What's new in HTCondor? What's coming?

HTCondor European Workshop 2020

Todd Tannenbaum
Center for High Throughput Computing
Department of Computer Sciences
University of Wisconsin-Madison


Why aren't there handles on doors aymore?
Wesley, it's time I tell you about the year 2020...
STARECAT.COM


'Can you
things fo
ew
:?'
I HELP YOU DO STUFF
AND I HELP UNDO ALL THE STUFF YOU DID

Release Series

- Stable Series (*bug fixes only*)
  - HTCondor v8.8.x – first introduced Jan 2019

    (Currently at v8.8.10)
- Development Series (*should be 'new features' series)*
  - HTCondor v8.9.x (Currently at v8.9.8)
- Since July 2019…
  - Public Releases: 10
  - Documented enhancements: ~98
  - Documented bug fixes: ~165
- Detailed Version History in the Manual
  - https://htcondor.readthedocs.io/en/latest/version-history/

What's new in v8.8 and/or cooking for v8.9 and beyond?

HTCondor v8.9.x Removes Support for:

- Goodbye RHEL/Centos 6 Support
- Goodbye Quill
- Goodbye "Standard" Universe
  - Instead self-checkpoint vanilla job support [1]
- Goodbye SOAP API
  - So what API beyond the command-line?

[1] https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=HowToRunSelfCheckpointingJobs

API Enhancements: Python, REST

Python

- Bring HTC into Python environments incl Jupyter
- HTCondor Bindings (`import htcondor`) are steeped in the HTCondor ecosystem
  - Exposed to concepts like Schedds, Collectors, ClassAds, jobs, transactions to the Schedd, etc
- **Added new Python APIs**: DAGMan submission, DAG creation (htcondor.dags), credential management (i.e. Kerberos/Tokens)
- Initial integration with **Dask**
- Released our **HTMap package**
  - No HTCondor concepts to learn, just extensions of familiar Python functionality. Inspired by BNL!

htcondor package

```
import htcondor

# Describe jobs
sub = htcondor.Submit('''
    executable = my_program.exe
    output = 'run$(ProcId).out'
    ''')

# Submit jobs
schedd = htcondor.Schedd()
with schedd.transaction() as txn:
  clusterid = sub.queue(txn,count = 10)

# Wait for jobs
import time
while len(schedd.query(
   constraint='ClusterId=='+str(clusterid),
   attr_list=['ProcId'])):
      time.sleep(1)
```

htmap package

```
import htmap

# Describe work
def double(x):
    return 2 * x

# Do work
doubled = htmap.map(double,range(10))

# Use results!
print(list(doubled))
# [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

See https://github.com/htcondor/htmap

REST API

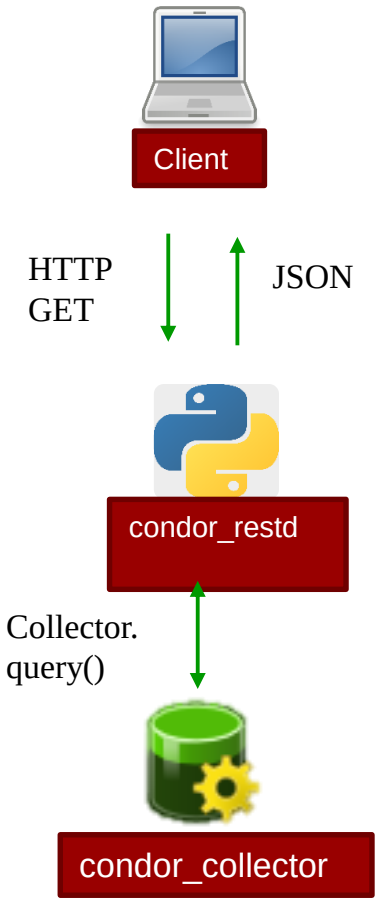
- Python (Flask) webapp for querying HTCondor jobs, machines, and config
- Runs alongside an HTCondor pool
- Listens to HTTP queries, responds with JSON
  - Built ontop of Python API
  - other cool tools coming

    courtesy Python API…

    ….like condor_watch_q !

https://htcondor.readthedocs.io/en/latest/man-pages/condor_watch_q.html

REST API, cont




