CENTER FOR
HIGH THROUGHPUT
COMPUTING

HTCondor

Monitoring and Controlling HTCondor with Python

John (TJ) Knoeller

HTCondor Week UK 2018

Overview

- Where we are
- ClassAds and ExprTrees
- htcondor bindings
- Examples

Design Philosophy

- ClassAds
  - everything is based on ClassAds
- Pythonic
  - Use iterators, exceptions, guards
  - ClassAds behave as much like a dict as reasonable
- Native Code
  - Uses the same library code as the HTCondor tools

Our Goal

- Complete
  - If you can do it with the command line tools, you should be able to do it with python
- Backward Compatible
  - APIs will stay stable as long as possible
  - Bad or broken APIs will be superseded, not removed
    - (queue_with_itemdata supersedes submitMany)
  - May use python DeprecationWarning

Where we are

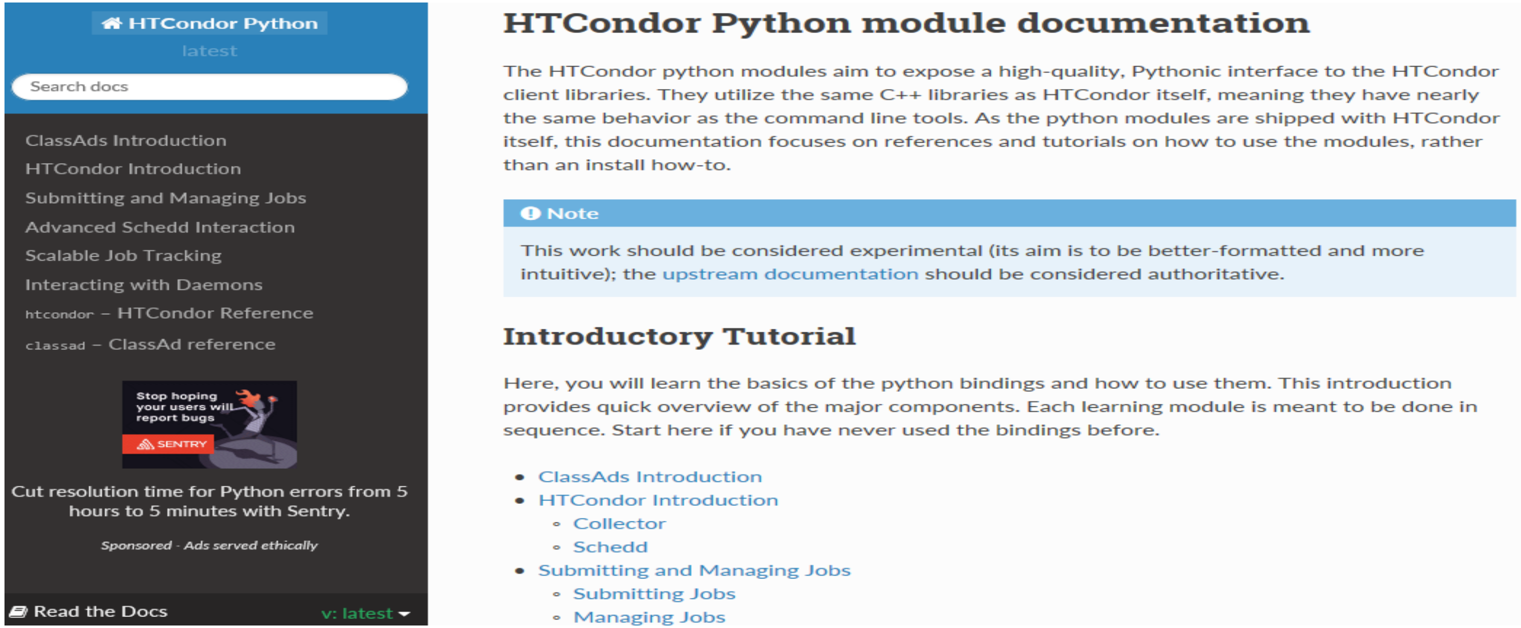
- In 8.6
  - Works with system python on Linux
  - Python2 or Python3 but not both
  - Windows is Python 2.7 only
- In 8.8
  - Plan to ship both Python2 and Python3 bindings
  - Windows x64 builds are Python 2.7 and Python 3.6

