Python and HTCondor

Brian Bockelman
HTCondor Week 2016
WARNING: Too Many Slides

The end of the presentation contains a few nifty, very advanced, examples we won’t get into.
Python in HTCondor

• The python bindings aim to…
  • Provide complete, high-quality, *pythonic* access to the HTCondor ecosystem,
  • Using the existing C++ libraries from the HTCondor team,
  • Oriented around the ClassAd language.
It all starts with a ClassAd…

- ClassAd objects mimic Python dictionaries. Values can include:
  - String, floats, integers, lists, and sub-ClassAds…
ClassAd Expressions

• … And Expressions!

```python
>>> import classad
>>> classad.ExprTree("foo + 4")
foo + 4
>>> ad = classad.ClassAd()
>>> ad["bar"] = classad.ExprTree("foo + 4")
>>> ad
[ bar = foo + 4 ]
>>> ad["foo"] = 4
>>> ad["baz"] = "testing"
>>> ad
[ bar = foo + 4; foo = 4; baz = "testing" ]
>>> ad["requirements"] = classad.ExprTree("baz is " %s % \n classad.quote(\"testing\")")
>>> ad
[ bar = foo + 4; foo = 4; requirements = baz is "\"testing\"; baz = "testing" ]
>>> ad["baz"] = classad.Attribute("foo").and_(ad["bar"])
>>> ad
[ bar = foo + 4; foo = 4; requirements = baz is "\"testing\"; \n baz = foo && foo + 4 ]
>>> ad.eval("requirements")
False
```
ClassAd Detour

• I have the following ClassAd:
  
  ```
  [ ...
   Owner="bbockelm";
   x509UserProxyFirstFQAN="/cms/uscms/
   Role=prod";
   ...
  ]
  ```

• I want the accounting group set to the VO name (cms):
  
  ```
  [ ...
   x509UserProxyFirstFQAN="/cms/uscms/
   Role=prod";
   AccountingGroup="cms.bbockelm";
   ...
  ]
  ```
ClassAd Detour

Use the `regexps` function!

```python
>>> import classad
>>> ad = classad.ClassAd("""[Owner="bbockelm"; x509UserProxyFirstFQAN="/cms/Role=prod"]""")
>>> ad
[ x509UserProxyFirstFQAN = "/cms/Role=prod"; Owner = "bbockelm" ]
>>> classad.ExprTree(r'regexps("/(\w+)/", "/cms/Role=prod", "\1")
regexp("/(\w+)/", "/cms/Role=prod", "\1")
>>> classad.Function("regexps", r"/(\w+)/", "/cms/Role=prod", r"\1")
regexp("/(\w+)/", "/cms/Role=prod", "\1")
>>> ad["VO"] = _
>>> ad["AccountingGroup"] = classad.ExprTree('strcat(VO, ".", Owner)')
>>> ad.eval("AccountingGroup")
'cms.bbockelm'
```
Parsing Ads

- The `classad.parse*` methods:
  - `parseNext()`: Parse a single ad from the input stream and stop.
  - `parseOne()`: Parse the entire input stream, returning a single ClassAd (merging all ads together if possible).
  - `parseAds()`: Parse the input stream into an iterator of ClassAds.
  - Will automatically detect both old-style and new-style formatting.

```plaintext
$ cat /tmp/foo
foo = bar
bar = 1

baz = 22
$ python
Python 2.6.6 (r266:84292, Jul 22 2015, 16:47:47)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> fp = open("/tmp/foo")
>>> classad.parseNext(fp)
[ bar = 1; foo = bar ]

>>> classad.parseNext(fp)
[ baz = 22 ]
>>> classad.parseNext(fp)
StopIteration
>>> fp.seek(0)
>>> classad.parseOne(fp)
[ bar = 1; bad = 22; foo = bar ]
```
Some Python basics

• import htcondor; import classad

• Use `dir()` to list object names in a module; use `help()` to get the per-method or class help.

• `print classad.version(), htcondor.version()`

• `htcondor.param[‘COLLECTOR_HOST’]` to access parameter value of COLLECTOR_HOST.
Collector Basics

• The Collector object allows one to locate daemons, query slot status, and advertise new ClassAds.