```
$ curl "http://localhost:9680/v1/status\
?query=startd\
&projection=cpus,memory\
&constraint=memory>1024"
```

```
[
    {
        "name": "slot4@siren.cs.wisc.edu",
        "type": "Machine",
        "classad": {
            "cpus": 1,
            "memory": 1813
        }
    },
…
]
```

REST API, cont

- Swagger/OpenAPI spec to generate bindings for Java, Go, etc.
- Evolving, but see what we've got so far at
  - https://github.com/htcondor/htcondor-restd

- Potential Future improvements
  - Allow changes (job submission/removal, config editing)
  - Add auth
  - Improve scalability
  - Run under shared port

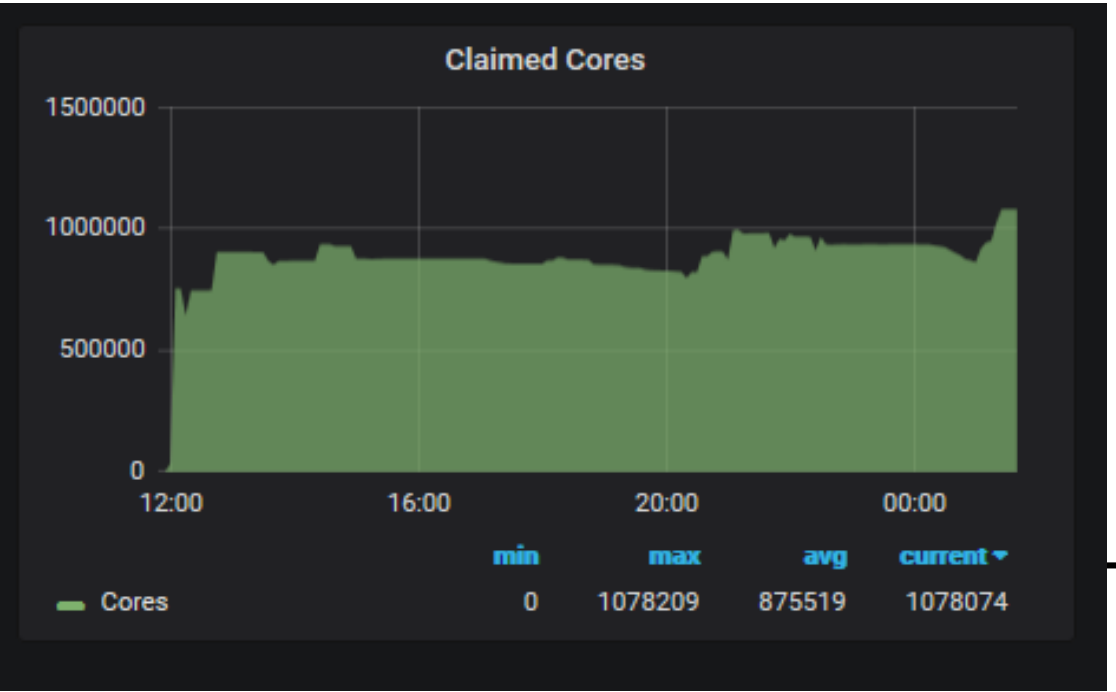
Federation of Compute resources: HTCondor Annexes

HTCondor "Annex"

- Instantiate an HTCondor Annex to dynamically add additional execute slots into your HTCondor environment
- Want to enable end-users to provision an Annex on
  - Clouds
  - HPC Centers / Supercomputers
    - Via edge services (i.e. HTCondor-CE)
  - Kubernetes clusters

http://news.fnal.gov/2018/07/fermilab-computing-experts-bolster-nova-evidence-1-million-cores-consumed/

CPU cores!



FNAL HEPCloud
NOvA Run
(via Annex at NERSC)

HTCondor

A cost effective ExaFLOP hour in the Clouds for IceCube

Published on February 5, 2020

Igor Sfiligoi
Lead Scientific Software Developer and Researcher at San Diego Supercomputer Center

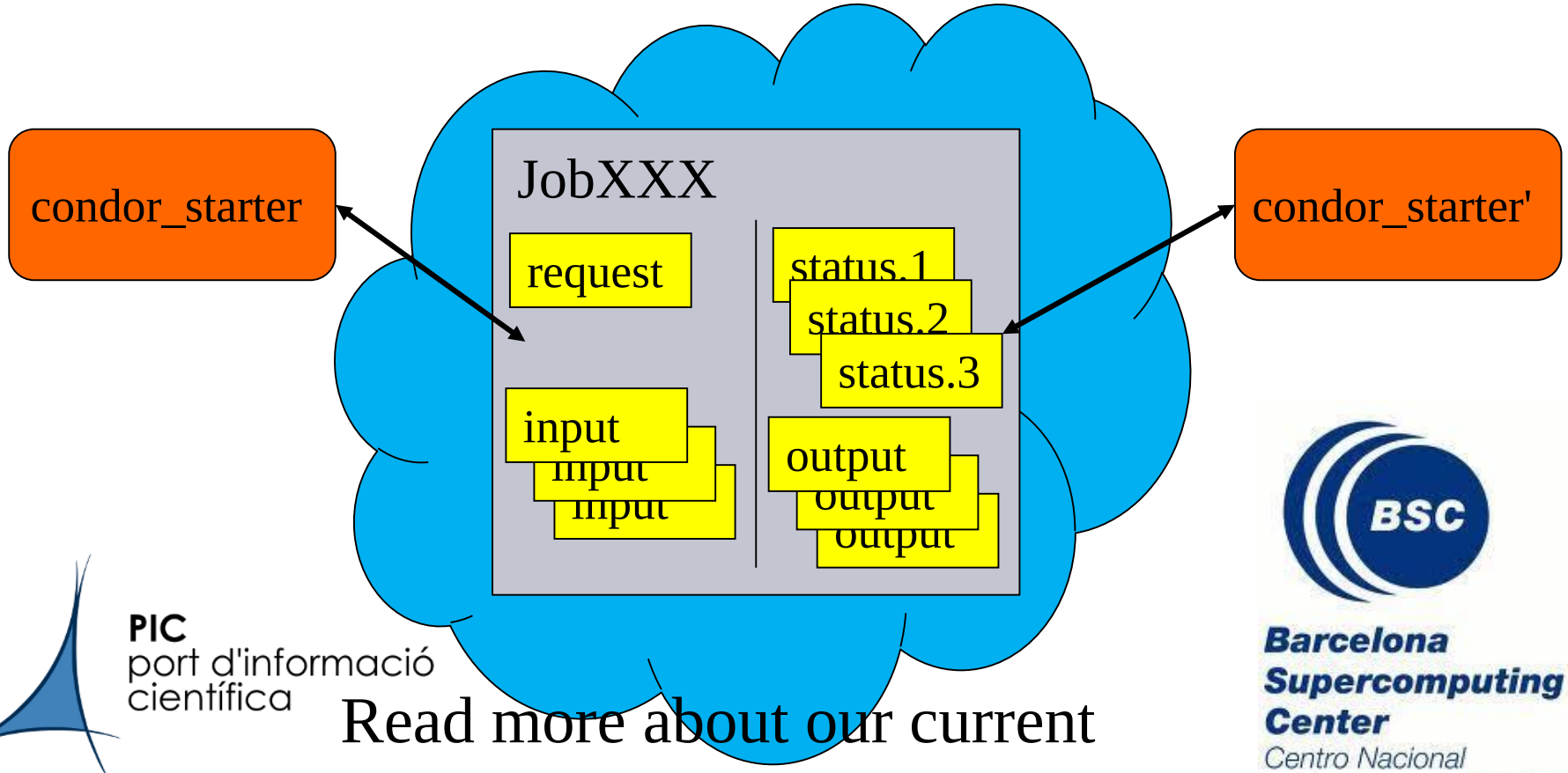
9 articles
✓ Following

We did it again! Another large scale Cloud GPU burst to run IceCube simulations, a dHTC application. If you missed our first run, see my other article.

