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

HTCondor Week 2015

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

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

Release Timeline

- Development Series
  - HTCondor v8.3.6 frozen, in beta test, release to web 6/9/15.
  - HTCondor v8.3.7 (final features for v8.3 series, default settings improvements), release 6/30/15.
  - HTCondor v8.3.8 (valgrind + Coverity + bug fixes), v8.4.0 Release Candidate, release 7/21/15.
- Stable Series
  - HTCondor v8.4.0 – first half of August
  - v8.2.9 will *likely* be the last v8.2.x released
  - Last Year: Condor v8.2.0 (June 24th 2014)
- Since HTCondor Week 2014: 17 releases, 2337 commits by 22 contributors

HTCondor v8.2 Enhancements

- EC2 Grid Job Improvements
- Better support for OpenStack
- Google Compute Engine Jobs
- HTCond… sub… i… BOINC
- Scalabil… …ov…
- GPU Su…
- New Co… …n File Constru… …ncluding includes, conditionals, meta-knobs

- Asynchronous Stage-out of Job Output
- Ga…ia Mo…i…g via co…
- …on…o…
- …y… …il… transfer …ch…duling via disk I/O Load
- Daily pool job run statistics via condor_job_report
- Monitoring via BigPanDAmon

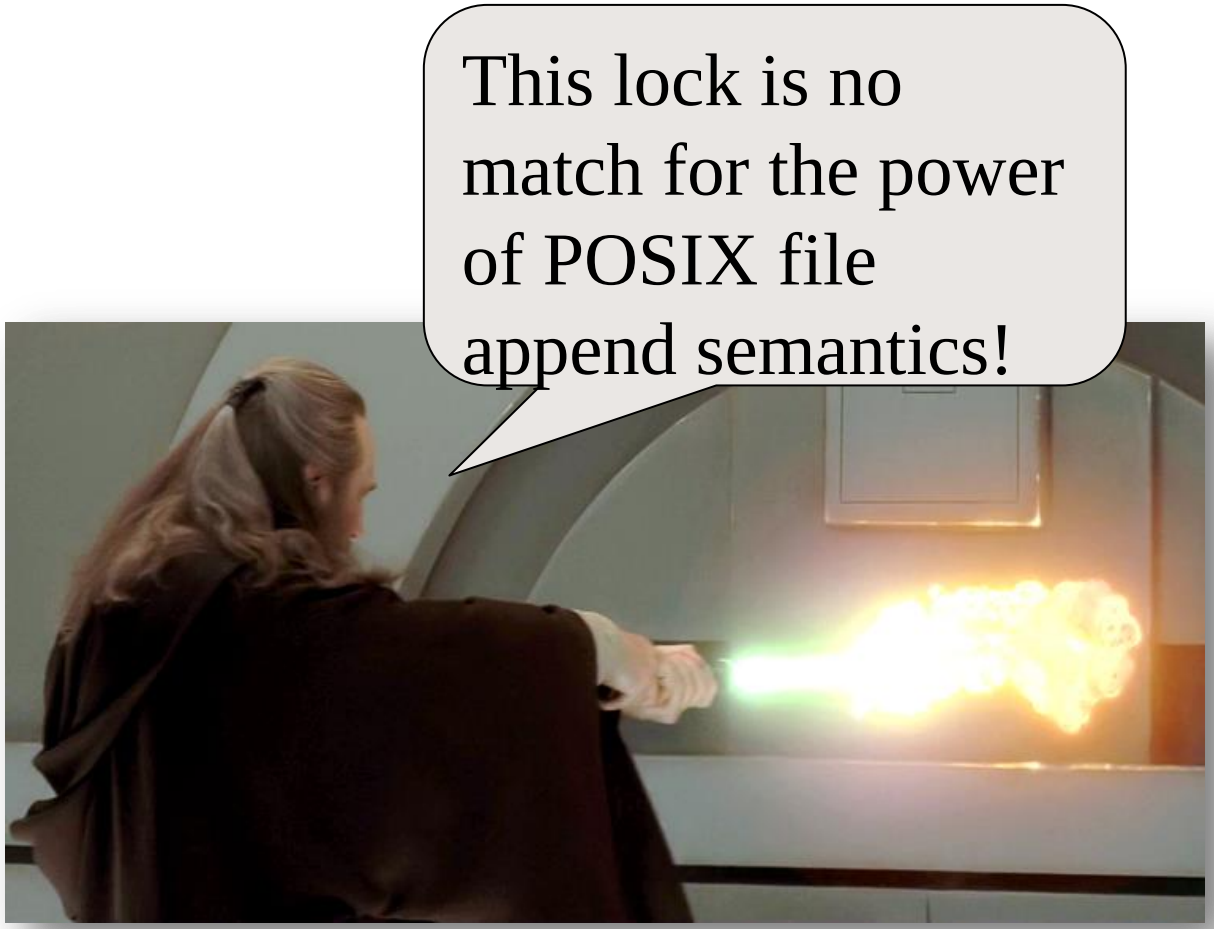
Last Year's News

Some HTCondor v8.3 Enhancements

- Scalability and stability
  - Goal: 200k slots in one pool, 10 schedds managing 400k jobs
  - Resolved developer tickets: 240 bug fix issues (v8.2.x tickets), 234 enhancement issues (v8.3 tickets)
- Docker Job Universe
- Tool improvements, esp condor_submit
- IPv6 mixed mode
- Encrypted Job Execute Directory
- Periodic application-layer checkpoint support in Vanilla Universe
- Submit requirements
- New packaging

Scalability Enhancement Examples

Elimination of File Locking on Job and Schedd Event Logs



- File lock requests on Linux are not scheduled
- Schedd observed blocked for minutes!

