# HPC backfill with HTCondor at CERN

Pablo Llopis Sanmillán, Nils Høimyr, Ben Jones, Luis Fernández Álvarez, Laurence Field

*HTCondor User Workshop Autumn 2020*

## Agenda / Objectives

1. Give a brief intro to why we want to backfill
2. Show how we're backfilling HPC resources with Grid jobs
3. Demonstrate with practical examples how to correctly map Grid job attributes to corresponding Slurm parameters. We'll get down to the nitty-gritty in the code.

## (Very brief) Overview of CERN Computing Infrastructure

- High Throughput Computing (powered by HTCondor)
- High Performance Computing (powered by Slurm)
- Volunteer Computing (powered by BOINC)
- All of these run on OpenStack

## HTCondor Batch Service

- Provides compute capacity to WLCG Tier-0
- Around 250K cores in 2 pools.
- One set of CEs for Grid jobs, another set of schedds for local CERN submissions
- No parallel universe submissions
- Lots of workernodes, very heterogeneous

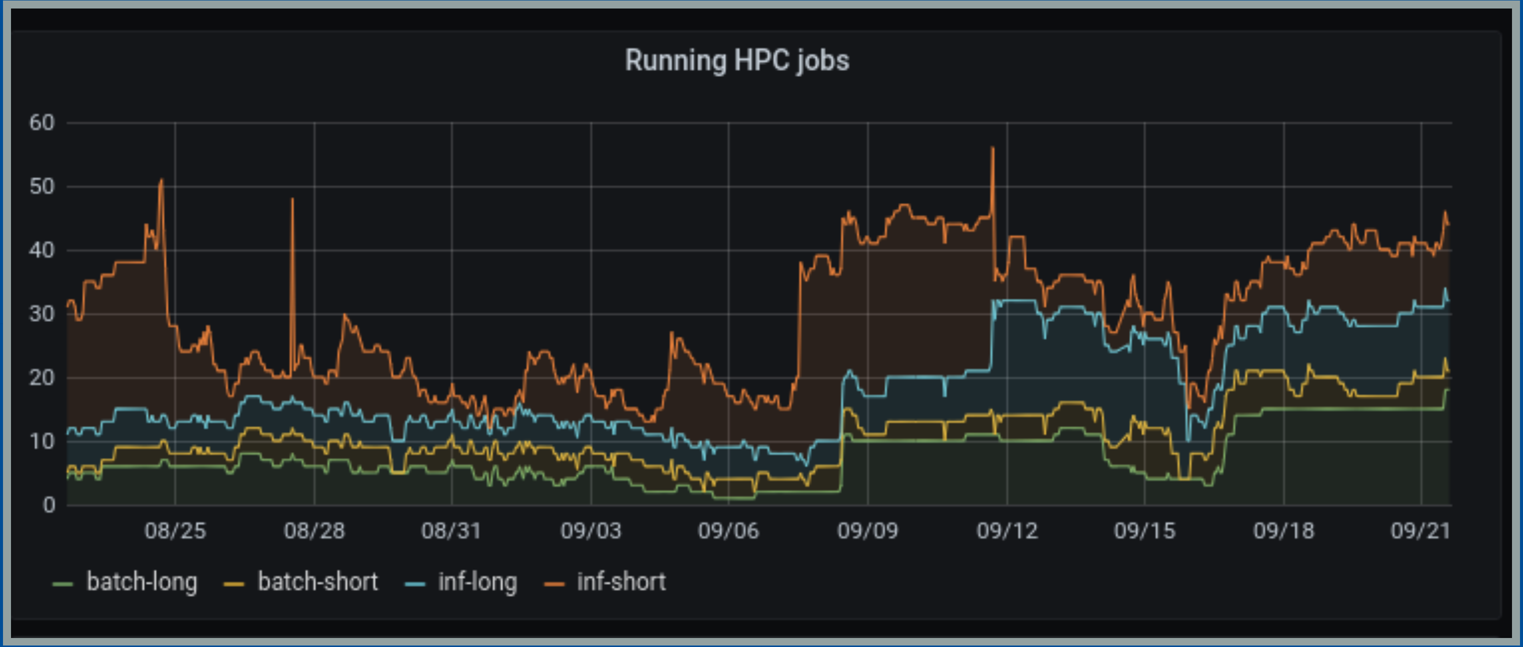
## Slurm HPC Service

- Supports Accelerator and Technology Sector and the Theory Department at CERN
- Four distinct clusters with low-latency interconnects
- Runs simulations (mostly MPI jobs) that won’t fit on a single node

