberjack |
|---|---|---|---|---|
| • Not different from what a typical HTC job would do. Split/starter joins the pool as a regular worker node startd. |  | Performance |  | • Improved performance. Entire workflows waiting for resources in the queue are easily transported and run in HPC/remote Schedd asap. |
| • User needs to have important knowledge of the code and the HPC machine specifics to operate. |  | Usability |  | • User does not have to worry about internals as the system is designed to handle the entire Lumberjack workflow. |
| • Depends entirely on Filesystem, from our experience, this is a fragile point for some HPCs.<br>• Proxy startds become unstable as the Edge node load increases with each split-starter instance. |  | Reliability |  | • If all components are properly configured before going into the HPC, the system is reliable to produce finished jobs.<br>• Implements periodic monitoring checks to avoid major disruptions and retries. |
| • The split-starter is tailored to the HPC batch system. Restrictions and challenges inherent to the HPC, specific networking and storage topologies. |  | Portability |  | • Needs an almost ‘perfect match’ of the original Schedd which can be tricky to achieve within typically restricted containers and complex Schedd setups. |
| • Error handling and prevention can be greatly improved. Prone to synchronization issues.<br>• No checkpointing made by default. |  | System resilience |  | • Exporting and re-importing queues is still manual. When errors occur, unchecked operations on SPOOL directories can disrupt the original schedd functionality. |

# Conclusions and looking ahead:

- Defining clear integration paths for split/starter or lumberjack with Decision Engine is an important next step as both approaches still require human interaction.
- Re-factored Decision Engine is in the works, we are expecting to integrate the split starter and possibly lumberjack after new release goes into production (deployment to start during summer 2022)
- Two different approaches implemented for a single HPC. The project will need to further evaluate realistic production scenarios and expand R&D effort to other machines and batch system flavors.
- Collaborating and dedicating effort to R&D provides powerful knowledge and insights, leads to improvements on existing systems and enables innovation and space for creative solutions.
- As the Exascale era comes closer, it's important for the project to be equipped with solid technical and operational capabilities to tackle the unknown territory of HL-LHC data coming into our system.