The objective this time, on top of (obviously) the science output, was to demonstrate how much compute can someone integrate during a regular working day, using only the two most cost effective SKUs for each Cloud provider. As before, we used Cloud resources from Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP), but we also mixed in the on-prem resources available though the Open Science Grid (OSG), XSEDE, the Pacific Research Platform (PRP) and others. As before, HTCondor was used as the workload management system.

https://www.linkedin.com/pulse/cost-effective-exaflop-hour-clouds-icecube-igor-sfiligoi/

No internet access to/from HPC nodes? File-based communication between execute nodes



PIC
port d'informació
científica

Read more about our current approach at http://tiny.cc/f158cz

BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

GPUs

- HTCondor has long been able to detect GPU devices and schedule GPU jobs (CUDA/OpenCL)
- *New in v8.8:*
  - Monitor/report job GPU processor utilization
  - Monitor/report job GPU memory utilization
- *In the works*: simultaneously run multiple jobs on one GPU device
  - Specify GPU memory for scheduling ?
  - NVIDIA Multi-Instance GPU (MIG) for partitioning ?
  - Working with LIGO on requirements

Containers and Kubernetes

HTCondor Singularity Integration

- What is Singularity?

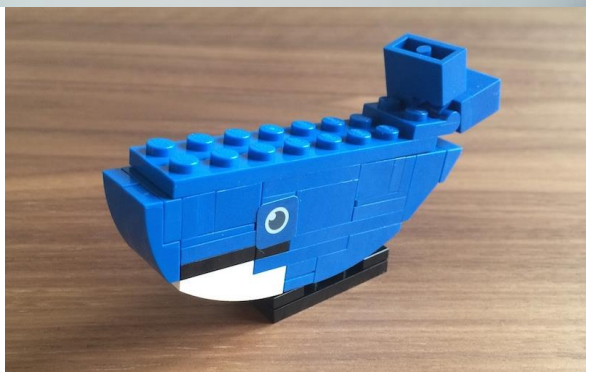
  Like Docker but…

  - No root owned daemon process, just a setuid
  - No setuid required (as of very latest RHEL7)
  - Easy access to host resources incl GPU, network, file systems

- HTCondor allows admin to define a policy (with access to job and machine attributes) to control