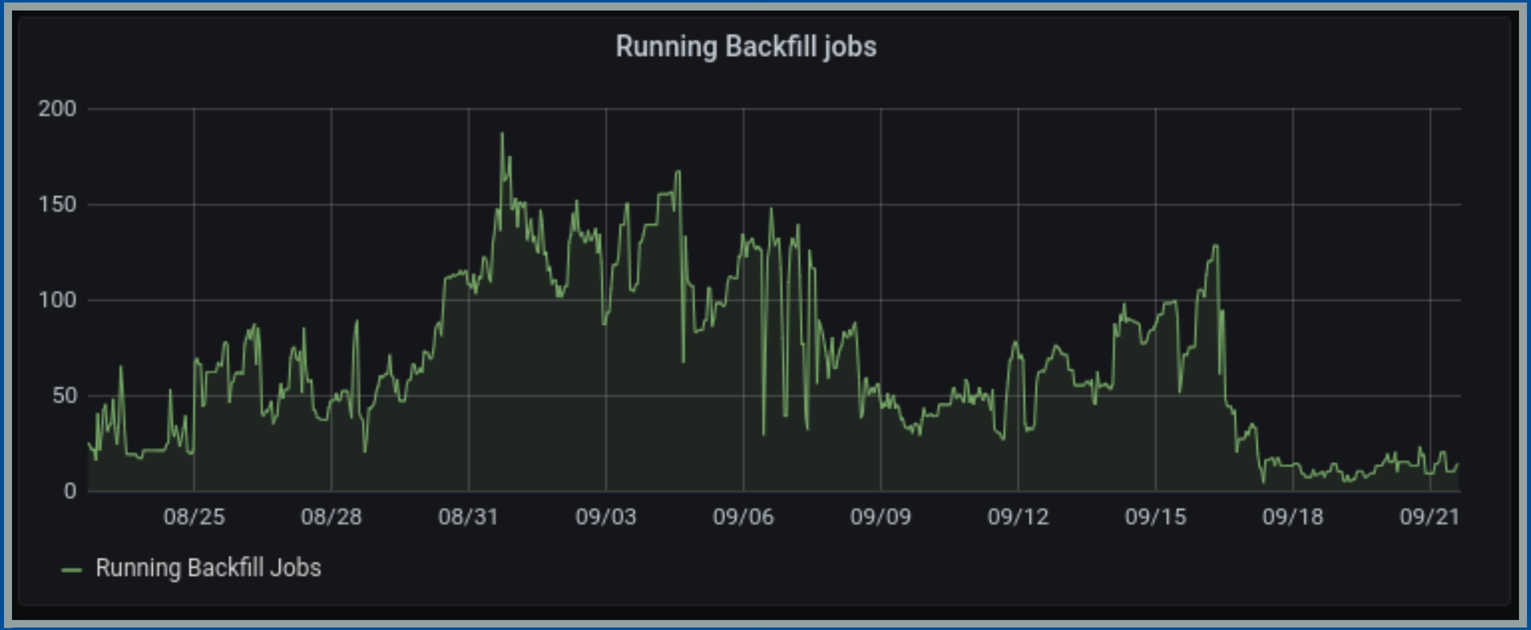
## Motivation: Maximise HPC resource utilisation

- HPC clusters are busy, but rarely to a 100%
  - Even if there are queued jobs, they may not be able to start due to size or time requirements.
  - Also some legitimate low usage periods every now and then
  - There’s always potential to fill the idle resources.