Condor_shadow resources

A condor_shadow spawned for each running job

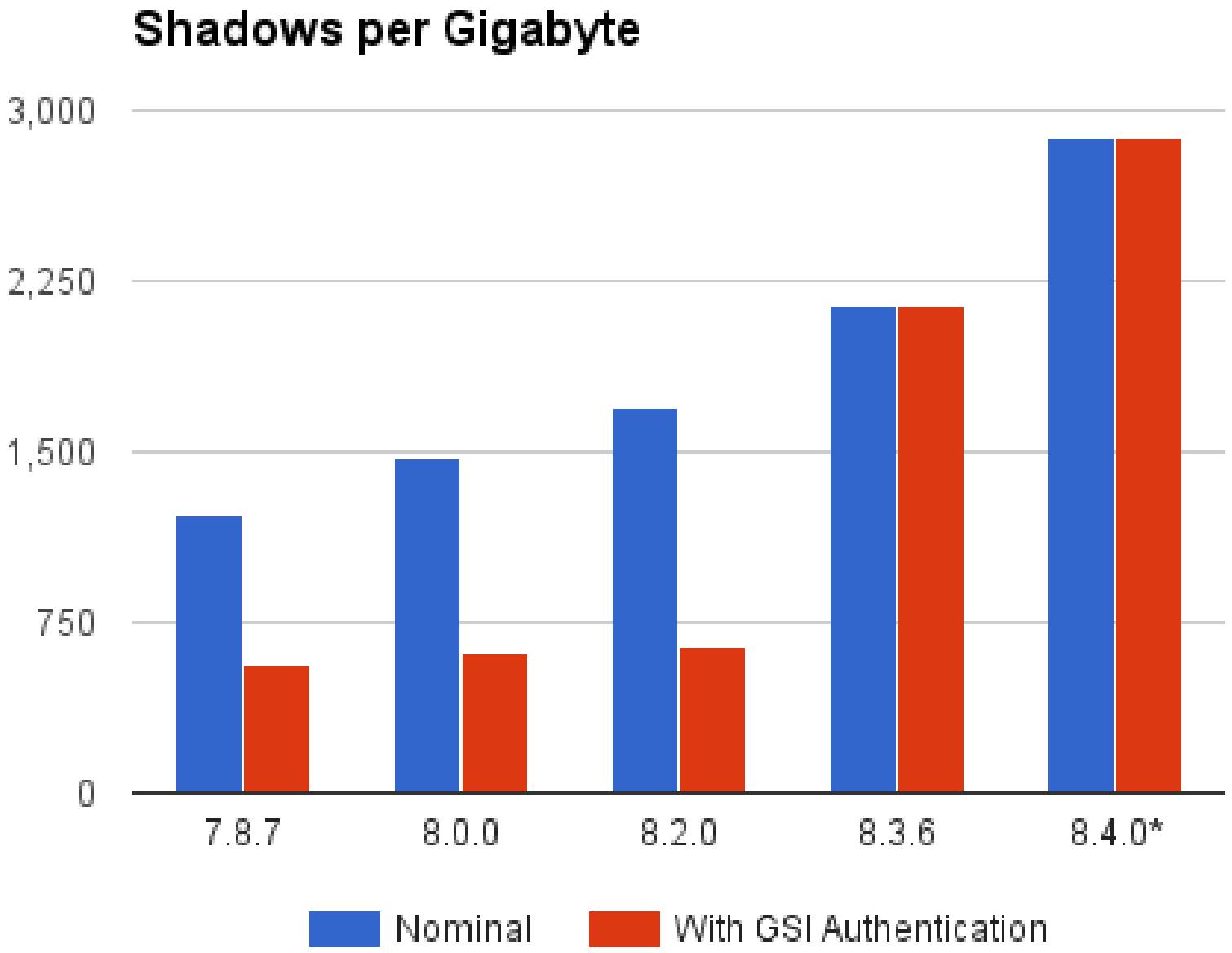
Upon spawn, shadow authenticates to schedd, startd (on execute host)

This authentication uses CPU, Memory

Solution:

Shadow Diet!!

Eliminate Authentication!



v7.8.7:
860KB/
1860KB

v8.4.0
386KB

Authentication Speedups

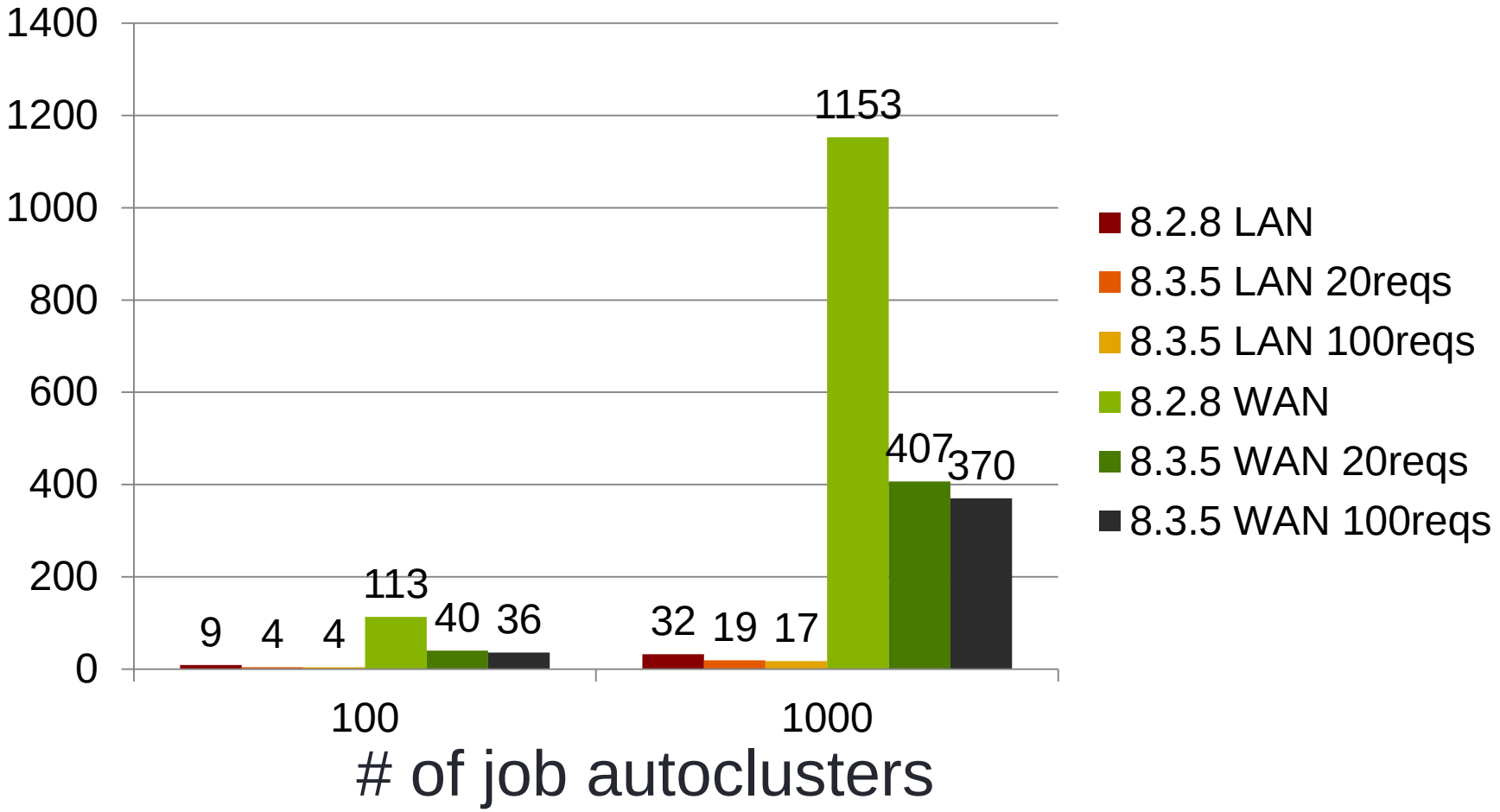
- FS (file system) and GSI authentication are now performed asynchronously
  - So now a Condor daemon can perform many authentications in parallel
  - CMS pool went from 200 execute nodes (glideins) per collector to 2000
- Can cache mapping of GSI certificate name to user name
  - Mapping can be heavyweight, esp if HTCondor has to contact an external service (LCMAPS…)
  - Knob name is `GSS_ASSIST_GRIDMAP_CACHE_EXPIRATION`

