HTCondor Administration Basics

Greg Thain
Center for High Throughput Computing

Overview

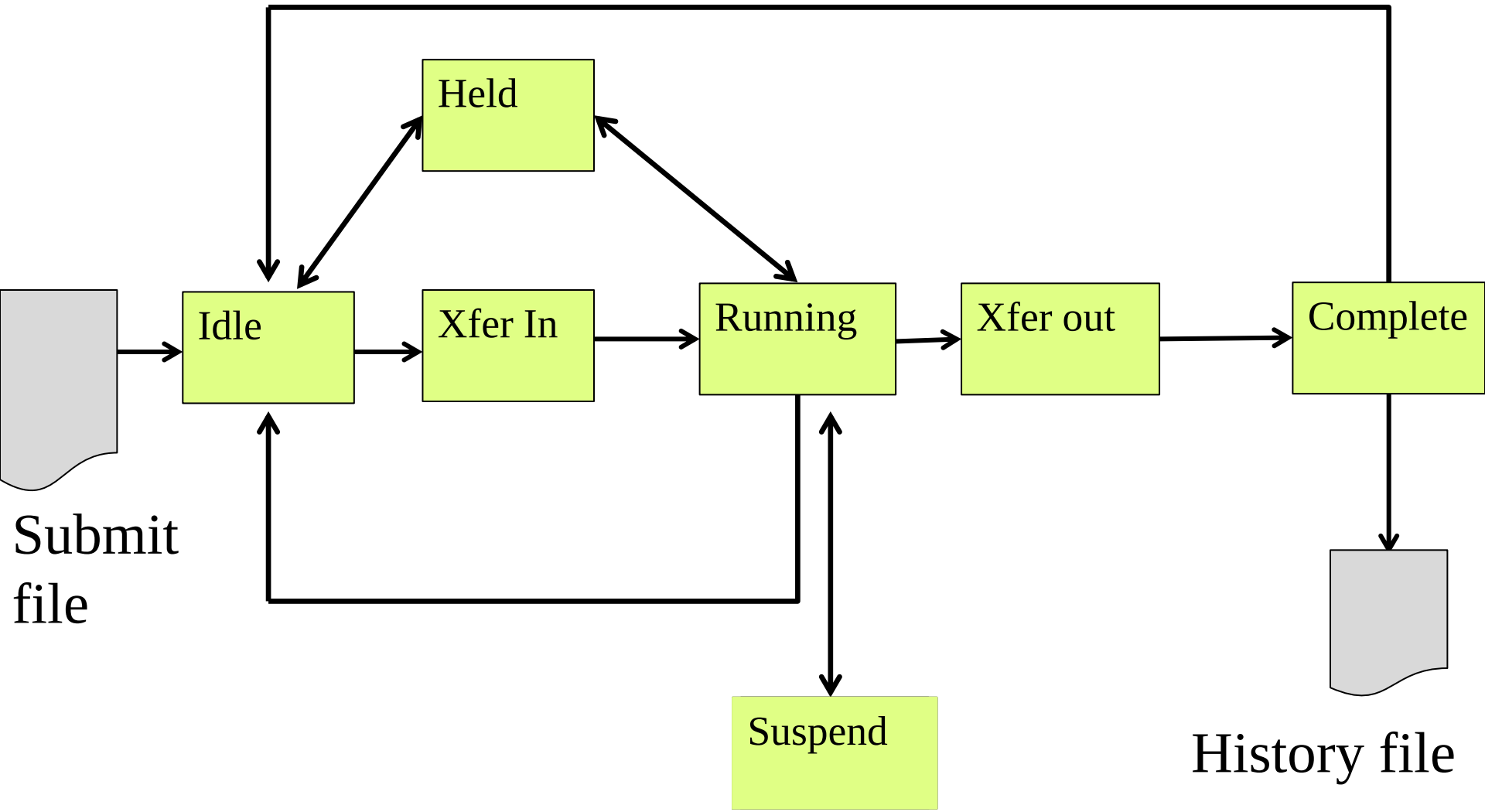
- HTCondor Architecture Overview
- Configuration and other nightmares
- Setting up a personal condor
- Setting up distributed condor
- Monitoring
- Rules of Thumb