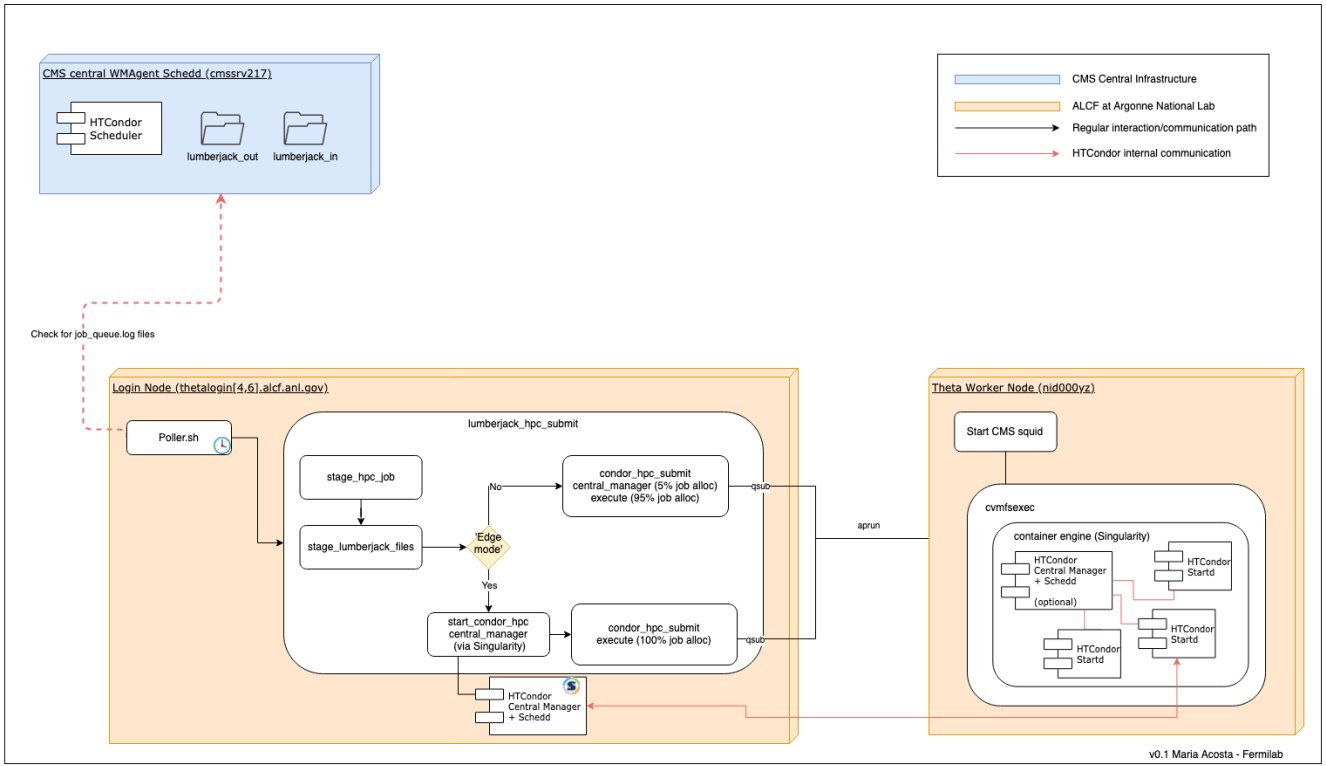


# Thanks ☺ Questions?

Maria Acosta - SCS/CSI
macosta@fnal.gov
@macosta on Slack

# Backup Slides

# Current component diagram (Lumberjack)



# A real Lumberjack run (1):

## Startup and job queue inspection:

```
└─[$] more 547878.output
08-09-2021_00-09-15 - nid00000 (1) -  Running a self-contained HTCondor pool with role: CentralManager
I am a central manager, writing my info at /lus/theta-fs0/projects/HEPCloud-FNAL/job_area/cobalt-cms-11488/90_central_mgr_info
08-09-2021_00-09-15 - nid00000 (1) -  Cleaning up possible leftovers from previous jobs
Called cleanup function
08-09-2021_00-09-15 - nid00000 (1) -  Deploying and starting local squid
08-09-2021_00-09-15 - nid00000 (1) -  Setting relevant environment variables
08-09-2021_00-09-15 - nid00000 (1) -  Configuring CVMFS, if successful, start HTCondor
08-09-2021_00-09-15 - nid00000 (1) -  Launching CVMFSexec and minicondor inside Singularity - cobalt-cms-11488.nid00000
CernVM-FS: loading Fuse module... done
CernVM-FS: mounted cvmfs on /local/scratch/uscms/cobalt-cms-11488/cvmfsexec/dist/cvmfs/config-osg.opensciencegrid.org
CernVM-FS: loading Fuse module... done
CernVM-FS: mounted cvmfs on /local/scratch/uscms/cobalt-cms-11488/cvmfsexec/dist/cvmfs/cms.cern.ch
CernVM-FS: loading Fuse module... done
CernVM-FS: mounted cvmfs on /local/scratch/uscms/cobalt-cms-11488/cvmfsexec/dist/cvmfs/unpacked.cern.ch
CernVM-FS: loading Fuse module... done
CernVM-FS: mounted cvmfs on /local/scratch/uscms/cobalt-cms-11488/cvmfsexec/dist/cvmfs/oasis.opensciencegrid.org
Looks like I'll be running a Lumberjack Schedd, inspecting my job_queue.log
```

# A real Lumberjack run (2):

Containerized HTCondor pool and Schedd startup on HPC worker, condor_q/condor_status verification:

```
-- Schedd: sched_cobalt-cms-11488@theta.alcf.anl.gov : <10.128.0.1:9618?... @ 09/08/21 00:10:49
Total for query: 50 jobs; 0 completed, 0 removed, 50 idle, 0 running, 0 held, 0 suspended
Total for all users: 50 jobs; 0 completed, 0 removed, 50 idle, 0 running, 0 held, 0 suspended

-------------
MyType              TargetType          Name

Submitter           None                cmsdataops@nid00000
Collector           None                coll_cobalt-cms-11488@theta.alcf.anl.gov
Negotiator          None                macosta@nid00000
DaemonMaster        None                master_cobalt-cms-11488@theta.alcf.anl.go
Scheduler           None                sched_cobalt-cms-11488@theta.alcf.anl.gov
Accounting          none                <none>
```