Faster assignment of resources from central manager to schedd

- Negotiator can ask the schedd for more than one resource request per network round trip.

```
NEGOTIATOR_RESOURCE_REQUEST_LIST_SIZE = 20
```

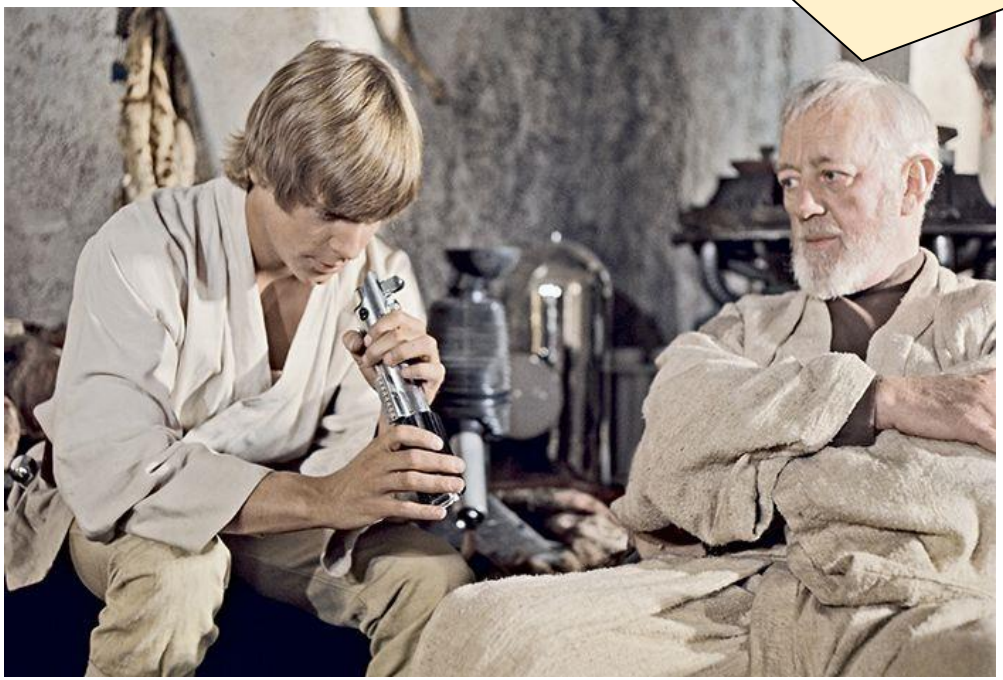
Impact of multiple resource requests Negotiation times for 1000 slot pool