# Backfill opportunities



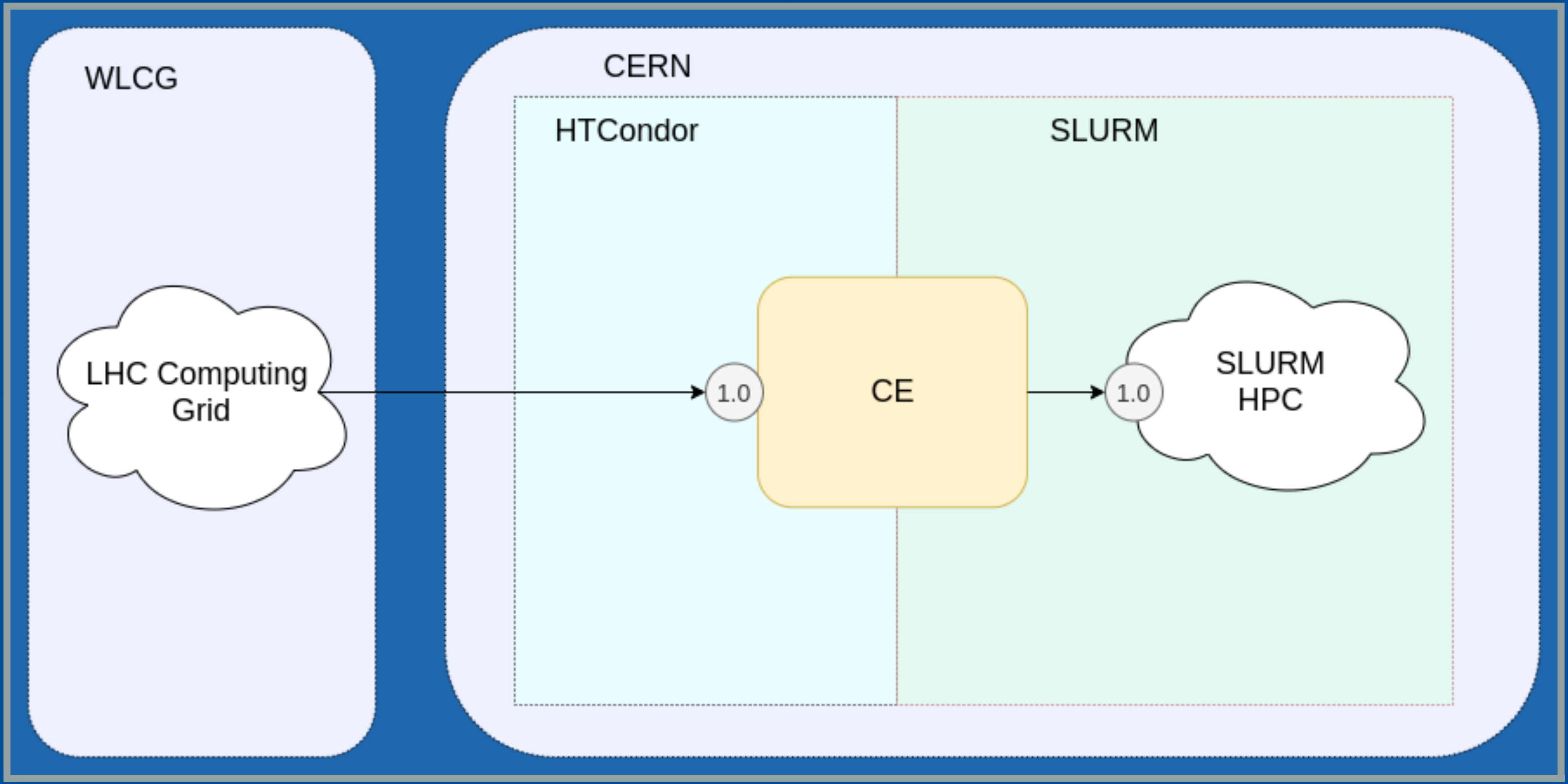
# Backfill in action



## Objective: HTCondor-C as a gateway to Slurm

- We setup a CE that will be dedicated to backfilling the Slurm HPC resources
- It's a very good fit, as Grid jobs are plentiful and are all single-node. Guaranteed to be able to "*fill in the holes*" of unused HPC resources.

