Python and HTCondor

Brian Bockelman

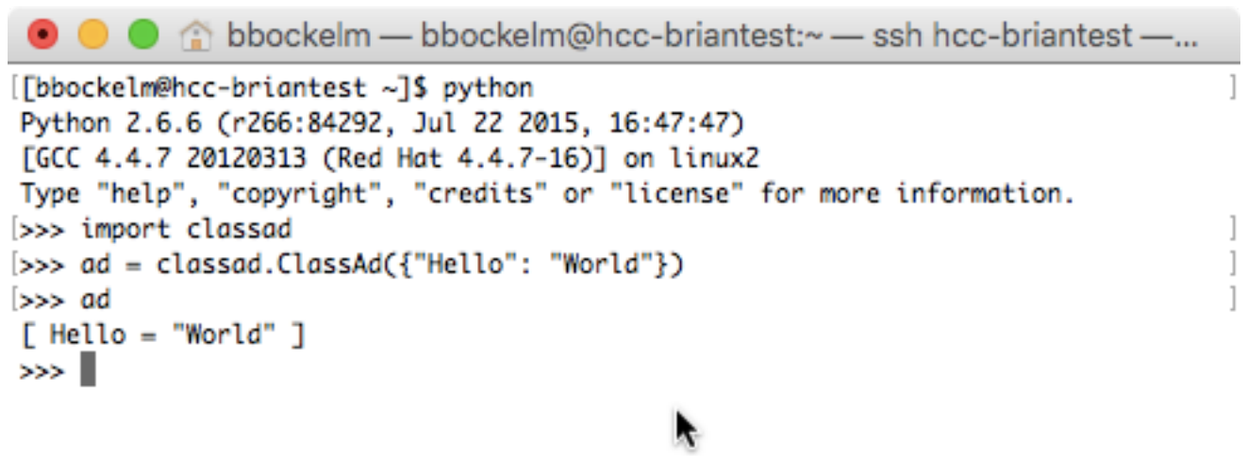
HTCondor / Arc-CE Workshop, Feb 2016

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!

```
>>> 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 =?= %s" % \
                                          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\""; \
  baz = foo && foo + 4 ]
>>> ad.eval('requirements')
False
```

```
bbockelm — bbockelm@hcc-briantest:~ — ssh hcc-briantest —...
Last login: Sat Feb 27 15:04:04 2016 from 12.250.207.42
[bbockelm@hcc-briantest ~]$ 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
>>> ad = classad.ClassAd({"Hello": "World"})
>>> ad
[ Hello = "World" ]
>>> ad["foo"] = "bar"
>>> ad
[ foo = "bar"; Hello = "World" ]
>>> ad["bar"] = [1,2,3]
>>> ad["baz"] = classad.ExprTree("bar")
>>> ad
[ bar = { 1,2,3 }; foo = "bar"; Hello = "World"; baz = bar ]
>>> ad["baz"]
bar
>>> ad.eval("baz")
[1L, 2L, 3L]
>>>
```

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!

```
>>> import classad
>>> ad = classad.ClassAd("""[Owner="bbockelm"; x509UserProxyFirstFQAN="/cms/Role=prod";]""")
>>> ad
[ x509UserProxyFirstFQAN = "/cms/Role=prod"; Owner = "bbockelm" ]
>>> classad.ExprTree(‘regexps(“/(\\\\w+)/", "/cms/Role=prod", "\\\\1")')
regexps(“/(\\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.