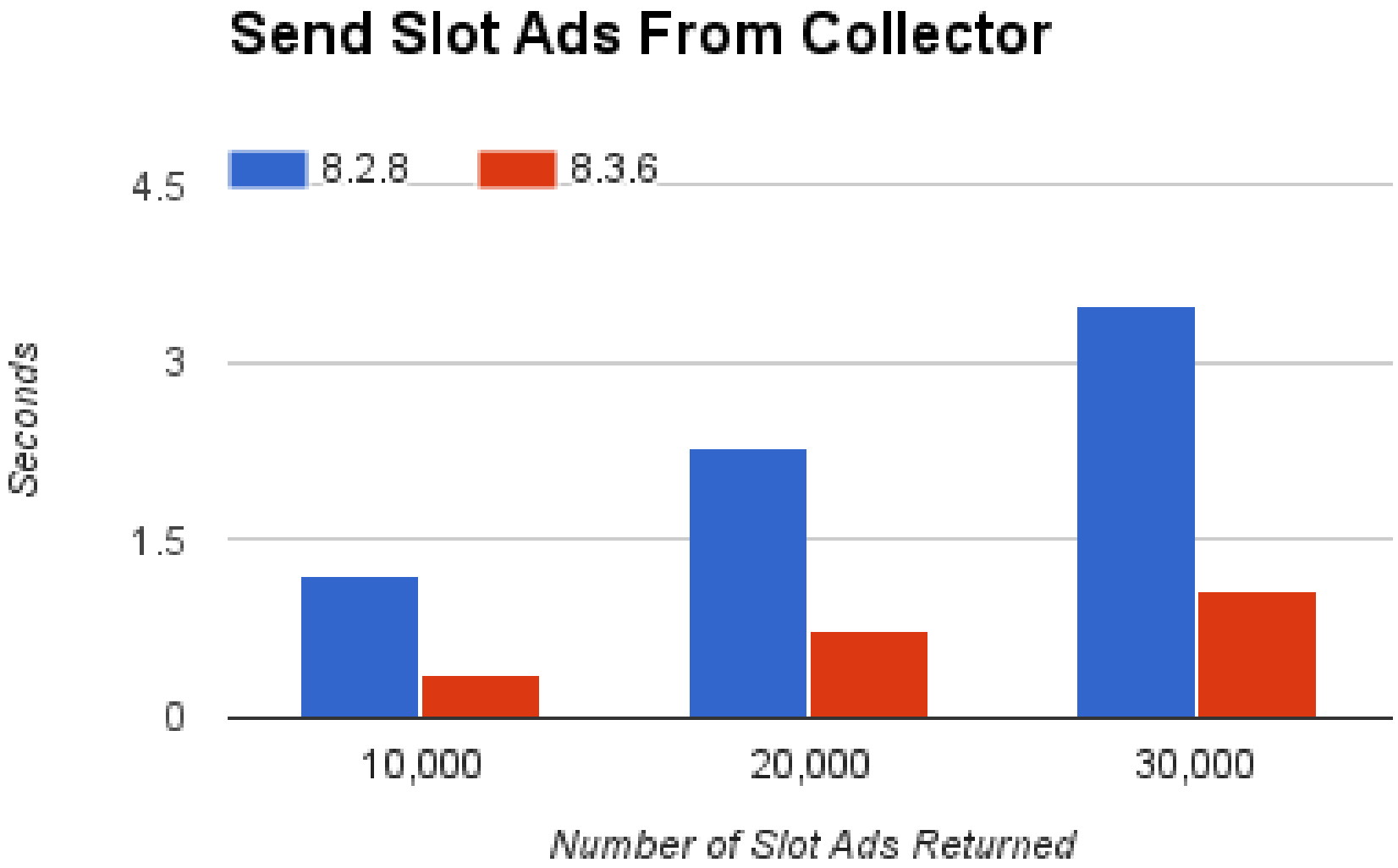


ClassAd Projection Improvements

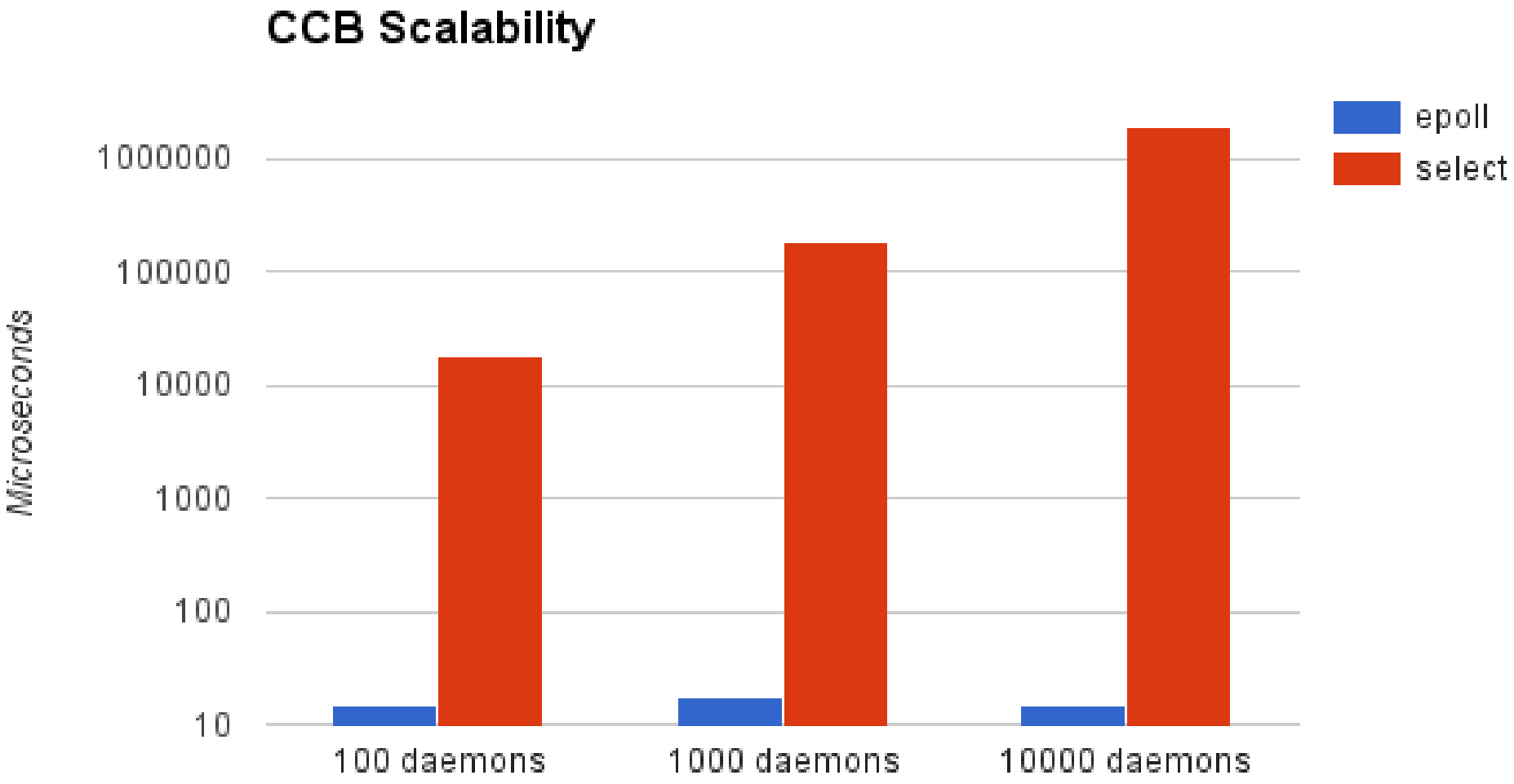
- Less CPU required to send big projections of ClassAds

"*ClassAds*. This is the weapon of sysadmin. Not as clumsy or random as a grep or regex. A more elegant weapon for a more civilized age…"