• The object takes the network location of the collector daemon for the constructor:

  • coll = htcondor.Collector("red-condor.unl.edu")
Collector Basics

```
Last login: Sat Apr 27 11:26:34 on ttys004
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> ad = htcondor.Collector("red-condor.unl.edu")
>>> ad["MyAddress"]
'129.93.239.129:41562'
'>>> ads = htcondor.Collector("
>>> for ad in ads: print ad["Name"]
... red-gw1.unl.edu
red-gw2.unl.edu
red.unl.edu
flocking@t3.unl.edu
t3.unl.edu
```
Collector Advanced

• For many queries, pulling all attributes from the collector is expensive.

• You can specify a projection list of attributes. HTCondor will return the minimum number of attributes containing the ones you specify.

• It will always pad in a few extra.
Collector - Advanced

```python
Last login: Mon Apr 29 08:47:21 on ttys004
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> import htcondor
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> ads = coll.query(htcondor.AdTypes.Startd, 'true', ['MyAddress', 'Name', 'Cpus', 'Memory'])
>>> len(ads)
4649
>>> ads[0]
[ CurrentTime = time(); Memory = 119; MyAddress = "<172.16.1.70:38406>"; Cpus = 0; Name = "slot1@node070.red.hcc.unl.edu"; MyType = "Machine"; TargetType = "Job"
]
```
Schedd Basics
Job ClassAds

• Python uses the *job* ClassAd, not the submit file language. This is less-used and less-documented.
  
  • Macros aren’t available!

• Don’t memorize the list of attributes!

  • If you’re unsure how a line in the submit file would convert to ClassAds, run `condor_submit -dryrun`.

  • Reference of all attributes: [http://research.cs.wisc.edu/htcondor/manual/v8.5/12_Appendix_A.html](http://research.cs.wisc.edu/htcondor/manual/v8.5/12_Appendix_A.html)

• Example minimal ClassAd:

  ```
  [ Cmd="/bin/sh";
    Arguments="-c 'echo Hello world && sleep 1m'";
    Out = "/tmp/test.out"; Err = "/tmp/test.err";
    UserLog = "'/tmp/test.log" ]
  ```
Submit ClassAds

• A few submit file / ClassAds translations:
  • error / Err
  • output / Out
  • executable / Cmd
  • should_transfer_files / ShouldTransferFiles
  • transfer_input_files / TransferIn
  • transfer_output_files / TransferOut

• From macros to ClassAds:
  • Instead of: error = “test.err.$(Process)"
  • Write: Err = strcat(“test.err”, ProcID)
This Makes My Head Hurt!

- It’s fairly jarring for users of HTCondor submit files to use bare ClassAds.
- Very different format, minimal online documentation.
- In the 8.5.x series*, we will expose an API that uses the more common submit-file syntax in python.

Hopefully; possibly very early in 8.7.x
Schedd Advanced

• A few useful Schedd methods:
  • **act**: Perform some action on one or more jobs (hold, release, remove, removeX, suspend, continue).
  • **edit**: Edit one or more job ClassAds
  • **reschedule**: Have Schedd request a new negotiation cycle.
Schedd Advanced

```python
[demo@ip-10-62-61-234 ~]$ python
Python 2.6.6 (r266:84292, Dec 7 2011, 20:48:22)
[GCC 4.4.6 20110731 (Red Hat 4.4.6-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> schedd = htcondor.Schedd()
>>> jobs = schedd.query('true', ['ClusterID', 'foo'])
>>> jobs
[[ MyType = "Job"; TargetType = "Machine"; ServerTime = 1367284751; ClusterID = 7; CurrentTime = time() ]]
>>> schedd.edit('ClusterID =? 7', "foo", "'bar'")
>>> schedd.query('true', ['ClusterID', 'foo'])
[[ MyType = "Job"; foo = "bar"; TargetType = "Machine"; ServerTime = 1367284782; ClusterID = 7; CurrentTime = time() ]]
>>> schedd.edit('ClusterID =? 7', "foo", "42")
>>> schedd.query('true', ['ClusterID', 'foo'])
[[ MyType = "Job"; foo = 42; TargetType = "Machine"; ServerTime = 1367284792; ClusterID = 7; CurrentTime = time() ]]
>>> schedd.act(htcondor.JobAction.Hold, ['7.0'])
[ TotalNotFound = 0; TotalPermissionDenied = 0; TotalAlreadyDone = 1; TotalJobAd
ts = 1; TotalSuccess = 0; TotalChangedAds = 0; TotalBadStatus = 0; TotalError = 0
]
>>> 
```
Schedd Advanced - File Transfer
Advanced querying