  - Singularity image to use
  - Volume (bind) mounts
  - Location where HTCondor transfers files

Docker Job Enhancements

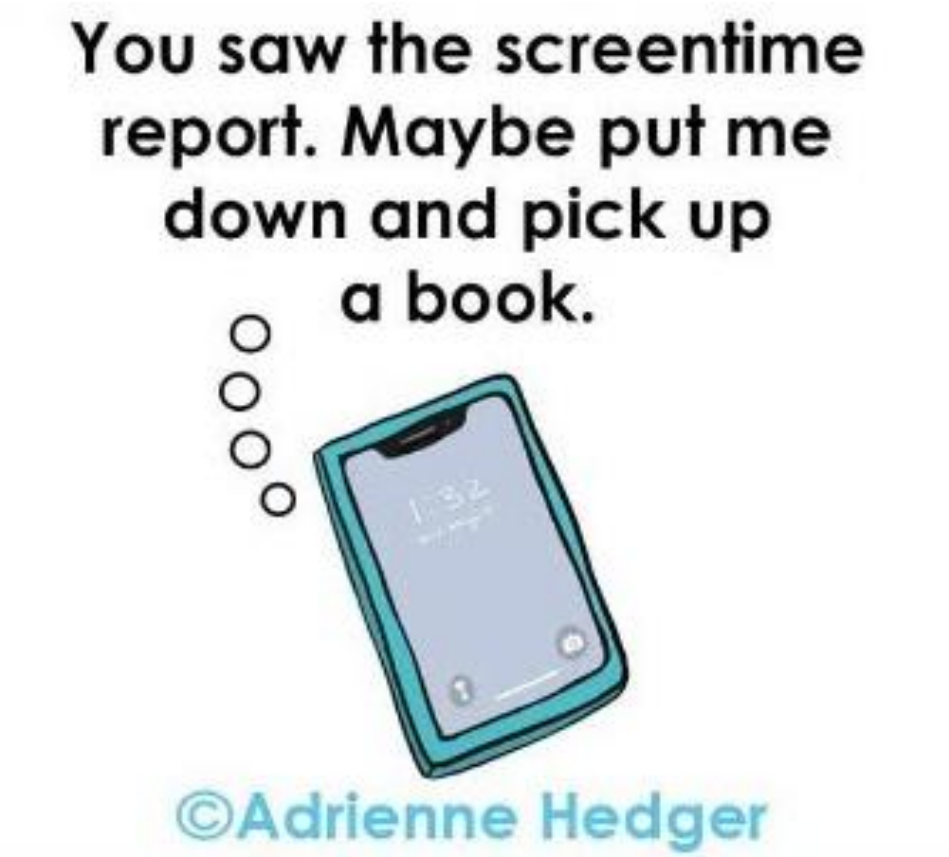
- Docker jobs get usage updates (i.e. network usage) reported in job classad
- Admin can add additional volumes
- Conditionally drop capabilities
- Condor Chirp support
- Support for condor_ssh_to_job
  - For both Docker and Singularity
- Soft-kill (SIGTERM) of Docker jobs upon removal, preemption



More work coming

- From "Docker Universe" to just jobs with a container image specified
- Kubernetes
  - Package HTCondor as a set of container images
    - Check it out https://github.com/htcondor/htcondor/blob/master/build/docker/services/README.md
    - Next talk!
  - Launch a pool in a Kubernetes cluster
    - Thursday's Talk!

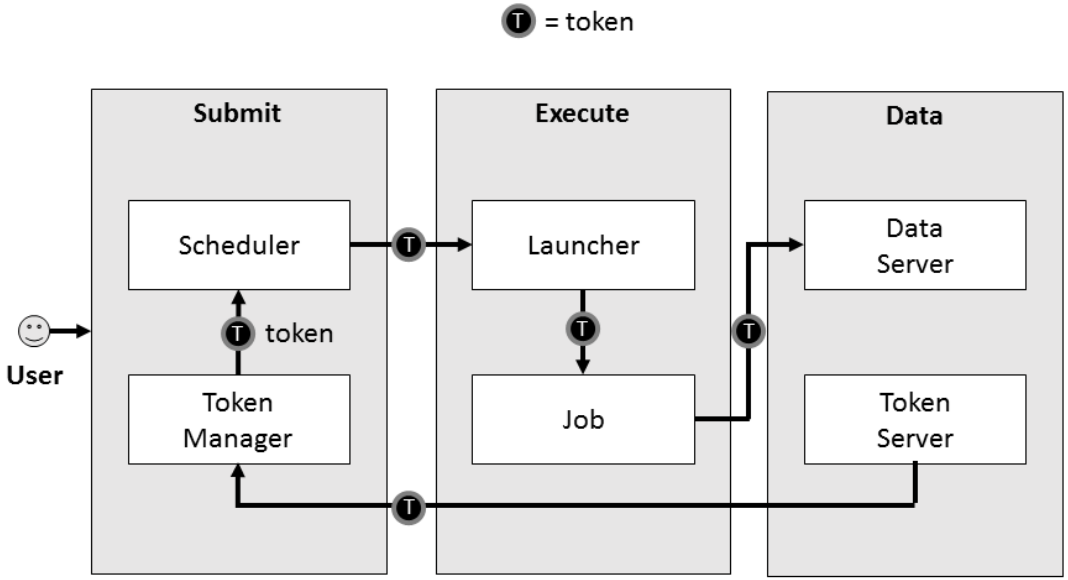
Security Changes and Enhancements



IDTOKENS Authentication Method

- Several Authentication Methods
  - File system (FS), SSL, pool password….
- Adding a new "IDTOKENS" method
  - Administrator can run a command-line tool to create a token to authenticate a new submit node or execute node
  - Users can run a command-line tool to create a token to authenticate as themselves
- "Promiscuous mode" support

SciTokens: From identity certs to authorization tokens



- HTCondor has long supported GSI certs
- Then added Kerberos/AFS tokens w/ CERN, DESY
- Now adding standardized token support
  - SciTokens (http://scitokens.org) for HTCondor-CE, data
  - OAuth 2.0 Workflow → Box, Google Drive, AWS S3, …

Data Management

Data Reuse Mechanism