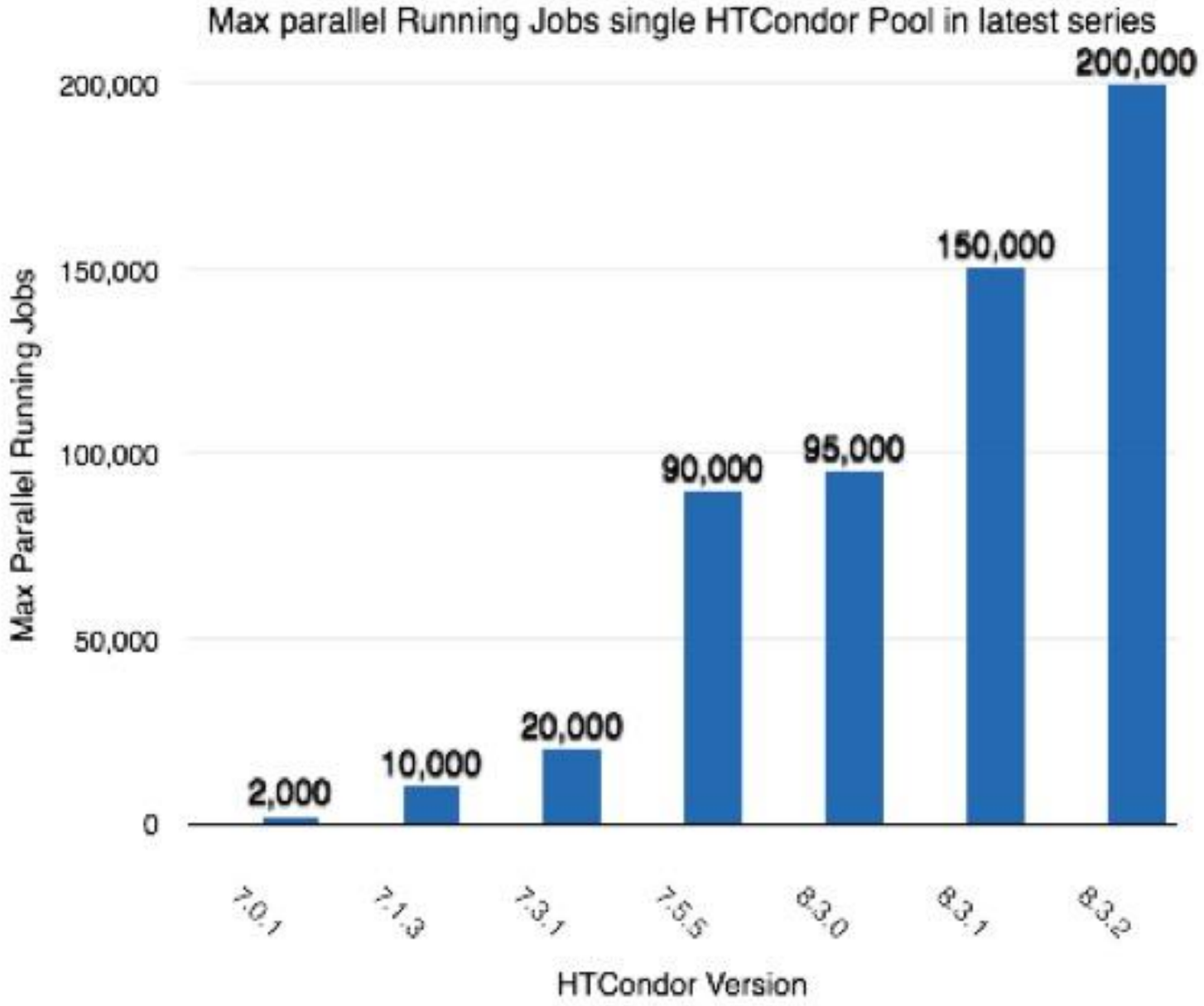

Send Slot Ads From Collector
8.2.8
8.3.6
4.5
3
1.5
0
Seconds
10,000
20,000
30,000
Number of Slot Ads Returned

Eliminate CCB service pauses



Query Responsiveness

- Improvement: Collector will not fork for queries to small tables
  - Load Collector with 100k machine ads
  - Before change: ~4.5 queries/second
  - After change: ~24.4 queries/second
- Improvement: Schedd condor_q quantum adjusted (to 100ms)
  - Load schedd with 100k jobs ads, 40Hz job throughput
  - Before change: ~135 seconds per condor_q
  - After change: ~22 seconds per condor_q


Max parallel Running Jobs single HTCondor Pool in latest series
Max Parallel Running Jobs
200,000
150,000
100,000
50,000
0
2,000
10,000
20,000
90,000
95,000
150,000
200,000
7.0.1
7.1.3
7.3.1
7.5.5
8.3.0
8.3.1
8.3.2
HTCondor Version

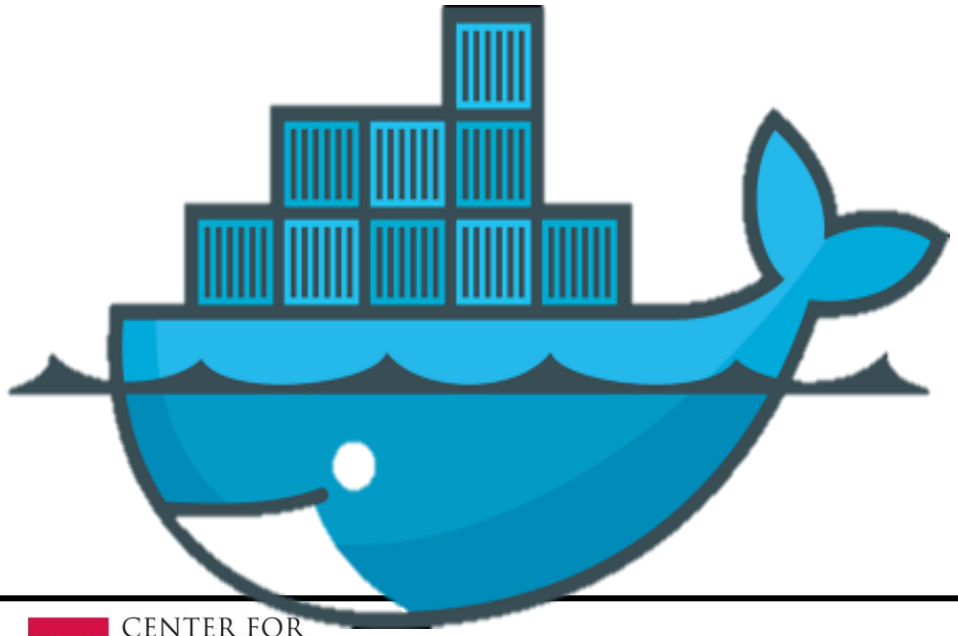
Container Support (*Black Box Applications*)

- HTCondor cgroup support now manages swap space i
  - request_sw
- [[Also a lot of
  manage netw
  - request_ne
- New job univ
  Containers

This is Docker

Docker manages Linux containers.