- There’s a few intricacies in querying:

- `query` versus `xquery` methods: Query returns a list (everything in memory); `xquery` returns an iterator (one ad in memory at a time).
  - **Filter constraint**: which jobs to return.
  - **Projection**: For matching jobs, which attributes to return?
  - **limit**: Maximum number of jobs to return.
  - **opts**: Return the “auto-clusters” - groups of identical jobs - instead of the jobs themselves.
  - **name**: Provide a tag name on the returned iterator.

- **poll method**: Given a set of iterators, return the first one that has data waiting.
Parting Thoughts

• Who is the target audience?

  • Site admins wanting to automate some operational task.

  • Application / middleware developers whose {webapp, middleware, etc} is implemented in python.
Advanced: Proceed With Caution
import time
import htcondor

coll = htcondor.Collector("vocms099.cern.ch")
queries = []
start = time.time()
print "Querying collector for schedds"
coll_query = coll.query(htcondor.AdTypes.Schedd)
end = time.time()
for schedd_ad in coll_query:
    schedd_obj = htcondor.Schedd(schedd_ad)
    if True or not schedd_ad["Name"].startswith('crab'):
        print "Starting query to", schedd_ad["Name"]
        # queries.append(schedd_obj.xquery(opts=htcondor.QueryOpts.AutoCluster))
        queries.append(schedd_obj.xquery())
end2 = time.time()
print "Found %d schedds to query." % len(queries)

job_counts = {}
for query in htcondor.poll(queries):
    schedd_name = query.tag()
    job_counts.setdefault(schedd_name, 0)
    count = len(query.nextAdsNonBlocking())
    job_counts[schedd_name] += count
    print "Got %d results from %s." % (count, schedd_name)

print job_counts
end3 = time.time()

print "Collector query time: %.2f" % (end-start)
print "Schedd query startup time: %.2f" % (end2-end)
print "Schedd query finish time: %.2f" % (end3-end)
print "Total time: %.2f" % (end3-start)
Interacting With User Logs

- The best way to keep track of job is to follow its “user log” file. Used to implement condor_wait.

- Does not require querying the schedd (taking up scarce resources) to get job state.

- Strange file format - don’t try to write your own parser!

- The read_events API provides an iterator which produces ClassAds - one ad per event.

```python
$ python
Python 2.6.6 (r266:84292, Jul 23 2015, 00:03:09)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> fp = open("/home/wmagent/testcondor.11677.89.log")
>>> htcondor.read_events(fp)
<htcondor.EventIterator object at 0x7fa0c35e09b0>
>>> iterator = htcondor.read_events(fp, True)
>>> iterator.next()
[ MyType = "SubmitEvent"; EventTime = "2016-02-29T13:33:56"; Cluster = 11677; Proc = 89; SubmitHost = "; EventTypeNumber = 0; Subproc = 0 ]
>>> iterator.next()
[ MyType = "ExecuteEvent"; EventTime = "2016-02-29T14:01:22"; Cluster = 11677; Proc = 89; EventTypeNumber = 1; Subproc = 0; ExecuteHost = "]
```
import htcondor

coll = htcondor.Collector()
private_ads = coll.query(htcondor.AdTypes.StartdPrivate)
startd_ads = coll.query(htcondor.AdTypes.Startd)

claim_ads = []
for ad in startd_ads:
    if "Name" not in ad: continue
    found_private = False
    for pvt_ad in private_ads:
        if pvt_ad.get('Name') == ad['Name']:
            found_private = True
            ad['ClaimId'] = pvt_ad['Capability']
            claim_ads.append(ad)

with htcondor.Schedd().negotiate("bbockelm@unl.edu") as session:

    found_claim = False
    for resource_request in session:
        for claim_ad in claim_ads:
            if resource_request.symmetricMatch(claim_ad):
                print "Sending claim for", claim_ad['Name']
                session.sendClaim(claim_ads[0])
                found_claim = True
                break
    if found_claim: break

26