Two Big HTCondor Abstractions

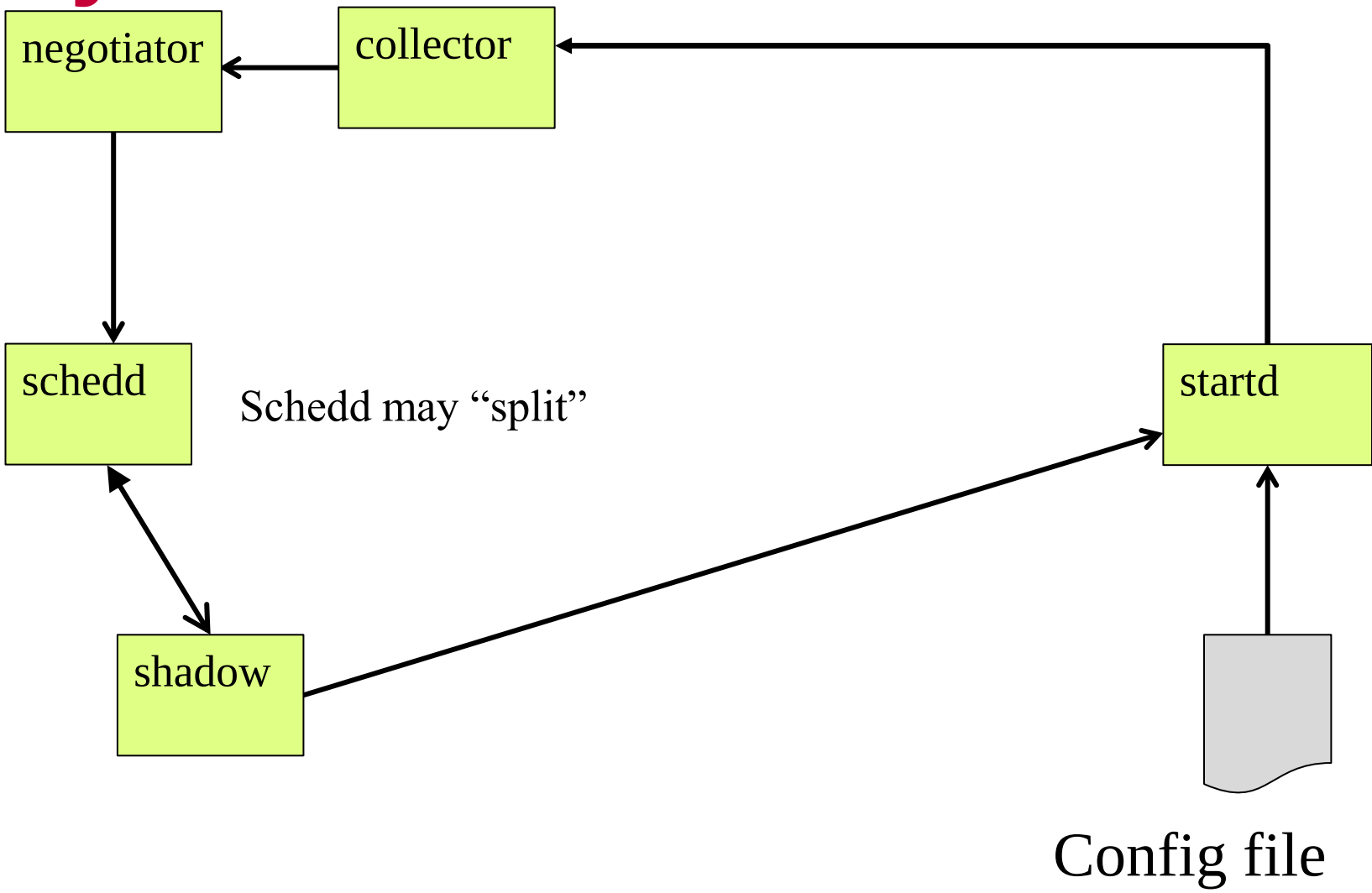
- Jobs

- Machines

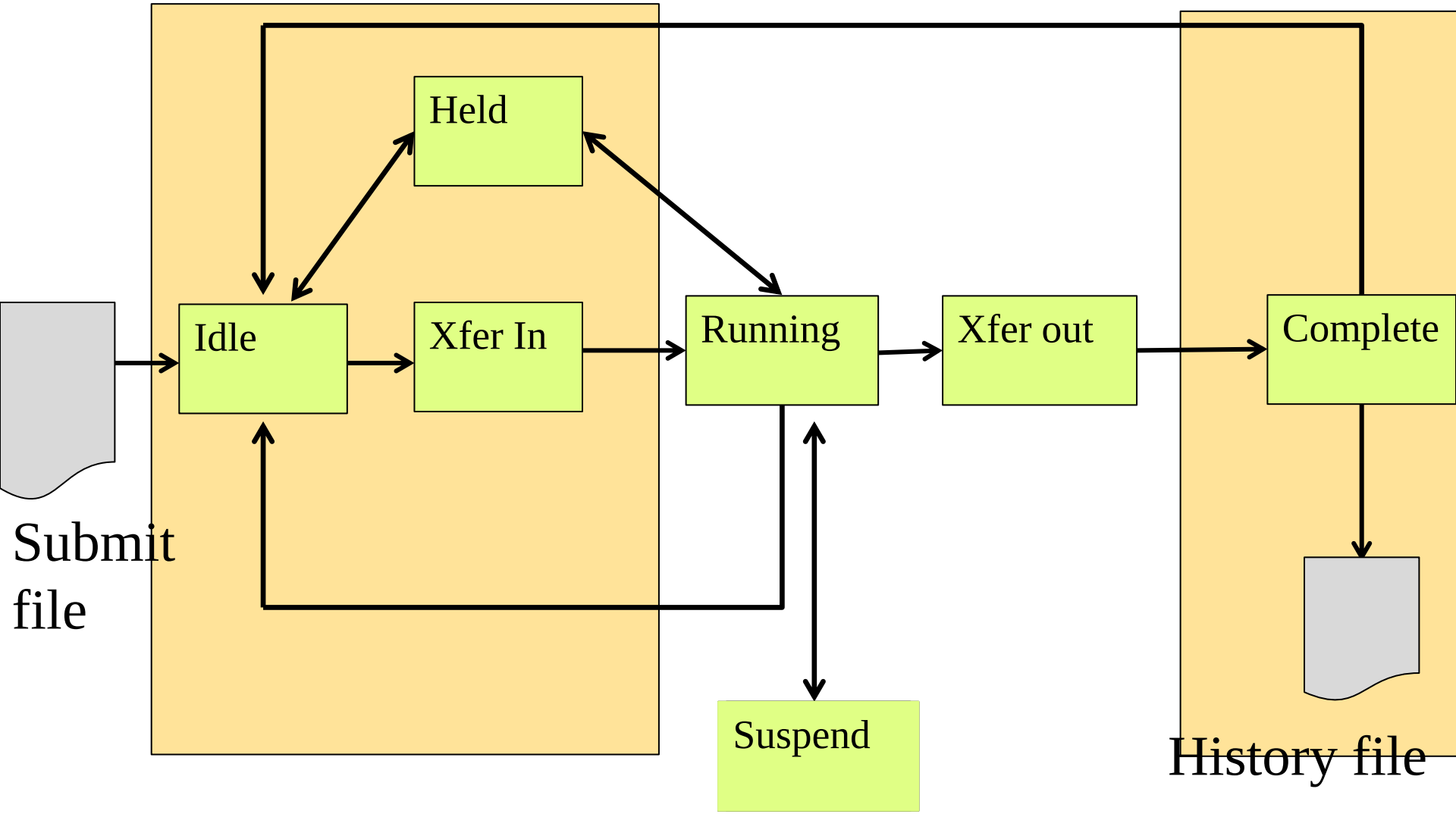
Life cycle of HTCondor Job



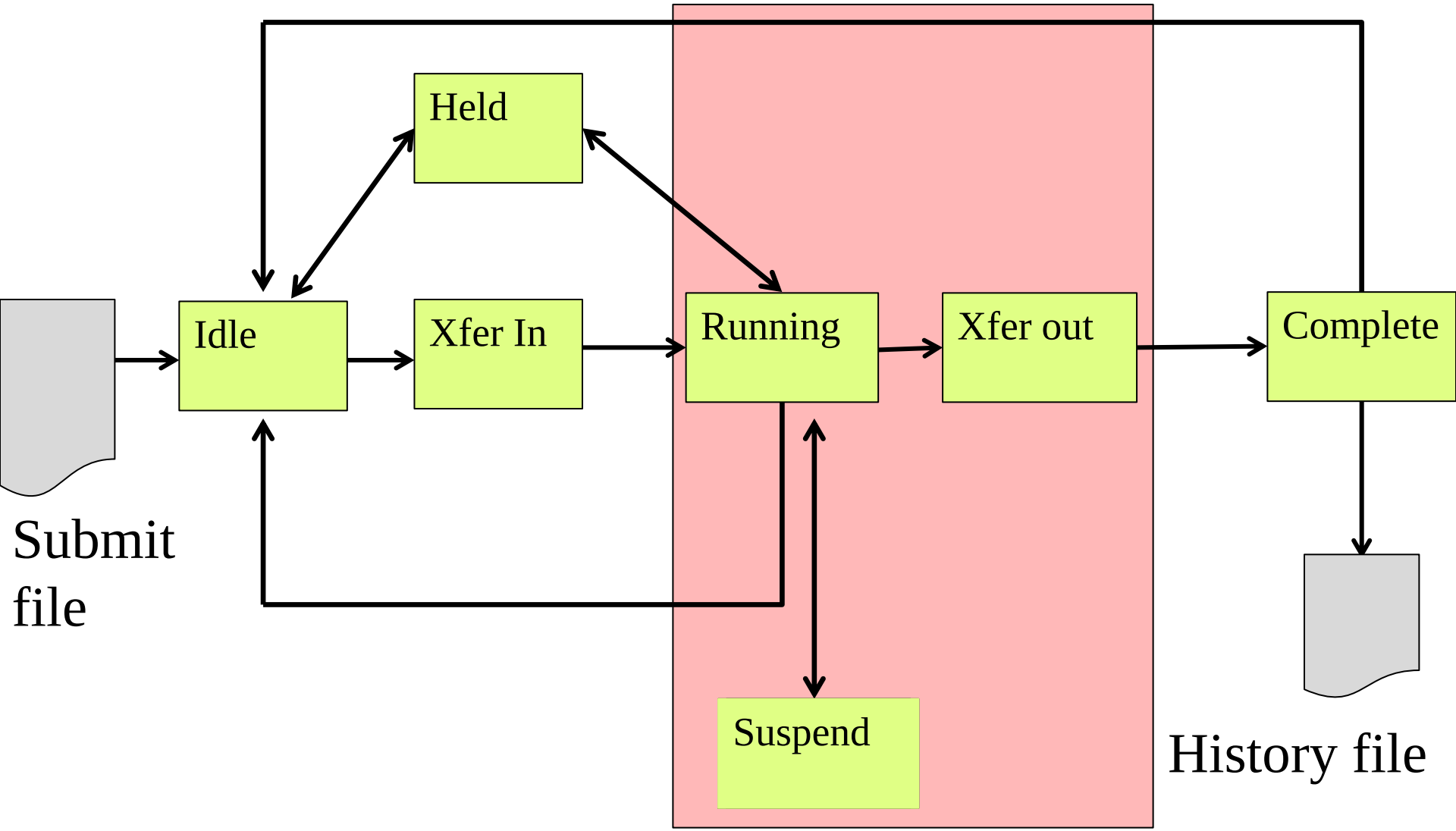
Life cycle of HTCondor Machine



“Submit Side”

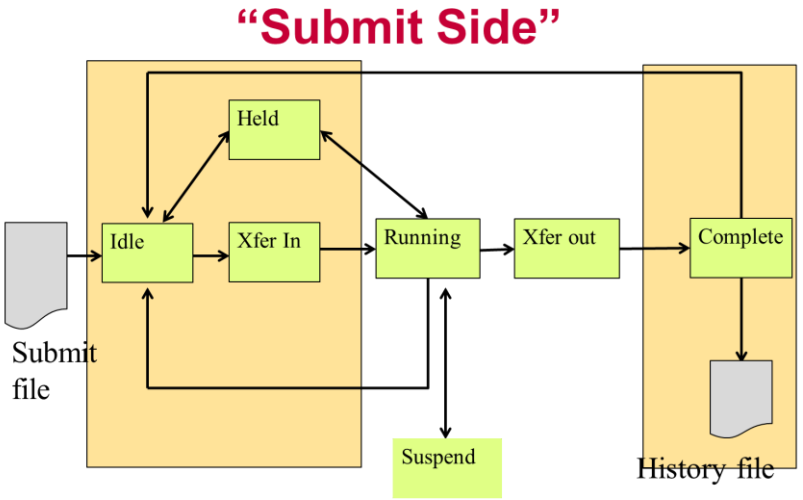

Held
Idle
Xfer In
Running
Xfer out
Complete
Submit file
Suspend
History file

“Execute Side”



The submit side

- Submit side managed by 1 condor_schedd process
- And one shadow per running job
  - condor_shadow process
- The Schedd is a database



5

- Submit points can be performance bottleneck

- Usually a handful per pool

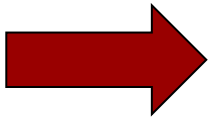
In the Beginning…

```
universe = vanilla
executable = compute
request_memory = 70M
arguments = $(ProcID)
should_transfer_input = yes
output = out.$(ProcID)
error = error.$(ProcId)
+IsVerySpecialJob = true
Queue
```