# A real Lumberjack run (3):

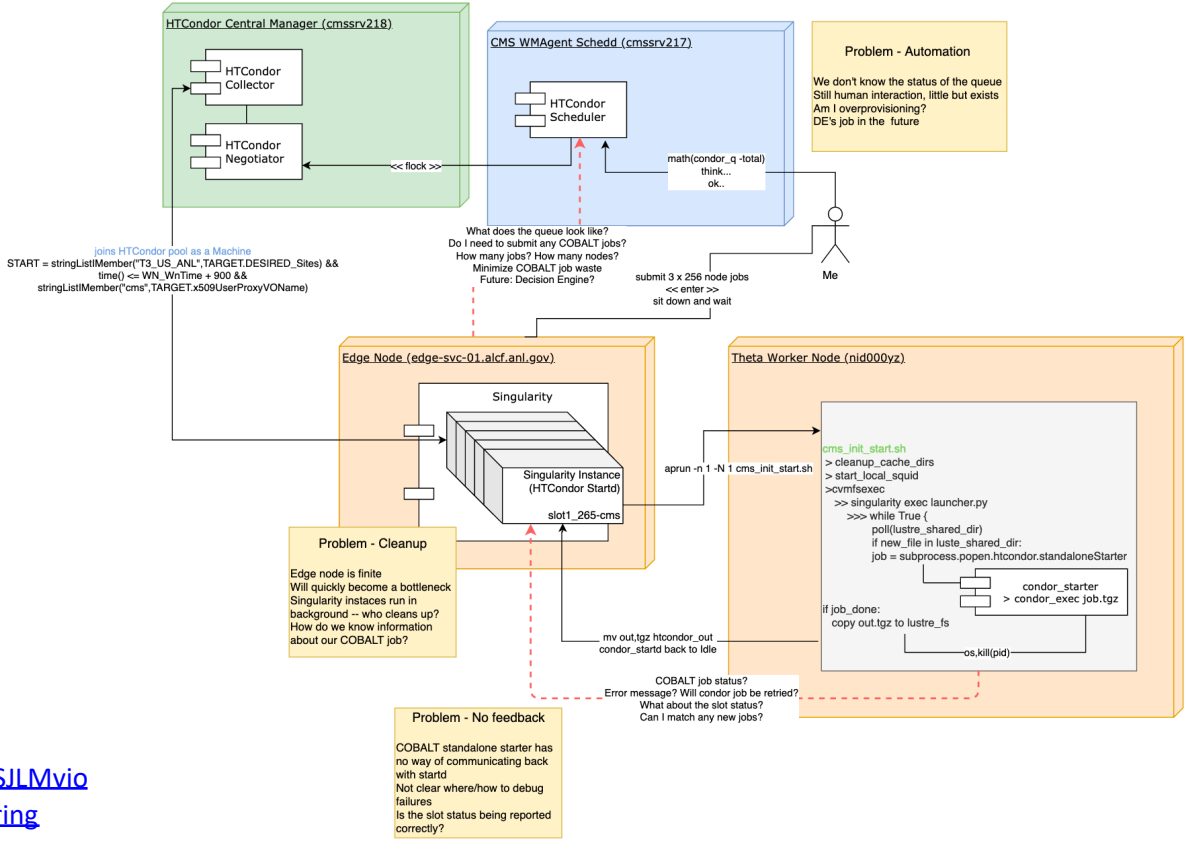
## Job stats periodic report:

```
------------
Total for all users: 50 jobs
------------
Job stats:
 0 completed completed
 0 removed removed
 50 idle idle
 0 running running
 0 held held
 0 suspended suspended
------------
0
50
------------
Job stats:
 0 completed completed
 0 removed removed
 0 idle idle
 50 running running
 0 held held
 0 suspended suspended
------------
```

## All jobs completed, interrupt HPC job, exit:

```
------------
Job stats:
 50 completed completed
 0 removed removed
 0 idle idle
 0 running running
 0 held held
 0 suspended suspended
------------
50
50
We are done with this Lumberjack queue, exiting peacefully
exit 0
Application 24458636 exit signals: Terminated
Application 24458636 resources: utime ~8s, stime ~8s, Rss ~14840, inblocks ~83066, outblocks ~233856
```