# HTCondor-C - Slurm gateway: bird's eye view



# Submission works like any of our other CEs

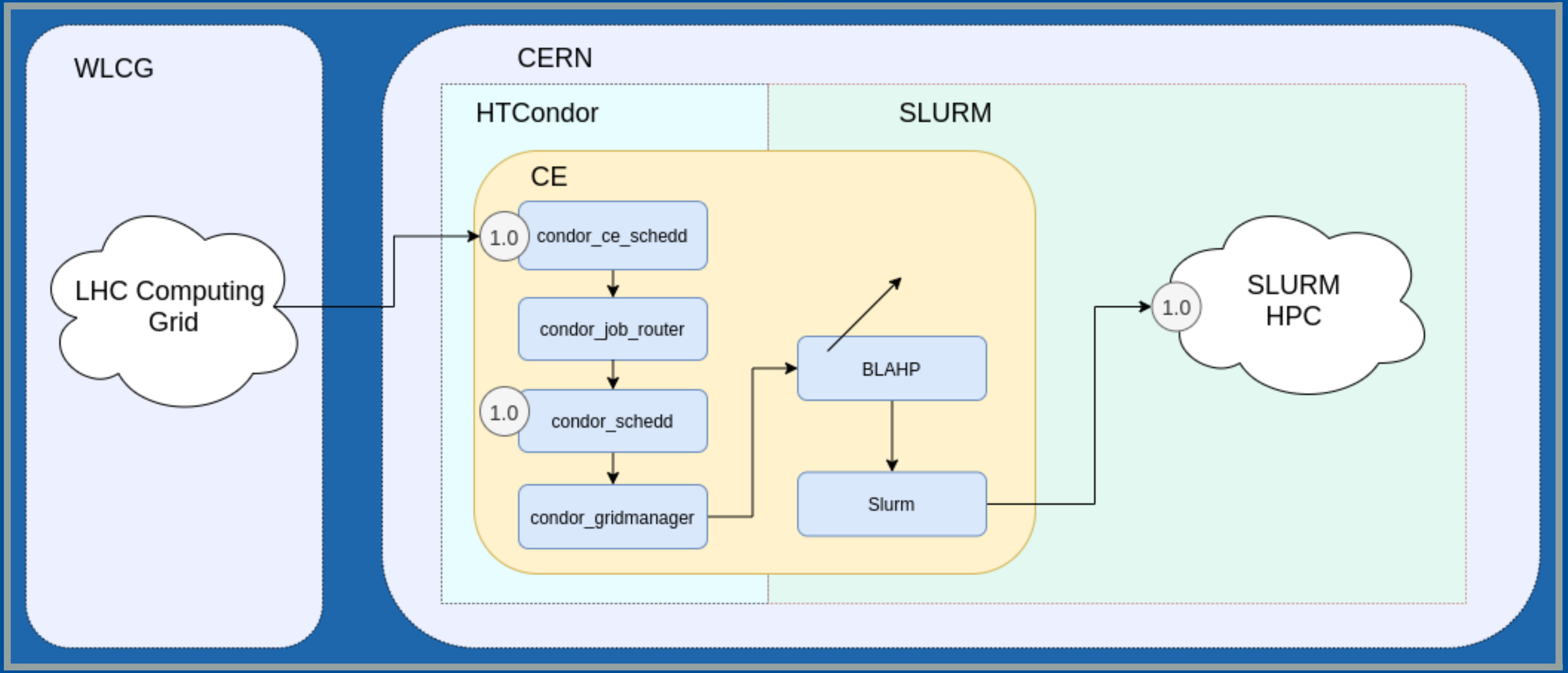
## submission

```
voms-proxy-init -voms=myteam
export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=GSI
condor_submit -pool cehost:9619 -remote cehost  -spool gridjob.sub
```

## gridjob.sub

```
universe                 = vanilla
use_x509userproxy        = true
executable               = myjob.sh
arguments                = $(ClusterId)$(ProcId)
output                   = $(ClusterId).$(ProcId).myjob.out
error                    = $(ClusterId).$(ProcId).myjob.err
log                      = $(ClusterId).$(ProcId).myjob.log
should_transfer_files    = YES
WhenToTransferOutput     = ON_EXIT
queue
```