HTCondor Submit file

From submit to schedd

```
universe = vanilla
executable = compute
request_memory = 70M
arguments = $(ProcID)
should_transfer_input = yes
output = out.$(ProcID)
error = error.$(ProcId)
+IsVerySpecialJob = true
Queue
```



```
JobUniverse = 5
Cmd = "compute"
Args = "0"
RequestMemory = 70000000
Requirements = Opsys == "Li..
DiskUsage = 0
Output = "out.0"
IsVerySpecialJob = true
```

condor_submit submit_file

- Submit file in, Job classad out
- Sends to schedd
- man condor_submit for full details

Other ways to talk to schedd

- Python bindings, SOAP, wrappers (like DAGman)

Condor_schedd holds all jobs

One pool, Many schedds

condor_submit –name
    chooses
Owner Attribute:
    need authentication
Schedd also called “q”
    not actually a queue

```
JobUniverse = 5
Owner = “gthain”
JobStatus = 1
NumJobStarts = 5
Cmd = “compute”
Args = “0”
RequestMemory = 70000000
Requirements = Opsys == “Li..
DiskUsage = 0
Output = “out.0”
IsVerySpecialJob = true
```

Condor_schedd has all jobs

- In memory (big)
  - condor_q expensive
- And on disk
  - Fsync's often
  - Monitor with linux
- Attributes in manual
- condor_q –l job.id
  - e.g. condor_q –l 5.0

```
JobUniverse = 5
Owner = "gthain"
JobStatus = 1
NumJobStarts = 5
Cmd = "compute"
Args = "0"
RequestMemory = 70000000
Requirements = Opsys == "Li..
DiskUsage = 0
Output = "out.0"
IsVerySpecialJob = true
```

What if I don't like those Attributes?

- Write a wrapper to condor_submit
- submit_exprs
- condor_qedit

Configuration of Submit side

- Not much policy to be configured in schedd

- Mainly scalability and security
- MAX_JOBS_RUNNING
- JOB_START_DELAY
- MAX_CONCURRENT_DOWNLOADS
- MAX_JOBS_SUBMITTED

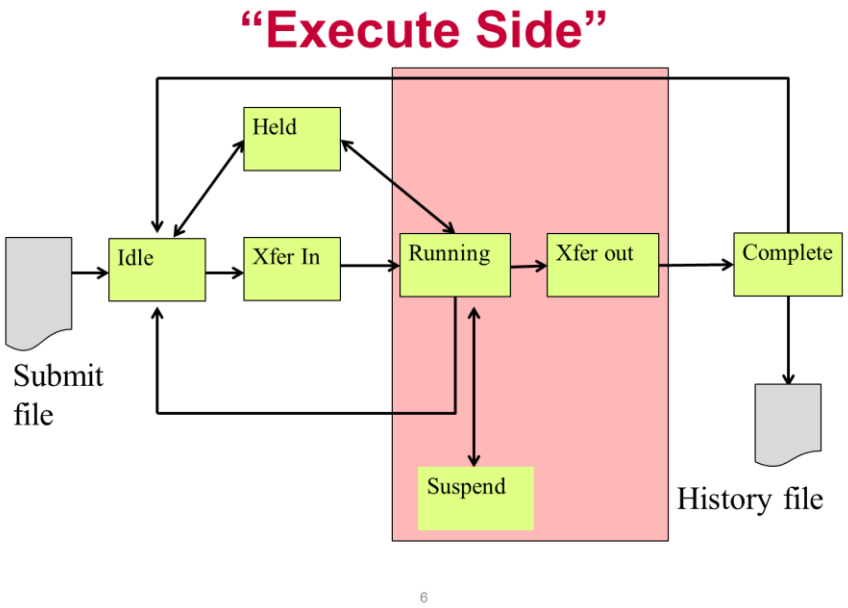
The Execute Side

Primarily managed by condor_startd process

With one condor_starter per running jobs

Sandboxes the jobs

Usually many per pool (support 10s of thousands)



Startd also has a classad

- Condor makes it up
  - From interrogating the machine
  - And the config file
  - And sends it to the collector
- condor_status [-l]
  - Shows the ad
- condor_status –direct daemon
  - Goes to the startd

Condor_status –l machine

```
OpSys = "LINUX“
CustomGregAttribute = “BLUE”
OpSysAndVer = "RedHat6"
TotalDisk = 12349004
Requirements = ( START )
UidDomain = “cheesee.cs.wisc.edu"
Arch = "X86_64"
StartdIpAddr = "<128.105.14.141:36713>"
RecentDaemonCoreDutyCycle = 0.000021
Disk = 12349004
Name = "slot1@chevre.cs.wisc.edu"
State = "Unclaimed"
Start = true
Cpus = 32
Memory = 81920
```

One Startd, Many slots

- HTCondor treats multicore as independent slots
- Start can be configured to:
  - Only run jobs based on machine state
  - Only run jobs based on other jobs running
  - Preempt or Evict jobs based on polic
- A whole talk just on this

Configuration of startd

- Mostly policy, whole talk on that
- Several directory parameters
- EXECUTE – where the sandbox is

- CLAIM_WORKLIFE
  - How long to reuse a claim for different jobs

The “Middle” side

- There’s also a “Middle”, the Central Manager:
  - A condor_negotiator
    - Provisions machines to schedds
  - A condor_collector
    - Central nameservice: like LDAP
- Please don’t call this “Master node” or head
- Not the bottleneck you may think: stateless

Responsibilities of CM

- Much scheduling policy resides here

- Scheduling of one user vs another
- Definition of groups of users
- Definition of preemption

The condor_master

- Every condor machine needs a master
- Like “~~systemd~~”, or “init”
- Starts daemons, restarts crashed daemons