```
$ 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 mor
>>> 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

bbockelm — Test Terminal — Python — 80×24

```
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
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> ad = coll.locate(htcondor.DaemonTypes.Schedd, "red.unl.edu")
>>> ad["MyAddress"]
'<129.93.239.129:41562>'
>>> ads = coll.locateAll(htcondor.DaemonTypes.Schedd)
>>> 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
>>>
```

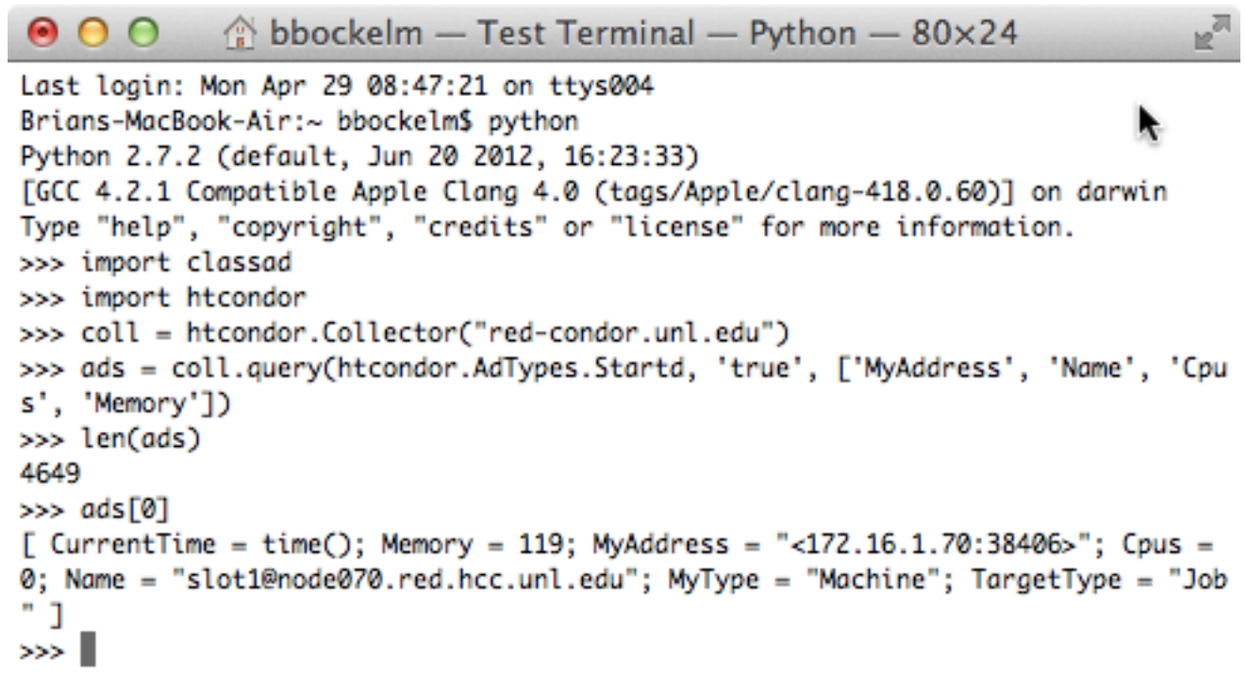
bbockelm — bbockelm@brian-test:~ — ssh — 80×24

```
Connection to red-man.unl.edu closed.
[bbockelm@brian-test ~]$ python
Python 2.4.3 (#1, Jan  8 2013, 21:12:22)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-52)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> import htcondor
>>> ad = classad.ClassAd({"MyType": "Demo", "foo": 1})
>>> coll = htcondor.Collector()
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[]
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[]
>>> ad = classad.ClassAd({"MyType": "Demo", "foo": 1, "Name": "DemoAd"})
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[[ LastHeardFrom = 1367202284; Name = "DemoAd"; MyType = "Demo"; foo = 1; Authen
ticatedIdentity = "unauthenticated@unmapped"; CurrentTime = time() ]]
>>>
```