# Current component diagram (split/starter)

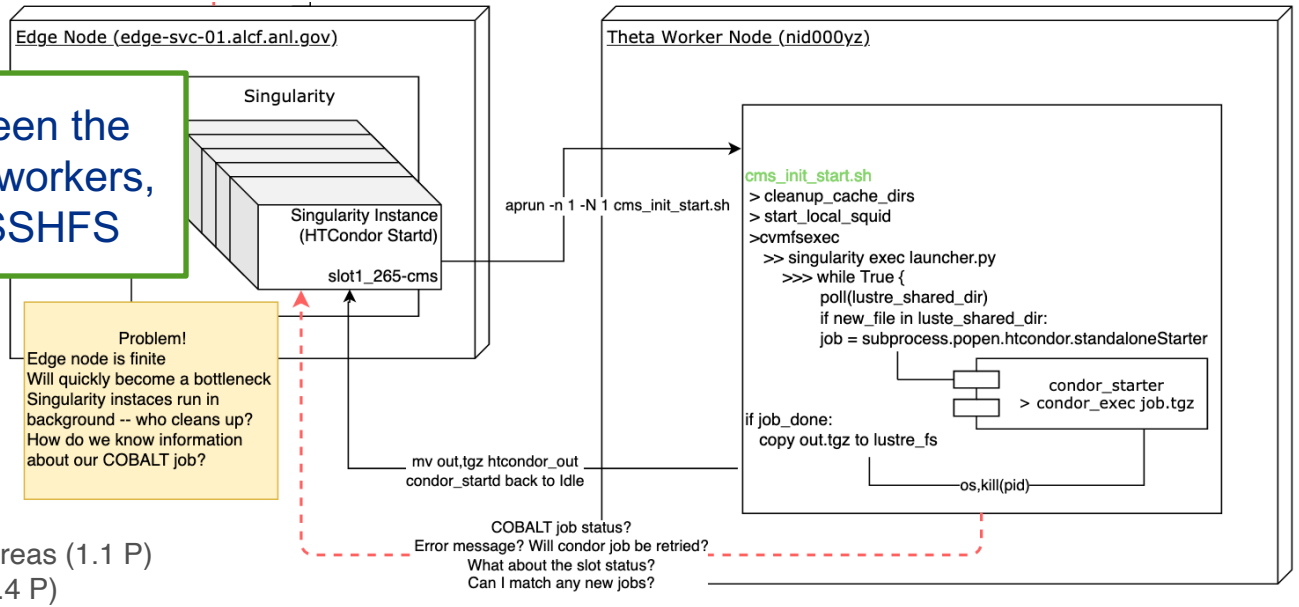



Original, more detailed diagram:
https://drive.google.com/file/d/1YbfxLSJLMviotmao3KTZ2FFbfz64Kutb/view?usp=sharing