Quick Review of Daemons

condor_master: runs on all machine, always

condor_schedd: runs on submit machine

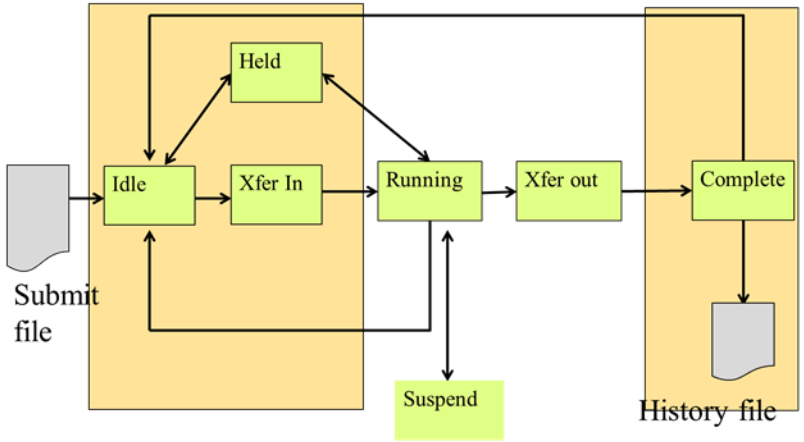
  condor_shadow: one per job

condor_startd: runs on execute machine

   condor_starter: one per job

condor_negotiator/condor_collector

Process View "Submit Side"



condor_master (pid: 1740)

condor_procd

fork/exec

condor_schedd

"Condor Kernel"

condor_q

condor_submit

"Tools"

fork/exec

condor_shadow

condor_shadow

condor_shadow

"Condor Userspace"

Process View: Execute

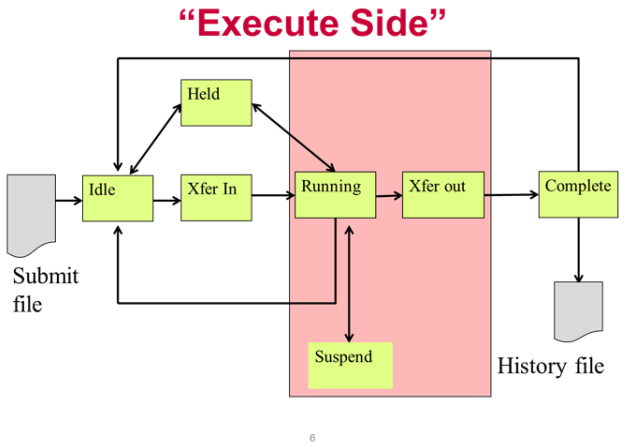
condor_master
(pid: 1740)

condor_procd

fork/exec

condor_startd

"Condor Kernel"



condor_status -direct "Tools"

condor_starter

condor_starter

condor_starter

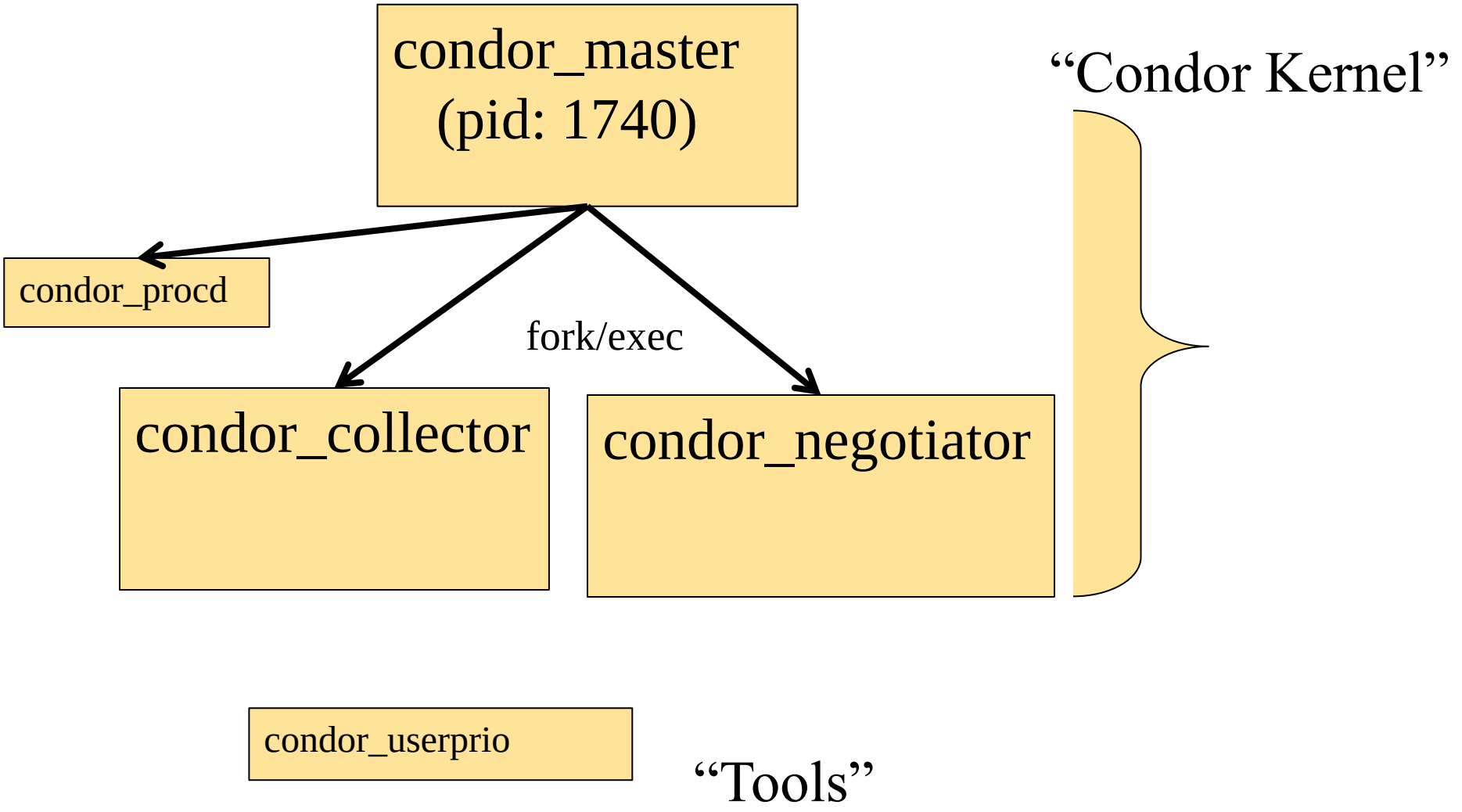
"Condor Userspace"