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



Schedd Basics

bbockelm — Test Terminal — Python — 80×24

```
Last login: Mon Apr 29 14:25:48 on ttys004
Brians-MacBook-Air:~ bbockelm$ grid-proxy-
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
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> schedd_ad = coll.locate(htcondor.DaemonTypes.Schedd, "red.unl.edu")
>>> schedd = htcondor.Schedd(schedd_ad)
>>> jobs = schedd.query()
>>> print jobs[0],
[
    CurrentTime = time();
    BufferSize = 524288;
    JobNotification = 0;
    BufferBlockSize = 32768;
    Err = "/var/lib/globus/job_home/uscmsPool2295/.globus/job/red/1629003077
2317236126.1905433861141216178/stderr";
    CumulativeSlotTime = 0;
    CoreSize = -1;
    NiceUser = false;
    x509UserProxyExpiration = 136742418
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
>>> cluster = schedd.submit(classad.ClassAd({"Cmd": "/bin/sh", "Arguments": "-c
'echo Hello world && sleep 1m'", "Out": "test.out", "Err": "test.err"}))
>>> cluster
4
>>> schedd.query('ClusterID =?= 4')
[[ NumCkpts_RAW = 0; BufferSize = 524288; NumJobMatches = 1; LastMatchTime = 136
7273629; LastJobLeaseRenewal = 1367273629; NiceUser = false; CoreSize = -1; Cumu
lativeSlotTime = 0; OnExitHold = false; GlobalJobId = "ip-10-62-61-234.ec2.inter
nal#4.0#1367273629"; RequestCpus = 1; Err = "test.err"; BufferBlockSize = 32768;
 TransferringInput = false; ImageSize = 100; CurrentTime = time(); WantCheckpoin
t = false; CommittedTime = 0; TargetType = "Machine"; WhenToTransferOutput = "ON
_EXIT"; ServerTime = 1367273642; Cmd = "/bin/sh"; JobUniverse = 5; BytesRecvd =
9.386720000000000E+05; ExitBySignal = false; StartdIpAddr = "<10.62.61.234:50475
>"; PublicClaimId = "<10.62.61.234:50475>#1367273386#7#..."; Iwd = "/home/demo";
 NumRestarts = 0; RemoteHost = "ip-10-62-61-234.ec2.internal"; CommittedSuspensi
onTime = 0; OrigMaxHosts = 1; Owner = "demo"; NumSystemHolds = 0; CumulativeSusp
ensionTime = 0; ShadowBday = 1367273629; RequestDisk = DiskUsage; Requirements =
 true && TARGET.OPSYS == "LINUX" && TARGET.ARCH == "X86_64" && TARGET.HasFileTra
nsfer && TARGET.Disk >= RequestDisk && TARGET.Memory >= RequestMemory; MinHosts
= 1; JobNotification = 0; NumCkpts = 0; LastSuspensionTime = 0; NumJobStarts = 0
= 0.0; JobStartDate = 1367273629; RootDi
StreamOut = false; WantRemoteIO = true;
iskUsage = 1; In = "/dev/null"; Periodic
teUserCpu = 0.0; LocalSysCpu = 0.0; Remo
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
 "Job"; CondorVersion = "$CondorVersion: 7.9.5 Apr 04 2013 BuildID: 114739 $"; J
obRunCount = 1; StreamErr = false; DiskUsage_RAW = 1; PeriodicHold = false; Proc
Id = 0; User = "demo@ip-10-62-61-234.ec2.internal"; TransferQueued = false; Last
JobStatus = 1; Arguments = "-c 'echo Hello world && sleep 1m'"; Out = "test.out"
; JobCurrentStartDate = 1367273629; JobStatus = 2; PeriodicRelease = false; Auto
ClusterAttrs = "JobUniverse,LastCheckpointPlatform,NumCkpts,RemoteGroup,Submitte
rGroup,SubmitterUserPrio,DiskUsage,ImageSize,RequestDisk,RequestMemory,Requireme
nts,NiceUser,ConcurrencyLimits"; RequestMemory = ifthenelse(MemoryUsage isnt und
efined,MemoryUsage,( ImageSize + 1023 ) / 1024); Args = ""; MaxHosts = 1; TotalS
uspensions = 0; CommittedSlotTime = 0; StartdPrincipal = "unauthenticated@unmapp
ed/10.62.61.234"; CondorPlatform = "$CondorPlatform: x86_64_RedHat6 $"; AutoClus
terId = 7; ShouldTransferFiles = "YES"; ExitStatus = 0; NumShadowStarts = 1; Mac
hineAttrCpus0 = 1; QDate = 1367273629; EnteredCurrentStatus = 1367273629 ]]
>>>
>>>
>>> open("test.out", "r").read()
'Hello world\n'
>>>
```