# HTCondor-C - Slurm gateway detailed view



# Inside the CE - JobRouter transform

- CE condor_schedd gets the job and it is processed by the JobRouter
- JobRouter applies Transforms to create the Grid job (in the local condor_schedd)
  - Note the GridResource line

```
JOB_ROUTER_ENTRIES @=jre
    [
      MaxIdleJobs = 200;
      MaxJobs = 800;
      name = "Slurm";
      set_PeriodicRemove = JobStatus == 5 ||
              (JobStatus == 1 && (CurrentTime - QDate) > 3600*6);
      set_requirements = true;
      GridResource = "batch slurm";
    ]
    @jre
```

## Inside the CE - GridManager launches Grid job

- GridResource attribute will determine how the condor_gridmanager launches the job
- GridManager launches `$(condor_config_val BATCH_GAHP)`
- In practice, ends up calling scripts that do job submission/cancel /status calls to the corresponding batch system.

```
$ condor_config_val BATCH_GAHP > /usr/libexec/condor/glite/bin/batch_gahp
```

# Caveats...

- The Slurm integration (as of today) assumes that SPOOL must be a shared filesystem between submitter and workernodes.
- Some job attributes don’t work with Slurm out of the box, such as multi-core, time limits, etc.
- Difficult to tell which Slurm jobid corresponds to which Condor jobid.

## Nitty-Gritty

- How are the Condor job attributes and requirements translated into Slurm?
- What’s this GAHP/BLAH thing?
- How can I run multi-core jobs in Slurm?
- How do I get Slurm submissions with specific options/requirements?

## Batch GAHP vs BLAH

**GAHP** stands for **G**rid **A**SCII **H**elper **P**rotocol. It's an abstraction so that clients can talk a simple protocol with a server that implements grid/cloud services. Several GAHP protocols and implementations exist.

Condor's Batch GAHP consists of the **BLAHP** protocol/implementation. The upstream project is called **BLAH**. Also referred to as "the **blahp**".

Implements scripts for *submit/hold/resume/status/cancel* for various batch systems, **Slurm** included.

## Condor BLAH vs upstream BLAH

Condor runs a forked version of BLAH.

Current Condor releases have renamed it to `batch_gahp`.

Condor and upstream have evolved in different (incompatible) ways.

There's an ongoing effort to merge codebases, so this will get simplified in the future.

## Translation of HTCondor to Slurm parameters

- HTCondor uses classad attributes to describe job properties
- Slurm uses #SBATCH lines in a job script to embed desired job properties.

Job properties: Number of cores, time limit, queue name, etc…

The blahp code is the main part handling the translation of Condor job attributes to Slurm #SBATCH directives.

# Translation of HTCondor to Slurm parameters (2)

- Parameters to be translated need to be supported by the blahp.
- Depending on your Condor version, this support will vary.
- The blahp server translates supported job **classad attributes** into **parameters** passed on to a **blah script**.
- In some Condor versions, the blahp server (i.e. `batch_gahp`) translates parameters that are unused and can be leveraged by patching the blahp scripts.
- This is not thoroughly documented. Use the code, Luke.