Job

Job

Job

Process View: Central Manager



Condor Admin Basics

Assume that HTCondor is installed

- Either with tarball
  - tar xvf htcondor-8.2.3-redhat6
- Or native packages
  - yum install htcondor

Let's Make a Pool

- First need to configure HTCondor
- 1100+ knobs and parameters!
- Don't need to set all of them…

Default file locations

```
BIN = /usr/bin
SBIN = /usr/sbin
LOG = /var/condor/log
SPOOL = /var/lib/condor/spool
EXECUTE = /var/lib/condor/execute
CONDOR_CONFIG =
/etc/condor/condor_config
```

Configuration File

- **(Almost)** all configure is in files, “root” is
- **CONDOR_CONFIG env var**
- **/etc/condor/condor_config**
- This file points to others
- All daemons share same configuration
- Might want to share between all machines (NFS, automated copies, puppet, etc)

Configuration File Syntax

```
# I'm a comment!
CREATE_CORE_FILES=TRUE
MAX_JOBS_RUNNING = 50
# HTCondor ignores case:
log=/var/log/condor
# Long entries:
collector_host=condor.cs.wisc.edu,\
    secondary.cs.wisc.edu
```

Other Configuration Files

- **LOCAL_CONFIG_FILE**
  - Comma separated, processed **in order**

```
LOCAL_CONFIG_FILE = \
   /var/condor/config.local,\
/shared/condor/config.$(OPSYS)
```

- **LOCAL_CONFIG_DIR**
  - **Files processed IN ORDER**

```
LOCAL_CONFIG_DIR = \
   /etc/condor/config.d
```

Configuration File Macros

- You reference other macros (settings) with:
  - `A = $(B)`
  - `SCHEDD = $(SBIN)/condor_schedd`
- Can create additional macros for organizational purposes

Configuration File Macros

- Can append to macros:

```
A=abc
A=$(A),def
```

- Don't let macros recursively define each other!

```
A=$(B)
B=$(A)
```

Configuration File Macros

- Later macros in a file overwrite earlier ones
  - B will evaluate to 2:

```
A=1
B=$(A)
A=2
```

Config file defaults

- CONDOR_CONFIG “root” config file:
  - /etc/condor/condor_config
- Local config file:
  - /etc/condor/condor_config.local
- Config directory
  - /etc/condor/config.d

Config file recommendations

- For “system” condor, use default
  - Global config file read-only
    - /etc/condor/condor_config
  - All changes in config.d small snippets
    - /etc/condor/config.d/05some_example
  - All files begin with 2 digit numbers

- Personal condors elsewhere

condor_config_val

- condor_config_val [-v] <KNOB_NAME>
  - Queries config files
- condor_config_val –set name value
- condor_config_val -dump

- Environment overrides:
- export _condor_KNOB_NAME=value
  - Trumps all others (so be careful)

condor_reconfig

- Daemons long-lived
  - Only re-read config files condor_reconfig command
  - Some knobs don't obey reconfig, require restart
    - DAEMON_LIST, NETWORK_INTERFACE
- condor_restart

Macros and Expressions Gotcha

- These are simple replacement macros
- Put parentheses around expressions

```
TEN=5+5
HUNDRED=$(TEN)*$(TEN)
```

  - HUNDRED becomes 5+5*5+5 or 35!

```
TEN=(5+5)
HUNDRED=($(TEN)*$(TEN))
```

  - ((5+5)*(5+5)) = 100

Got all that?

Let's make a pool!

- "Personal Condor"
  - All on one machine:
    - submit side IS execute side
  - Jobs always run
- Use defaults where ever possible
- Very handy for debugging and learning

Minimum knob settings

- DAEMON_LIST
  - What daemons run on this machine
- CONDOR_HOST
  - Where the central manager is
- Security settings
  - Who can do what to whom?

Other interesting knobs

`LOG = /var/log/condor`

Where daemons write debugging info

`SPOOL = /var/spool/condor`

Where the schedd stores jobs and data

`EXECUTE = /var/condor/execute`

Where the startd runs jobs

Minimum knobs for personal Condor

- In `/etc/condor/config.d/50PC.config`

```
# GGT: DAEMON LIST for PC has everyone
DAEMON_LIST = MASTER, NEGOTIATOR,\
     COLLECTOR, SCHEDD, STARTD

CONDOR_HOST = localhost
ALLOW_WRITE = localhost
```

Does it Work?

```
$ condor_status
Error: communication error
CEDAR:6001:Failed to connect to <128.105.14.141:4210>

$ condor_submit
ERROR: Can't find address of local schedd

$ condor_q
Error:
Extra Info: You probably saw this error because the
condor_schedd is not running on the machine you are
trying to query…
```

Checking…

```
$ ps auxww | grep [Cc]ondor
$
```

Starting Condor

- condor_master –f
- service start condor

```
$ ps auxww | grep [Cc]ondor
$
condor  19534  50380          Ss   11:19   0:00 condor_master
root   19535  21692           S    11:19   0:00 condor_procd -A
/scratch/gthain/personal-condor/log/procd_pipe -L
/scratch/gthain/personal-condor/log/ProcLog -R 1000000 -S 60 -D -C
28297
condor   19557  69656        Ss   11:19   0:00 condor_collector -f
condor   19559  51272        Ss   11:19   0:00 condor_startd -f
condor   19560  71012        Ss   11:19   0:00 condor_schedd -f
condor   19561  50888        Ss   11:19   0:00 condor_negotiator -f
```

Notice the UID of the daemons

Quick test to see it works

```
$ condor_status
# Wait a few minutes…
$ condor_status
Name                OpSys      Arch   State     Activity LoadAv Mem

slot1@chevre.cs.wi LINUX      X86_64 Unclaimed Idle      0.190 20480
slot2@chevre.cs.wi LINUX      X86_64 Unclaimed Idle      0.000 20480
slot3@chevre.cs.wi LINUX      X86_64 Unclaimed Idle      0.000 20480
slot4@chevre.cs.wi LINUX      X86_64 Unclaimed Idle      0.000 20480

-bash-4.1$ condor_q
-- Submitter: gthain@chevre.cs.wisc.edu : <128.105.14.141:35019> :
chevre.cs.wisc.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
$ condor_restart # just to be sure…
```

Some Useful Startd Knobs

- `NUM_CPUS = X`
  - How many cores condor thinks there are
- `MEMORY = M`
  - How much memory (in Mb) there is
- `STARTD_CRON_...`
  - Set of knobs to run scripts and insert attributes into startd ad (See Manual for full details).