Provides :



- Process space
- NATed network
- Root file system (image)
- Namespace for images
- UID space

Condor startd detects docker

```
$ condor_status -l | grep -i docker
HasDocker = true
DockerVersion = "Docker version
1.5.0, build a8a31ef/1.5.0“

$ condor_status -const HasDocker
```

Docker Universe

```
universe = docker
docker_image = deb7_and_HEP_stack
transfer_input_files = some_input
executable = /bin/my_executable
arguments = arg1
output = out
error = err
log = log
queue
```

Docker Universe Job Is still a job

- Docker containers have the job-nature
  - condor_submit
  - condor_rm
  - condor_hold
  - Write entries to the job event log(s)
  - condor_dagman works with them
  - Policy expressions work.
  - Matchmaking works
  - User prio / job prio / group quotas all work
  - Stdin, stdout, stderr work
  - Etc. etc. etc.*

Scratch dir == Volume

Means normal file xfer rules apply

- transfer in, transfer out
- subdirectory rule holds
- condor_tail works

Any changes to the container are not xfered

Container is removed when executable exits

Docker Resource limiting

```
RequestCpus = 4
RequestMemory = 1024M
```

RequestCpus translated into cgroup shares

RequestMemory enforced

- If exceeded, job gets OOM killed
- job goes on hold

Surprises with Docker Universe

Condor_ssh_to_job doesn't work (*yet*…)

Condor_chirp doesn't work (*yet*…)

Suspend doesn't work

Can't access NFS/shared filesystems

Networking is only NAT

Many condor_submit improvements

You submit your jobs with *that* script??!? You're braver than I thought!

More ways to Queue 'foreach'

```
Queue <N> <var> in (<item-list>)
Queue <N> <var> matching (<glob-list>)
Queue <N> <vars> from <filename>
Queue <N> <vars> from <script> |
```

- Iterate <items>, creating <N> jobs for each item
- In/from/matching keywords control how we get <items>
- There's more. See the manual for details.

Example: Queue in

```
Args = $(Item)
Queue 2 in ( alpha, beta delta gamma )
```

- Produces 8 jobs (2 for each item)
- It unrolls to this submit file:

```
Item=alpha
Step=0
Queue
Step=1
Queue
Item=beta
Step=0
Queue
  ...
```

Queue matching files

```
Queue 3 Item matching (*.dat, m*)
```

- Produces 3 jobs for each file that matches *.dat *or* m* (or both)
- $(Item) holds each filename in turn

Queue from

`Queue from <filename>`

- Read <filename> and treat lines as items

`Queue from <script> |`

- Execute <script> and treat output lines as items

Condor_q new arguments

- -limit <num>
  - Show at most <num> records
- -totals
  - Show only totals
- -dag <dag-id>
  - Show all jobs in the dag
- -autocluster -long
  - Group and count jobs that have same requirements
  - …perfect for provisioning systems

Tool improvement questions?

IPv6 Support

- New in 8.4 is support for “mixed mode,” using IPv4 and IPv6 simultaneously.
- A mixed-mode pool’s central manager and submit nodes must each be reachable on both IPv4 and IPv6.
- Execute nodes and (other) tool-hosting machines may be IPv4, IPv6, or both.
- ENABLE_IPV4 = TRUE
  ENABLE_IPV6 = TRUE

How Mixed Mode Works

- Each 8.4 daemon includes its IPv4 and its IPv6 address in its advertisement.
- Older clients ignore the new information and just use IPv4. (This was the tricky part.)
- 8.4 clients decide which address to use based on their own configuration.
- We Boldly Claim™ that everything will Just Work™.

IPv6 questions?

Encrypted Execute Directory

- Jobs can request (or admins can require) that their scratch directory be encrypted in realtime
  - /tmp and /var/tmp output also encrypted
  - Put `encrypt_execute_directory=True` in job submit file (or condor_config)
- Only the condor_starter and job processes can see the cleartext
  - Even a root ssh login / cron job will not see the cleartext
  - Batch, interactive, and condor_ssh_to_job works

Authorization Propagation

- When making network connections, the server has to decide if it authorizes the client:
  - ALLOW_READ, ALLOW_WRITE, etc.

```
ALLOW_ADMINISTRATOR = tannenba@cs.wisc.edu
```

Authorization Propagation

- In HTCondor 8.2.X and earlier, if the server did not authorize the client, it simply closed the TCP connection
- This caused a lot of frustration for clients, as commands would fail with cryptic error messages, or sometimes no error at all!

Authorization Propagation

- Send a command:

```
% condor_restart -master
Sent "Restart" command to local master
```

- But did it take effect? MasterLog:

```
05/20/15 06:22:59 PERMISSION DENIED to
unauthenticated@unmapped from host
128.105.121.64 for command 453 (RESTART)
```

Authorization Propagation

- In 8.3.6 and beyond, authorization information is given back to the client during the command protocol
- No extra network round trips needed!

```
% condor_restart -master
SECMAN:2010:Received "DENIED" from server for user
zmiller@cs.wisc.edu using method FS.
Can't send Restart command to local master
```

Periodic Application-Level Checkpointing in the Vanilla Universe

- Experimental feature!
- If requested, HTCondor periodically sends the job its checkpoint signal and waits for the application to exit.
- If it exits with code 0, HTCondor considers the checkpoint successful and does file transfer, and re-executes the application.
- Otherwise, the job is requeued.