## Tracking parameters through code

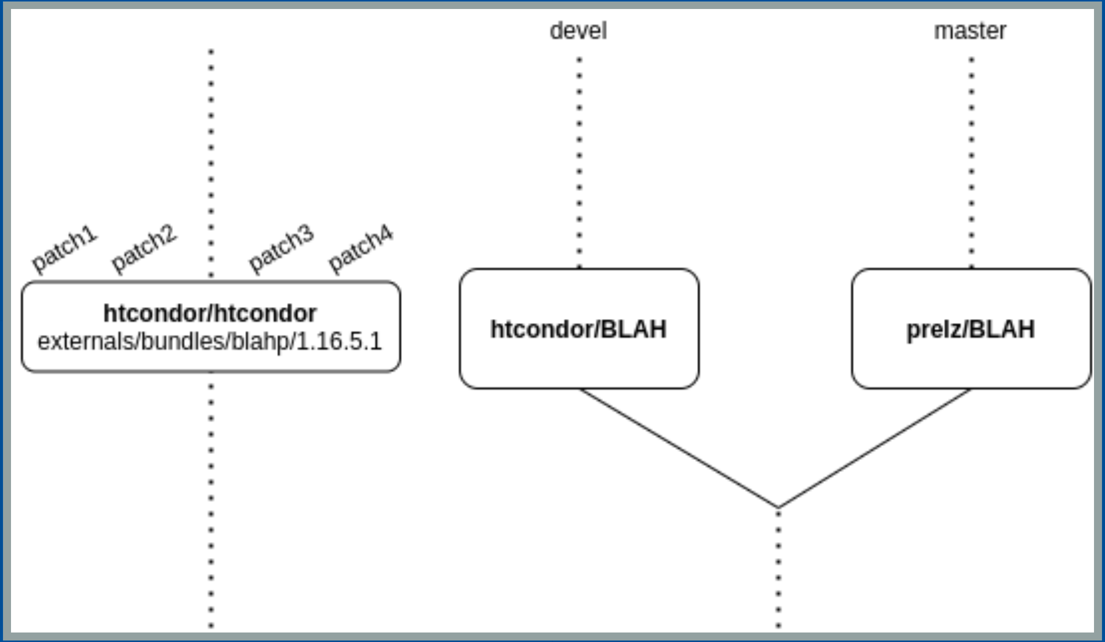
**Tracking attributes from condor job classads to the final Slurm submit file involves mostly three different files:**

1. blahp `server.c`. Check the *server.c* `cmd_submit_job` function. Translates condor attributes into blahp script argument options.
2. blahp script `blah_common_submit_functions.sh` function `bls_parse_submit_options` handles argument options and sets corresponding variables in script.
3. blahp submit script `slurm_submit.sh` uses values set in variables to write the final Slurm submit script.

## Note

Tracking blahp code for specific Condor versions isn’t very convenient. While the devel BLAH code in Condor can be conveniently browsed at https://github.com/htcondor/BLAH, the code corresponding to a specific Condor version can be tracked through the different branches at the externals directory that consists of a set of patches to an older upstream BLAH (not found on GitHub, but here).

# Tracking parameters through code (2)



Therefore, when tracking the code for parameters, we will have to browse through either

1. Code in blahp-1.16.5.1's server.c
2. The patches in externals/bundles/blahp
3. For the slurm blah scripts, you can read them: at either
   3.1) In your local installation, under `$(GLITE_LOCATION)/bin`.
   3.2) Upstream in the gridmanager

## Practical example: Selecting a Slurm partition

Let’s say we want Condor jobs to be able to select a Slurm **partition**.

Let’s also assume we want to use the latest Condor stable release (*8.8*).

To understand how parameter translation works for this version, we can navigate through the code in two directions:

1. Either start tracking from the slurm_submit.sh which sets the parameters, working our way backwards until we see which classad attribute is mapped (or not) to this parameter.
2. Start at the gridmanager and find which classad ends up mapping to which Slurm parameter in blah’s slurm_submit.sh.

Usually find 1. easier as there is only one way to do select job attributes in Slurm.

## Tracking steps:

1. In the slurm_submit.sh Slurm's partition parameter (`#SBATCH -p`) is being given the value `$bls_opt_queue`.
2. Which is being set via the blah script parameter `-q` in blah_common_submit_functions.sh. This parameter is already there in the original 1.16.5.1 source code that we mentioned.
3. Which is mapped by the `cmd_submit_job` function in blahp-1.16.5.1 server.c:

```
(set_cmd_string_option(&command, cad, "Queue",       "-q", NO_QUOTE)
```

**Conclusion:**

The `Queue` job classad attribute will map to the Slurm `#SBATCH -p` parameter.

## Practical example: Runtime limit

Same assumptions as before: Condor 8.8 and we'll track parameters backwards starting at slurm_submit.sh.

1. The slurm_submit.sh sets `#SBATCH -t` to `$((bls_opt_runtime / 60))`.
2. Which corresponds to the `-t` blah script parameter in *blah_common_submit_functions.sh*.
3. Which is mapped by the blahp server via the `BatchRuntime` classad.