Brief Diversion into daemon logs

- Each daemon logs mysterious info to file
- $(LOG)/DaemonNameLog
- Default:
  - /var/log/condor/SchedLog
  - /var/log/condor/MatchLog
  - /var/log/condor/StarterLog.slotX
- Experts-only view of condor

Let's make a "real" pool

- Distributed machines makes it hard
  - Different policies on each machines
  - Different owners
  - Scale

Most Simple Distributed Pool

- Requirements:
  - No firewall
  - Full DNS everywhere (forward and backward)
  - We've got root on all machines
- HTCondor doesn't require any of these
  - (but easier with them)

What UID should jobs run as?

- Three Options (all require root):
  - Nobody UID
    - Safest from the machine's perspective
  - The submitting User
    - Most useful from the user's perspective
    - May be required if shared filesystem exists
  - A "Slot User"
    - Bespoke UID per slot
    - Good combination of isolation and utility

UID_DOMAIN SETTINGS

```
UID_DOMAIN = \
same_string_on_submit
TRUST_UID_DOMAIN = true
SOFT_UID_DOMAIN = true
```

If UID_DOMAINs match, jobs run as user, otherwise “nobody”

Slot User

```
SLOT1_USER = slot1
SLOT2_USER = slot2
…
STARTER_ALOW_RUNAS_OWNER = false
EXECUTE_LOGIN_IS_DEDICATED=true
```

Job will run as slotX Unix user

FILESYSTEM_DOMAIN

- HTCondor can work with NFS
  - But how does it know what nodes have it?
- WhenSubmitter & Execute nodes share
  - `FILESYSTEM_DOMAIN` values
    - e.g `FILESYSTEM_DOMAIN = domain.name`
- Or, submit file can always transfer with
  - `should_transfer_files = yes`
- If jobs always idle, first thing to check

3 Separate machines

- Central Manager
- Execute Machine
- Submit Machine

Central Manager

/etc/condor/config.d/51CM.local:

```
DAEMON_LIST = MASTER, COLLECTOR,\
    NEGOTIATOR
CONDOR_HOST = cm.cs.wisc.edu
# COLLECTOR_HOST = \
    $(CONDOR_HOST):1234
ALLOW_WRITE = *.cs.wisc.edu
```

Submit Machine

/etc/condor/config.d/51Submit.local:

```
DAEMON_LIST = MASTER, SCHEDD
CONDOR_HOST = cm.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
UID_DOMAIN = cs.wisc.edu
FILESYSTEM_DOMAIN = cs.wisc.edu
```

Execute Machine

/etc/condor/config.d/51Execute.local:

```
DAEMON_LIST = MASTER, STARTD
CONDOR_HOST = cm.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
UID_DOMAIN = cs.wisc.edu
FILESYSTEM_DOMAIN = cs.wisc.edu
FILESYSTEM_DOMAIN = $(FULLHOSTNAME)
```

Now Start them all up

- Does order matter?
  - Somewhat: start CM first
- How to check:
- Every Daemon has classad in collector
  - condor_status –schedd
  - condor_status –negotiator
  - condor_status -all

condor_status -any

```
MyType              TargetType          Name

Collector           None                Test Pool@cm.cs.wisc.edu
Negotiator          None                cm.cs.wisc.edu
DaemonMaster        None                cm.cs.wisc.edu
Scheduler           None                submit.cs.wisc.edu
DaemonMaster        None                submit.cs.wisc.edu
DaemonMaster        None                wn.cs.wisc.edu
Machine             Job                 slot1@wn.cs.wisc.edu
Machine             Job                 slot2@wn.cs.wisc.edu
Machine             Job                 slot3@wn.cs.wisc.edu
Machine             Job                 slot4@wn.cs.wisc.edu
```

Debugging the pool

- condor_q / condor_status
- condor_ping ALL –name machine
- Or
- condor_ping ALL –addr ‘<127.0.0.1:9618>’

What if a job is always idle?

- Check userlog – may be preempted often
- run condor_q –better_analyze job_id

Whew!

Monitoring: Jobs and Daemons

The Wrong way

- condor_q
  - Accurate
  - Slow
  - Talks to the single-threaded schedd
  - Output subject to change

The (still) Wrong way

- condor_q –format “%d” SomeAttribute
  - Output **NOT** subject to change
  - Only sends minimal data over the wire

A quick and dirty shortcut

```
$ condor_status -submitter Name                              Machine
RunningJobs IdleJobs HeldJobs

c4e4user@rcac.purdue.edu     carter-ssg.rcac.pu          0        7        0
beissinger@chtcsubmit.ansci. chtcsubmit.ansci.w          0        0       48
fabiane@chtcsubmit.ansci.wis chtcsubmit.ansci.w          0        2        8
gthain@cm.chtc.wisc.edu      cm.chtc.wisc.edu            0        0        0
group_cmspilot.osg_cmsuser19 cmsgrid01.hep.wisc          0        0        0
group_cmspilot.osg_cmsuser20 cmsgrid01.hep.wisc          0        1        0
group_cmspilot.osg_cmsuser20 cmsgrid02.hep.wisc          0        1        0
chen436@rcac.purdue.edu      coates-fe00.rcac.p          0        1        0
```

condor_userprio

- Command usage:

```
condor_userprio –most
```

```
                                     Effective  Priority
User Name                            Priority    Factor   In Use (wghted-hrs) Last Usage
---------------------------------------------- --------- ------ ----------- ----------
lmichael@submit-3.chtc.wisc.edu          5.00     10.00      0        16.37    0+23:46
blin@osghost.chtc.wisc.edu               7.71     10.00      0      5412.38    0+01:05
osgtest@osghost.chtc.wisc.edu           90.57     10.00     47     45505.99      <now>
cxiong36@submit-3.chtc.wisc.edu        500.00   1000.00      0         0.29    0+00:09
ojalvo@hep.wisc.edu                    500.00   1000.00      0    398148.56    0+05:37
wjiang4@submit-3.chtc.wisc.edu         500.00   1000.00      0         0.22    0+21:25
cxiong36@submit.chtc.wisc.edu          500.00   1000.00      0        63.38    0+21:42
```