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
- 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)

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

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
[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; Cl
usterID = 7; CurrentTime = time() ]]
>>> schedd.act(htcondor.JobAction.Hold, ['7.0'])
[ TotalNotFound = 0; TotalPermissionDenied = 0; TotalAlreadyDone = 1; TotalJobAd
s = 1; TotalSuccess = 0; TotalChangedAds = 0; TotalBadStatus = 0; TotalError = 0
 ]
>>>
```

Schedd Advanced - File Transfer

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
[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
>>> import classad
>>> schedd = htcondor.Schedd()
>>> ad_results = []
>>> cluster = schedd.submit(classad.ClassAd({"Cmd": "/bin/sh", "Arguments": "-c
'echo Hello world && sleep 1m'"}), 1, True, ad_results)
>>> cluster
5
>>> ad_results[0]
[ BufferSize = 524288; NiceUser = false; CoreSize = -1; CumulativeSlotTime = 0;
OnExitHold = false; RequestCpus = 1; Err = "/dev/null"; BufferBlockSize = 32768;
 ImageSize = 100; CurrentTime = time(); WantCheckpoint = false; CommittedTime =
0; TargetType = "Machine"; WhenToTransferOutput = "ON_EXIT"; Cmd = "/bin/sh"; Jo
bUniverse = 5; ExitBySignal = false; HoldReasonCode = 16; Iwd = "/home/demo"; Nu
mRestarts = 0; CommittedSuspensionTime = 0; Owner = undefined; NumSystemHolds =
0; CumulativeSuspensionTime = 0; RequestDisk = DiskUsage; Requirements = true &&
 TARGET.OPSYS == "LINUX" && TARGET.ARCH == "X86_64" && TARGET.HasFileTransfer &&
 TARGET.Disk >= RequestDisk && TARGET.Memory >= RequestMemory; MinHosts = 1; Job
Notification = 0; NumCkpts = 0; LastSuspensionTime = 0; NumJobStarts = 0; WantRe
moteSyscalls = false; JobPrio = 0; RootDir = "/"; CurrentHosts = 0; StreamOut =
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID"])
[[ MyType = "Job"; TargetType = "Machine"; ServerTime = 1367275365; ClusterID =
5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 5; TargetType = "Machine"; ServerTime = 136727539
3; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.spool(ad_results)
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 2; TargetType = "Machine"; ServerTime = 136727541
1; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 4; TargetType = "Machine"; ServerTime = 136727547
1; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.retrieve('Cluster =?= 5')
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 4; TargetType = "Machine"; ServerTime = 136727550
4; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.act(htcondor.JobAction.Remove, ['5.0'])
[ TotalNotFound = 0; TotalPermissionDenied = 0; TotalAlreadyDone = 0; TotalJobAd
s = 1; TotalSuccess = 1; TotalChangedAds = 1; TotalBadStatus = 0; TotalError = 0
 ]
>>>
>>>
>>>
```

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.

Putting it all together - Non-blocking, concurrent queries

```
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)
```

Don’t memorize!

Instead, make sure you get the concept

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 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 = "<128.142.209.43
addrs=128.142.209.43-4080&noUDP&sock=9265_e974_13>"; EventTypeNumber = 0; Subproc = 0 ]
>>> iterator.next()
[ MyType = "ExecuteEvent"; EventTime = "2016-02-29T14:01:22"; Cluster = 11677; Proc = 89; EventTypeNumber = 1; Subproc
ExecuteHost = "<10.29.44.3:11644?CCBID=128.142.141.24:9730%3faddrs%3d128.142.141.24-9730#208326&addrs=10.29.44.3-11644
```

Bonus: Building a Negotiator

```
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
```