- Lots of data is shared across jobs
- Data Reuse mechanism work in v8.9 can cache job input files on the execute machine
  - On job startup, submit machine asks execute machine if it already has a local copy of required files
  - Cache is limited in size by administrator, LRU replacement
- Todo list includes using XFS Reflinks
  https://blogs.oracle.com/linux/xfs-data-block-sharing-reflink

File Transfer Improvements

- If you use HTCondor to manage credentials, **we include file transfer plugins for Box.com, Google Drive, AWS S3, and MS One Drive cloud storage** for both input files and output files, and credentials can also be used with HTTP URL-based transfers. Available in 8.9.4.
- **Error messages *greatly* improved**: URL-based transfers can now provide sane, human-readable error messages when they fail (instead of just an exit code). Available in 8.8 series.
- URLs for output: Individual **output files can be URLs**, allowing stdout to be sent to the submit host and large output data sent elsewhere. Available in 8.9.1.
- **Smarter retries**. Including retries triggered by low throughput. Available in 8.9.2.
- Via both job attributes and entries in the job's event log, **HTCondor tells you the time when file transfers are queued, when transfers started, and when transfers completed**.
- **Performance improvements.** No network turn-around between files, And all transfers to/from the same endpoint happen over the same TCP connection. Available v8.9.2
- Have an interesting use case? **Jobs can now supply their own file transfer plugins** — great for development! Available in 8.9.2.

