HTCondor for Non-Humans

Todd L Miller
Center for High-Throughput Computing

Slicers and Droids

Monitoring

- REST API
- event log
- condor_adstash

Control

- Command-line interface
- Python bindings

The read-only REST API

- REST APIs are a common Web standard, even usable from bash.
- Not in the HTCondor manual; see the GitHub page.
- Provided by a WSGI application using the HTCondor Python bindings.
- Returns JSON blobs about the job queue, history, config, and machine status.
- Not currently officially supported.
  - Let us know if you end up using it!

- See recorded talk from HTCondorWeek 2020 for more details.

Event Logs (A New Hope)

```
040 (1295.000.000) 2022-05-03 12:01:23 Started transferring input files
     Transferring to host: <1.2.3.4:42717…>
…
040 (1295.000.000) 2022-05-03 12:01:23 Finished transferring input files
…
001 (1295.000.000) 2022-05-03 12:01:24 Job executing on host: <1.2.3.4:42717?…>
```

- Set `log = file.log` in your job descriptions.
- You can use `condor_userlog` to generate interesting summaries.
- You can use `condor_wait` instead of polling on job completion.
- Use `condor_watch_q`, not `watch condor_q`.
  - No schedd load!
  - Built on the Python `JobEventLog` API, which you could also use…

Global Event Logs (The Administrator Strikes Back)

- Set `EVENT_LOG = $(LOG)/GlobalEventLog` in your HTCondor config.
  - Creates a single file to which the events about all jobs in a given schedd are written.
- Can add job ClassAd attributes to global event log events by setting `EVENT_LOG_JOB_AD_INFORMATION_ATTRS`.
  - Helps you look for jobs or events of particular interest, e.g., GPU requests.
  - Partially obsoleted by execute event attributes in 10.6
- The global event log can also be written in JSON:
  `EVENT_LOG_FORMAT_OPTIONS = JSON, UTC_TIMESTAMP`
  - Can ease interoperation with other monitoring systems or tools.

condor_adstash

- Forwards *condor_schedd* and/or *condor_startd* job histories to ElasticSearch.
  - Can also write the JSON results to a file, instead.
  - Keeps track of up to where in the history it's reported.
- See the man page.
- (Also look up *condor_gangliad* if you've already got Ganglia set up.)

schedd and startd cron

- The output of a daemon ClassAd hook modifies that daemon's ClassAd.
  - Can be run once at a start-up, at each reconfig, periodically, or continuously.
- Useful for running health checks on a machines, e.g., HAS_CVMFS might be tested from time to time and be important for match-making.
- Can help with monitoring or accounting on a schedd.
  - Modifying the schedd's ad is less important, but you don't have to have a separately-configured cron job, and you can select between killing a job that's still running at the start of its next period (and running the new instance) or letting it finish.

Command-Line Interface

- `condor_q` options and `condor_status` options
  - `-format` for machine-readable custom output
  - `-json` for machine-readable standard output
  - `-print-format` format templates
- The `htcondor` noun-verb tool
  - job, jobset, dag, eventlog, annex

- *condor_chirp* (or `htcondor.htchirp`) and *condor_ssh_to_job*
  - job-specific monitoring
    - update job ad
    - write message to job event log (`ulog` command)
  - call-and-response computational steering?

Why did it have to be snakes?

Five Python modules in the bindings:

- `classad` - parsing, creating, and modifying ClassAds and expressions
- `htcondor` - job submission, configuration, daemon control, event logs

- `htcondor.htchirp` - jobs modifying their own ads, doing file transfer
  - pure Python implementation to simplify distribution with a job
- `htcondor.dags` - trying to make generating DAG files easier
- `htcondor.personal` - control "personal" installs, suitable for testing

The `htcondor` Python module

- Query the collector, the schedd, or other daemons (startd) directly.
- Access the HTCondor configuration system.
- Submit jobs.
- Manage credentials.
- Turn daemons on or off or reconfigure them.
- Drain startds.
- Poll or wait for job events.

- We have interactive tutorials.
- There's a recorded tutorial from last year.

Mat Told Me to Include This Slide

```
local_provider_name = htcondor.param.get('LOCAL_CREDMON_PROVIDER_NAME')
if local_provider_name is None:
    print('Local provider not named, aborting.')
    exit(1)

magic = f"LOCAL:{local_provider_name}"
binary_magic = bytes(magic, 'utf-8')

credd = htcondor.Credd()
credd.add_user_cred(htcondor.CredTypes.Kerberos, binary_magic)
```

Questions?