Note that this code is not in the original 1.16.5.1 sources, but is contained in a patch instead.

## Practical example: multi-core jobs

- Grid jobs traditionally use the `xcount` job classad attribute for the number of desired cpu cores.
- The Slurm equivalent to allocate cores is `--cpus-per-task`.

Let’s assume the latest Condor stable release (*8.8*)

In the slurm_submit.sh script there is no trace of using anything like `--cpus-per-task`.

But wait! There is mention of number of nodes:

```
# Simple support for multi-cpu attributes
if [[ $bls_opt_mpinodes -gt 1 ]] ; then
  echo "#SBATCH -N $bls_opt_mpinodes" >> $bls_tmp_file
fi
```

In Slurm `-N` is multi-node, not multi-cpu or multi-core. We could re-use this by rewriting it:

```
if [[ $bls_opt_mpinodes -gt 1 ]] ; then
  echo "#SBATCH --nodes=1" >> $bls_tmp_file
  echo "#SBATCH --ntasks=1" >> $bls_tmp_file
  echo "#SBATCH --cpus-per-task=$bls_opt_mpinodes" >> $bls_tmp_file
fi
```

*(Note: this is actually how it will work in future releases)*

But how is `$bls_opt_mpinodes` set? In the argument parsing done in `blah_common_submit_functions.sh` we can see that it corresponds to the blah parameter `-n`:

```
n) bls_opt_mpinodes="$OPTARG";;
```

And in the blahp server.c, it's the classad attribute `NodeNumber` which maps to option `-n`:

```
(set_cmd_int_option   (&command, cad, "NodeNumber", "-n", INT_NOQUOTE) ..
```

*(Except for the changes described for slurm_submit.sh, this is all found in the original blahp-1.16.5.1 sources.)*

Therefore: All we need is to make sure that our jobs use `NodeNumber`. But we had established that we'd use the `xcount` attribute, I hear you say?

We can use the *JobRouter transforms* to set `NodeNumber` to whatever `xcount` is set.

Actually, it looks like this transform is already included! (In our `JOB_ROUTER_DEFAULTS_GENERATED`)

Therefore we "only" need to patch the `slurm_submit.sh` (if using <= 8.8) for this to work.

## Job Parameters Summary (docs PR)

| htcondor 8.8 | htcondor 8.9 | Slurm |
|---|---|---|
| `RequestMemory` | `RequestMemory` | `--mem` |
| `BatchRuntime` | `BatchRuntime` | `--time` |
| `BatchProject` | `BatchProject` | `--account` |
| *Unsupported*[1] | *Unsupported*[1:1] | `--cpus-per-task` |
| `Queue` | `Queue` | `--partition` |
| *Unsupported*[1:2] | `Queue`[2] | `--clusters` |

1. Usable via classad `NodeNumber` by patching *slurm_submit.sh*. ↩ ↩ ↩

2. Queue parameter syntax: “`queue@cluster`” ↩

## Additional tips

In addition, to add *constant* sbatch lines or any other preamble that you need to include in *every* slurm submit script, use `slurm_local_submit_attributes.sh`.

This script will be sourced and its output redirected into the slurm submit script, so echo the contents you wish to add.

e.g.

```
echo “#SBATCH --time=1-0”
echo “#SBATCH --partition=backfill”
```

## Additional tips (2)

You can track which Grid job maps to which Vanilla job using:

- `RoutedToJobId` in the Vanilla job (CE schedd)
- `RoutedFromJobId` in the Grid job (local schedd)

There is no built-in mapping to the Slurm jobid. You can add the following towards the end of slurm_submit.sh to map CE schedd's jobid to Slurm jobid:

```
logger "Running SLURM job $jobID in $PWD"
```

## Additional tips (3)

On the Slurm side, we don’t want to increase queueing time for HPC users: Partition-based preemption.

```
PreemptType=preempt/partition_prio
PreemptMode=CANCEL
```

Our `backfill` partition encompasses all other partitions, with the following options:

```
PriorityTier=0 GraceTime=300
```

Or `priority/multifactor` config has a big value for `PriorityWeightPartition` to ensure HPC job priority.

## Questions?