# HTCondor split-starter – Final implementation

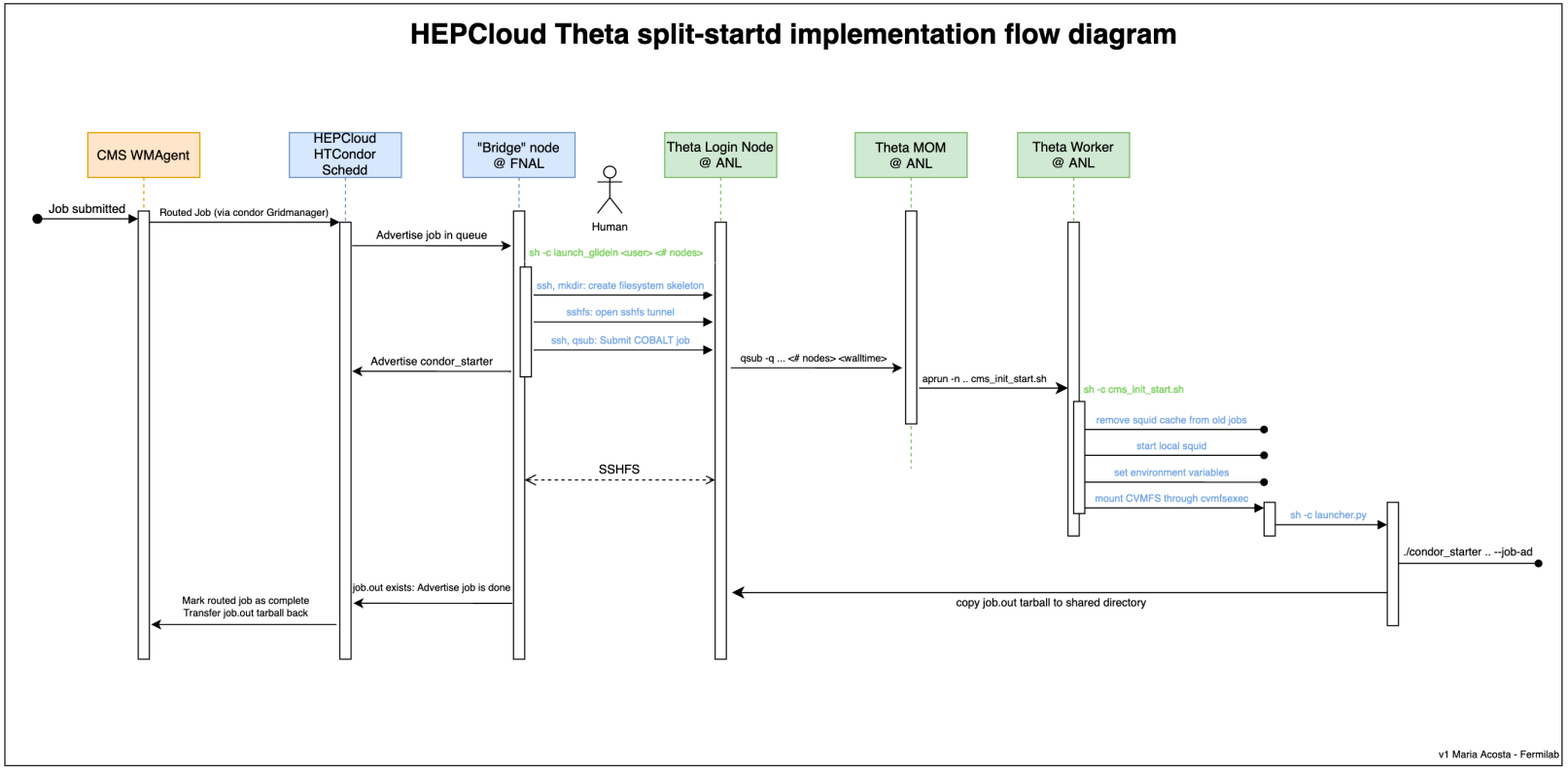
HEPCloud project requested the allocation of a Machine connected to the internal network at ALCF and with inbound and outbound connection to the outside world.

✓ Shared storage between the Edge machine and HPC workers, removed the need for SSHFS




edge-svc-01.alcf.anl.gov
RHEL 7.9 (Maipo)
16 Intel Xeon Gold CPUs @ 3.20GHz
100Gb RAM
120 GB local disk
Shared Filesystems mounted:
/gpfs/mira-home (GPFS) for User home areas (1.1 P)
/lus/theta-fs0 (Lustre) for Project areas (1.4 P)
No root access or permissions for installing software.
Singularity available.

# Flowchart diagram of the setup/algorithm



# Automation: Poller.sh

- A rather rudimentary effort at automating some of this
- Cron job runs under my user ‘macosta’ on edge-node every 00 minute of every hour

```
>> cleans up Stale HTCondor Singularity startd instances
    i.e from finished/exited/killed COBALT jobs
>> Queries the condor_schedd for Idle jobs matching our Startd ‘START’ expression:
START = stringListIMember("T3_US_ANL",TARGET.DESIRED_Sites) && \
                time() <= WN_WnTime + 900 && \
                stringListIMember("cms",TARGET.x509UserProxyVOName)
If condor_q has Idle jobs ’WE’ could run:
    Check my COBALT queue, am I waiting for resources already?
        $(qstat –u myusername)
       No?. submit resource requests (COBALT jobs)
       Yes?. ok. Keep waiting
    OK, I’m done for now. Will come back in an hour.
    Return 0
```

✓ No more “pushing the button”,
✓ No more complicated ** math **
✓ ✓ Just keep a close eye @ cronjob output

# Resources

- https://www.alcf.anl.gov/support-center/theta/theta-thetagpu-overview
- https://indico.cern.ch/event/957688/contributions/4058330/attachments/2123148/3573944/RD_on_Connecting_Centers_with_limited_no_outbound_connectivity_OC_week_Oct_2020_1.pdf
- https://indico.cern.ch/event/957688/contributions/4058338/attachments/2123212/3574056/2020_CMSOC_KITHPC.pdf
- https://indico.cern.ch/event/957688/contributions/4058325/attachments/2123167/3573982/20201014_PIC_HTCondor_BSC.pdf
- https://indico.cern.ch/event/936993/contributions