```
executable = myprogram.exe

transfer_input_files = box://htcondor/myinput.dat

use_oauth_services = box

queue
```

Scalability Enhancements

- Central manager now manages queries
  - Queries (ie condor_status calls) are queued; priority is given to operational queries
- More performance metrics (e.g. in collector, DAGMan)
- *In v8.8 late materialization* of jobs in the schedd to enable submission of very large sets of jobs
  - Submit / remove millions of jobs in < 1 sec
  - More jobs materialized once number of idle jobs drops below a threshold (like DAGMan throttling)

Late materialization

This submit file will stop adding jobs into the queue once 50 jobs are idle:

```
executable = foo.exe
arguments = -run $(ProcessId)
materialize_max_idle = 50
queue 1000000
```

From Job *Clusters* to Job *Sets*

- Job "clusters" (even with late materialization) mostly behave as expected
  - Can remove all jobs in a cluster
  - Can edit all jobs in a cluster
- But some operations are missing
  - Append jobs to a set (in a subsequent submission)
  - Move an entire set of jobs from one schedd to another
  - Job set **aggregates** (for use in polices?)

From Job *Clusters* to Job *Sets*, cont

- Users want to think about a set of jobs as it relates to their mental model, e.g.
  - Set of jobs analyzing genome 52
  - Set of jobs doing analysis on image captures from date xxx
- Initial support for sets in v8.9:
  - User supplies a set name upon submission
  - All jobs with the same name are in the same set
  - Aggregate statistics on set written to History file when last job in a set completes
  - More set operations to come…

DAGMan: Scalability Optimizations

- **DAGMan memory diet**! Dedup node string data, edges are vectors instead of lists, etc:
  - In one DAG, these reduced the memory footprint from 50 GB to 4 GB
- Can **now submit jobs directly** (and faster!) without forking condor_submit
- Introduced **join nodes**, which dramatically reduce the number of edges in dense DAGs. In one particularly dense DAG with ~300,000 edges, join nodes resulted in the following improvements:
  - ~660M edges reduced to ~1.5M edges.
  - condor_dagman memory footprint dropped from 90 GB to 1 GB
  - Parsing time reduced from 1 hour to 20 seconds

Workflows: Provisioner Nodes

- Working to implement **provisioner nodes**
  - Special node that runs for the duration of a workflow
- Responsible for provisioning compute resources on remote clusters (Amazon EC2, Microsoft Azure, etc.)
- Important: also responsible for **deprovisioning** resources after they are no longer needed.
  - These resources cost money.
  - If we fail to deprovision them, this can incur large costs.
  - Recovery from failures is a first class citizen.

Workflows: What's Coming Next

- Sets, Sets, Sets!
  - New syntax in the DAGMan language to describe sets of jobs
- Defining ranges in DAG declarations
  - New syntax to declare ranges of objects
  - No more 10,000 JOB statements to declare 10,000 jobs whose only difference is a numeric suffix.
- condor_dagedit
  - Ability to edit certain properties of in-progress DAGs (MaxJobs, MaxIdle, etc.)
- Dataflow mode
  - Ability to skip jobs that have already been run (like /usr/bin/make!)

Thank You!