About Python3

- The way file objects are passed from python to c++ code is **completely different** in python 3
  - Doesn't work at all on Windows
- A few APIs need to change as a result
  - EventIterator, LogReader

Read the docs

› https://htcondor-python.readthedocs.io



HTCondor Python
latest

Search docs

ClassAds Introduction
HTCondor Introduction
Submitting and Managing Jobs
Advanced Schedd Interaction
Scalable Job Tracking
Interacting with Daemons
htcondor – HTCondor Reference
classad – ClassAd reference

Stop hoping your users will report bugs
SENTRY

Cut resolution time for Python errors from 5 hours to 5 minutes with Sentry.

Sponsored · Ads served ethically

Read the Docs v: latest

HTCondor Python module documentation

The HTCondor python modules aim to expose a high-quality, Pythonic interface to the HTCondor client libraries. They utilize the same C++ libraries as HTCondor itself, meaning they have nearly the same behavior as the command line tools. As the python modules are shipped with HTCondor itself, this documentation focuses on references and tutorials on how to use the modules, rather than an install how-to.

Note

This work should be considered experimental (its aim is to be better-formatted and more intuitive); the upstream documentation should be considered authoritative.

Introductory Tutorial

Here, you will learn the basics of the python bindings and how to use them. This introduction provides quick overview of the major components. Each learning module is meant to be done in sequence. Start here if you have never used the bindings before.

- ClassAds Introduction
- HTCondor Introduction
  - Collector
  - Schedd
- Submitting and Managing Jobs
  - Submitting Jobs
  - Managing Jobs

JupyterHub Tutorials

- https://hcc-anvil-175.29.unl.edu
  - Login with university credential
  - Spawns a docker instance with a private HTCondor
- Much of this talk is taken from there

ClassAds

- import classad
  - provides the ClassAd and ExprTree classes
- defines class ClassAd
  - Behaves like a dict {key : value}
- values are of type ExprTree which can be
  - a simple value (int, real, bool, string)
  - a special value (Undefined, Error)
  - an expression which evaluates to one of the above

ExprTree

- ExprTree is a ClassAd expression
  - Literals (int, real, bool, string) are automatically converted to python types
  - Undefined and Error have no python equivalent
- Evaluated lazily, only when explicitly asked
- When eval() method of ExprTree is called
  - returns a Literal (or Undefined/Error)

ExprTree examples

```
e = classad.ExprTree("1 + 4")
print "Expr %s is of type %s" % (e, type(e))
Expr 1 + 4 is of type <class 'classad.ExprTree'>

v = e.eval()
print "It evaluates to %s of type %s" % (v, type(v))
It evaluates to 5 of type <type 'long'>
```

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

Evaluate in ClassAd context

```
ad = classad.ClassAd('[a=1; b=4; tot=a+b]')
print ad['a']
print ad['tot']
print ad['tot'].eval()
1
a + b
5
print ad.eval('tot')
5
```

Expressions vs Literals

- Many classads in HTCondor have values that **can** be expressions but are **usually** literals
- HTCondor daemons almost always evaluate rather than look-up attributes in ClassAds
- You should do the same
  - Use ad['name'] if you are sure it's a literal
  - Use ad.eval('name') if you don't know

- And now, we get to the HTCondor part...

bindings for Major Daemons

› import htcondor
- htcondor.Collector()
  - query, directQuery, locate, advertise : condor_status, advertise
- htcondor.Schedd()
  - query, xquery, history : look at jobs
  - act, edit : change jobs
  - transaction, spool, retrieve : submit jobs
  - submit, submitMany : obsolete submit interface
  - negotiate, reschedule : specialized users only

bindings for Minor daemons

- htcondor.Startd()
  - drainJobs, cancelDrainJobs
- htcondor.Negotiator()
  - setPriority, setFactor, resetUsage, ...
  - getPriorities, getResourceUsage
    - (In 8.7 you can query Accounting ads from the Collector)

Submit object

- htcondor.Submit()
  - queue, queue_with_itemdata
- Wraps up a HTCondor submit file
- New capabilities in 8.7
- A whole talk on this...