Example Submit File

```
universe                = vanilla
executable              = self-checkpointing
transfer_executable     = true
should_transfer_files   = true
when_to_transfer_output = ON_EXIT_OR_EVICT
+WantCheckpointSignal   = true
+CheckpointSig          = "SIGTERM"
stream_output           = true
stream_error            = true
```

Submit Requirements

- allow administrator to decide which jobs enter the queue
- are a named set of ClassAd constraints
- each constraint evaluated in the context of the schedd and job ad; any failure causes the whole submission to fail
- evaluated in listed order
- rejection (error) message may be customized

Submit Requirements Example

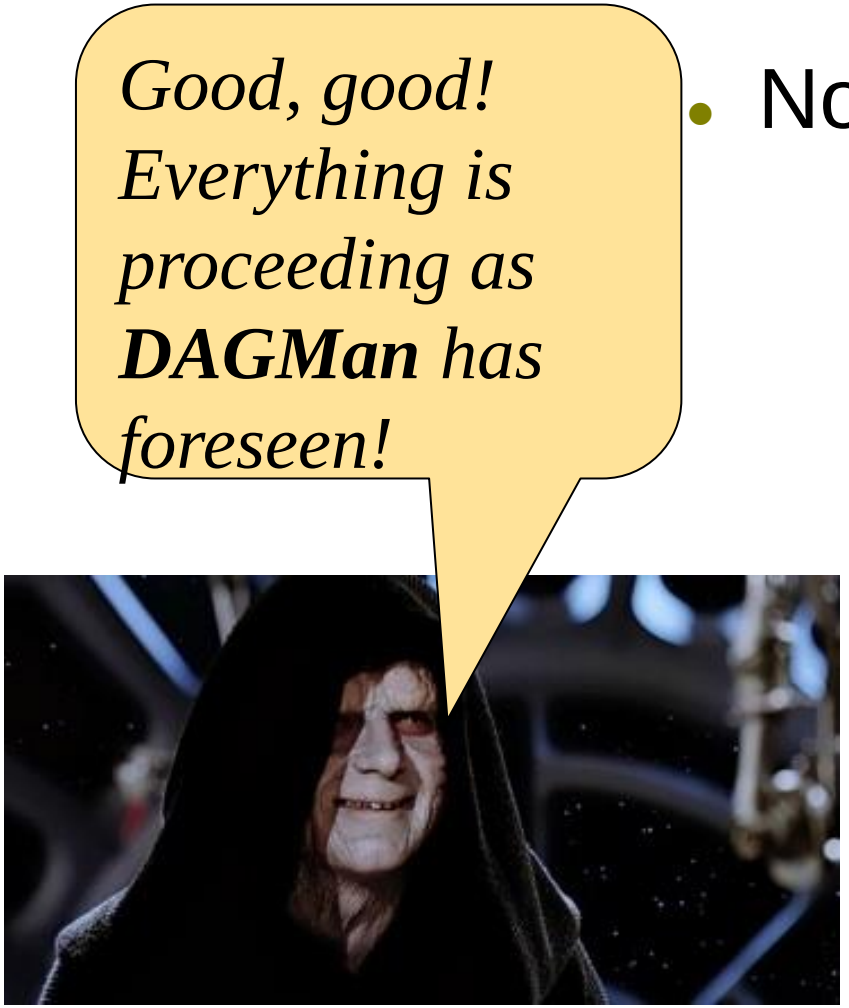
```
SUBMIT_REQUIREMENT_NAMES =
   NotStdUniv, MinimalRequestMemory
SUBMIT_REQUIREMENT_NotStdUniv =
   JobUniverse != 1
SUBMIT_REQUIREMENT_MinimalRequestMemory =
   RequestMemory > 512
SUBMIT_REQUIREMENT_NotStdUniv_REASON =
   "This pool doesn't do standard universe."
```

Questions on periodic file transfer or submit requirements?

DAGMan changes since last year

- PRE/POST script retry after delay (DEFER option)
- DAGMan handles submit file “foreach” syntax
- Configuration:
  - Maxpre, maxpost default to 20 (was 0)
  - Maxidle defaults to 1000 (was 0)
  - Fixed DAGMan entries in param table

DAGMan changes (cont)



- Node status file:
  - Format is now ClassAds
  - More info (retry number, procs queued and held for each node)
  - Fixed bug: final DAG status not always recorded correctly
  - ALWAYS-UPDATE option
  - Now works on Windows

DAGMan changes (cont)

- **`dagman.out`** file:
  - Node job hold reason in **`dagman.out`**
  - DAG_Status in **`dagman.out`**
- **`-DoRecovery`** command-line option
- Stricter checking of SPLICE syntax
- No (unused) command socket
- Stork no longer supported

HTCondor RPM Packaging

- More Standard Packaging
  - Matches OSG and Fedora package layout
  - Built with rpmbuild
  - Source RPM is released
    - Can rebuild directly from the source RPM
    - Build requirements are enforced by rpmbuild
  - Partitioned into several binary RPMs
    - Pick and choose what you need

HTCondor Binary RPM Packages

RPM	Description
condor	Base package
condor-all	Includes all the packages in a typical installation
condor-bosco	BOSCO – Manage jobs on remote clusters
condor-classads	HTCondor classified advertisement language
condor-classads-devel	Development support for classads
condor-debuginfo	Symbols for libraries and binaries
condor-externals	External programs and scripts
condor-externals-libs	External libraries
condor-kbdd	HTCondor Keyboard Daemon
condor-procd	HTCondor Process Tracking Daemon
condor-python	Python Bindings for HTCondor
condor-static-shadow	Static Shadow (Use 32-bit shadow on 64-bit system)
condor-std-universe	Standard Universe Support
condor-vm-gahp	VM Universe Support

HTCondor Debian Packaging

› More Standard Packaging
  - Matches debian package layout
  - Built with pbuilder
  - Source package is released

deb	Description
condor	Base Package
condor-dbg	Symbols for libraries and programs
condor-dev	Development files for HTCondor
condor-doc	HTCondor documentation
libclassad-dev	Development files for Classads
libclassad7	Classad runtime libaries

Statistics

- Time spent on DNS Lookups
  - Special counter for ‘slow’ lookups
- Counter for ResourceRequestsSent
- Per-user file transfer stats in Submitter ads
- New knob acts a whitelist for Statistics publication to the Collector


FUTURE

What to do with all these statistics?

- Aggregate and send them to Ganglia!
  - condor_gangliad introduced in v8.2
- In addition (or instead) of sending to Ganglia, aggregate and make available in JSON format over HTTP
- View some basic historical usage out-of-the-box by pointing web browser at central manager (modern CondorView)…
- ….Or upload JSON to influxdb, couchdb, …

› Moving/Caching Job Input Data

- Full session on data management right after lunch today!

› Security Credentials

- Kerberos Ticket Management and Delegation

More Schedd Scalability

- *Late materialization* of jobs in the schedd to enable submission of very large sets of jobs, e.g.

```
queue 1000000
```

  - More jobs materialized once number of idle jobs drops below a threshold (like DAGMan throttling)
- No “walking” of the job queue
  - Internally means more indexes, priority queues, aggregates

New condor_q default output

- Proposed new default output of condor_q will show summary of current users jobs.

```
-- Submitter: adam      Schedd: submit-3.chtc.wisc.edu
OWNER      IDLE RUNNING   HELD  SUBMITTED  DESCRIPTION  JOBIDs
adam          -       1      -  3/22 07:20 DAG: 221546  230864.0
              -       -      1  3/23 08:57 AtlasAnlysis 263203.0
              -       1      -  3/27 09:37 matlab.exe   307333.0
            133      21      -  3/27 11:46 DAG: 311986  312342.0 ... 313304.0

In the last 20 minutes:
    0 Job(s) were Completed
    5 Job(s) were Started                                312690.0 ... 312695.0
    1 Job(s) were Held                                   263203.0
        263203.0  5/11 07:22 Error from slot1@eee.chtc.wisc.edu: out of disk
```

Native OpenStack Support

- Speak OpenStack's NOVA protocol
  - Better supported than EC2 compatibility interface
  - Allows better error handling
  - Provides richer set of controls on instances
  - Potential to obtain and manage resources beyond servers

Partitionable Slots (Pslots)

- Partitionable Slots (Pslots) contains unclaimed machine resources
- Dynamic slots (Dslots) are created with enough resources to run a series of jobs; Dslots can't be resized, split, or merged
- When the schedd is done using a Dslot, its resources are returned to the unclaimed Pslot and the Dslot is destroyed.
- Can easily lead to starvation of larger jobs

Current Solution: Draining

- condor_drain <machine>
  - No new jobs may start until all jobs gracefully evicted from the machine and all resources returned to pslot
- condor_defrag daemon selects machines for draining
  - Doesn't use job mix for decisions on
    - How many machines to drain
    - Which machines to drain
    - Which users/jobs should get drained machines

Better options to condor_defrag

- We're looking for better solutions
- Currently considering two options
  - Directed Draining
  - Pslot Claiming

I am altering the slot. Pray I don't alter it any further!

Directed Draining

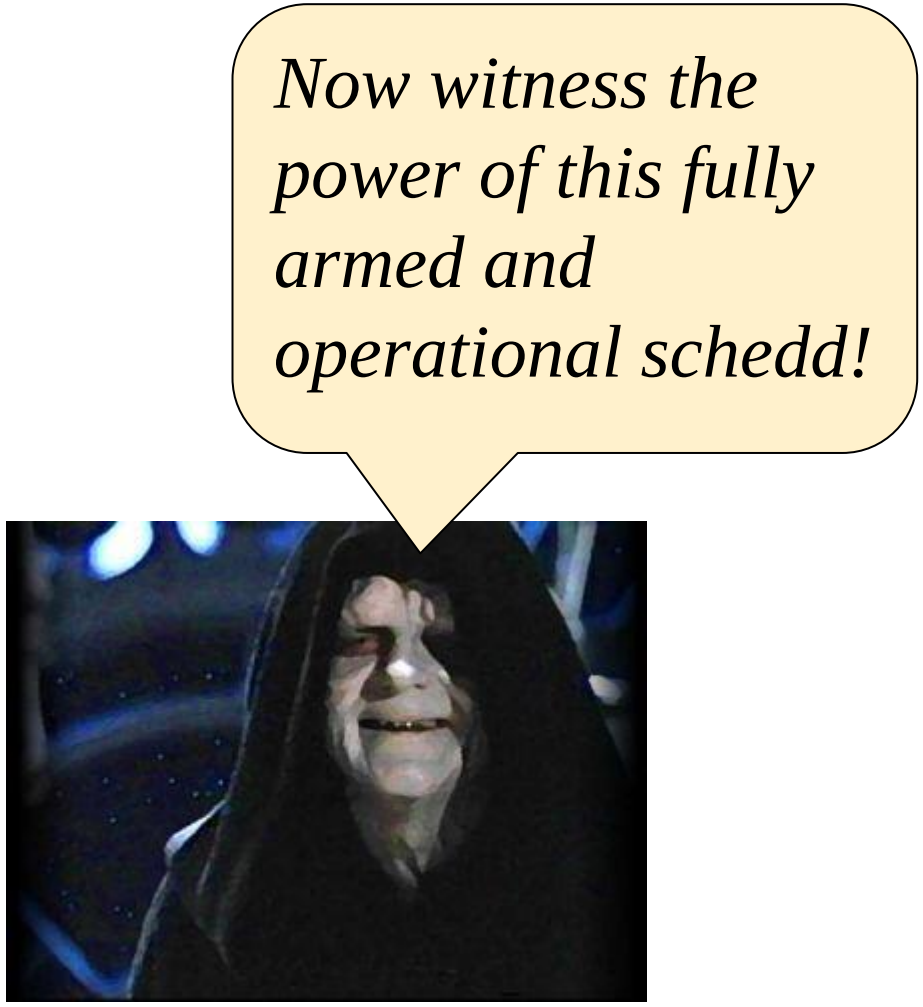
- Negotiator considers all resources of machine when matching (pslot + dslots)
  - Publishes information about how many more-desirable jobs would match each machine if drained
- condor_defrag daemon can use this information when deciding how many machines and which machines to drain

Pslot Claiming

- Whole machines are assigned to users by negotiator
  - Pslot is claimed by schedd
- Removes need for condor_defrag, as schedd divides pslot to run jobs
  - Can redivide as needed to run different sized jobs
  - Can sublet unused resources
  - Can quickly evict subletters

Pslot claiming, cont.

- More scalable to do matchmaking at the level of the machine.
- More power to the schedd, which can be scaled horizontally.



Questions on Partitionable Slot Changes?

or OpenStack support?

Thank You!


Please help us, high throughput computing. You're our only hope!