HTCondor config

- htcondor.version()
  - get the HTCondor version
- htcondor.param['knob']
  - get the expanded value of the config knob
- htcondor.reload_config()
  - reread the HTCondor config files
- htcondor.RemoteParam(daemonAd)
  - query the configuration of a daemon

Log Readers

- htcondor.JobEventLog (new in 8.7.10!)
  - Iterate a job's log as a stream of JobEvent(s)
  - Supersedes htcondor.EventIterator
- htcondor.EventIterator
  - For completed jobs, iterate the job's log as a stream of events
  - Let us know if you are using EventIterator,
    - (We think no-one is using this and would like to kill it)
- htcondor.LogReader
  - Reads the schedd's job queue log file

Find a Startd

```
coll = htcondor.Collector()
startd = coll.locate(htcondor.DaemonTypes.Startd, "host")
print startd['MyAddress']
print "type is %s, size is %d" % (type(startd), len(startd))
"<127.0.0.1:64900?addrs=127.0.0.1-64900>"
type is <class 'classad.ClassAd'>, size is 6

ads = coll.query(htcondor.AdTypes.Startd, 'Machine=="host"')
print "type is %s, size is %d" % (type(ads), len(ads))
print "type is %s, size is %d" % (type(ads[0]), len(ads[0]))
type is <type 'list'>, size is 4
type is <class 'classad.ClassAd'>, size is 144
```

locate vs query

- collector.locate()
  - Returns an ad for a single daemon
  - Just enough attributes to open a socket to that daemon
- collector.query()
  - Returns a list of ads from one of the collections
  - A LOT of attributes unless you use a projection
    - always use a projection!

directQuery

- directQuery is a locate followed by a query
  - Use it to query the startd or schedd directly
    - Get more verbose statistics
- Equivalent to this

```
coll = htcondor.collector()
schedd_ad = coll.locate(htcondor.DaemonTypes.Schedd)
daemon = htcondor.collector(schedd_ad['MyAddress'])
daemon.query(htcondor.AdTypes.Schedd, statistics='ALL:2')
```

Query jobs from a Schedd

```
schedd = htcondor.Schedd() # get the local schedd
jobs = schedd.query('ClusterId=23', ['JobStatus'], limit=2)
for job in jobs : print job.__repr__()
[MyType="Job", JobStatus=5, ClusterId=23, ProcId=0]
[MyType="Job", JobStatus=5, ClusterId=23, ProcId=1]

ads = schedd.query(opts=htcondor.QueryOpts.SummaryOnly)
print(ads[0])
[
 AllUsesrIdle = 0;
 AllUsersHeld = 2;
...
```

XQuery jobs from a Schedd

```
schedd = htcondor.Schedd() # get the local schedd
attrs = ['ClusterId', 'ProcId', 'JobStatus']
for job in schedd.xquery(projection=attrs):
   print job.__repr__()
[MyType="Job", JobStatus=5, ClusterId=23, ProcId=0]
[MyType="Job", JobStatus=5, ClusterId=23, ProcId=1]
```

› xquery returns an iterator
  - This has less overhead but...
  - It's bad to walk this iterator slowly

query vs query vs xquery

- You can send 'collector' queries to a schedd, but not visa versa.
- (x)query methods on class Schedd returns jobs
- There is no xquery for the Collector. (it cannot do async replies)

So - What's Missing?

- What do we need to add?
  - Credd
  - ?

HT
CENTER FOR
HIGH THROUGHPUT
COMPUTING

HTCondor

Any Questions?