For individual jobs

- User log file

```
001 (50288539.000.000) 07/30 16:39:50 Job executing on host:
<128.105.245.27:40435>
...
006 (50288539.000.000) 07/30 16:39:58 Image size of job updated: 1
        3  -  MemoryUsage of job (MB)
        2248  -  ResidentSetSize of job (KB)
...
004 (50288539.000.000) 07/30 16:40:02 Job was evicted.
        (0) Job was not checkpointed.
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
        41344  -  Run Bytes Sent By Job
        110  -  Run Bytes Received By Job
        Partitionable Resources :    Usage  Request Allocated
           Cpus                 :                1         1
           Disk (KB)            :       53       1   7845368
           Memory (MB)          :        3    1024      1024
```

Condor statistics

- Condor_status –direct –schedd –statistics 2
- (all kinds of output), mostly aggregated
- NumJobStarts, RecentJobStarts, etc.
- See manual for full details

DaemonCoreDutyCycle

- Most important statistic
- Measures time not idle
- If over 95%, daemon is probably saturated

Disaggregated stats

SCHEDD_COLLECT_STATS_FOR_Gthain = (Owner==“gthain")

Schedd will maintain distinct sets of status per owner, with name as prefix:

GthainJobsCompleted = 7

GthainJobsStarted = 100

Even better

```
SCHEDD_COLLECT_STATS_BY_Owner = Owner
```

For **all** owners, collect & publish stats:

gthainJobsStarted = 7

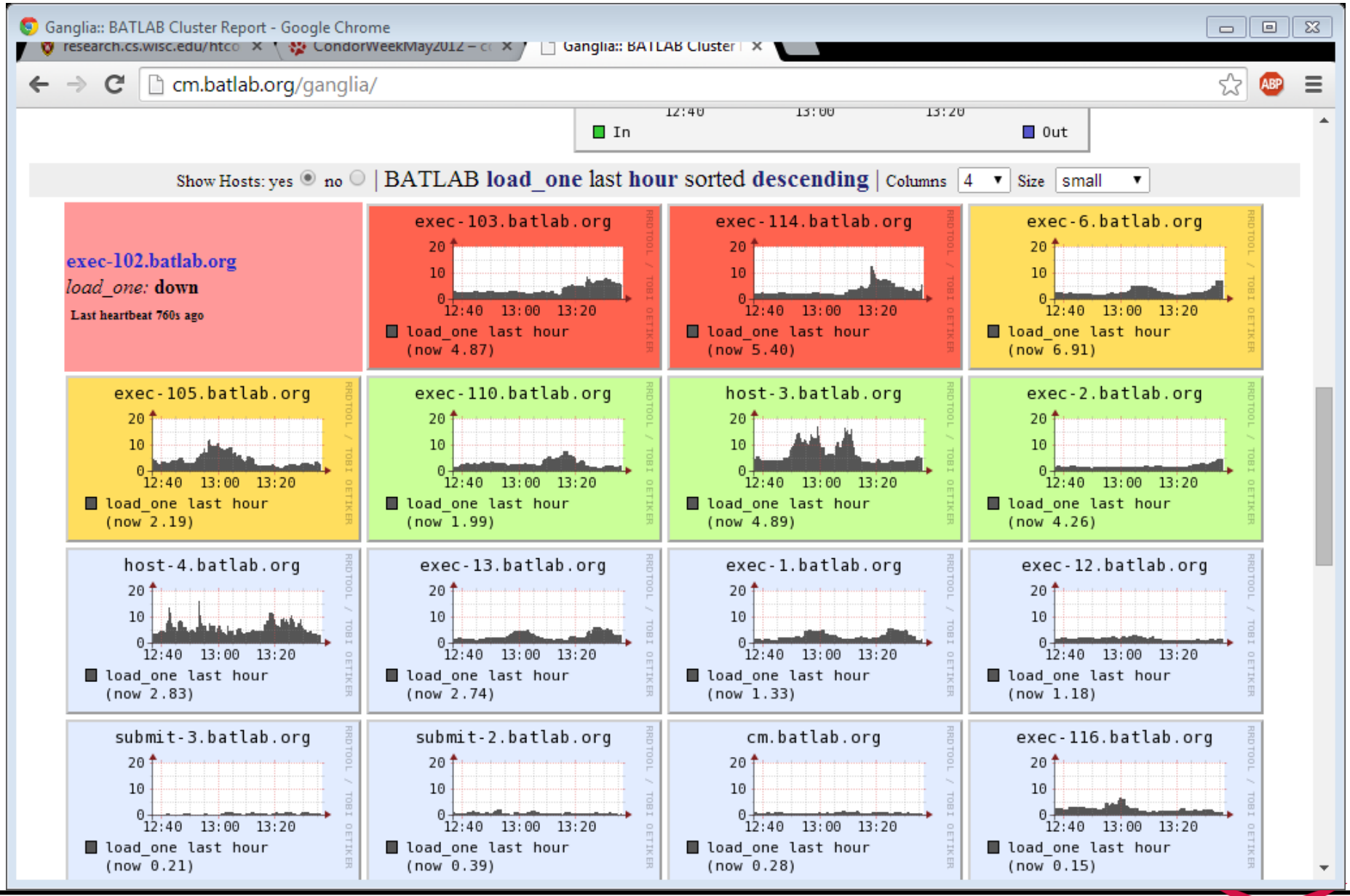
tannenbaJobsStarted = 100

Ganglia and condor_gangliad

- Condor_gangliad runs condor_status
- DAEMON_LIST = MASTER, GANGLIAD,…
- Forwards the results to gmond

- Which makes pretty pictures
- Gangliad has config file to map attributes to ganglia, and can aggregate:

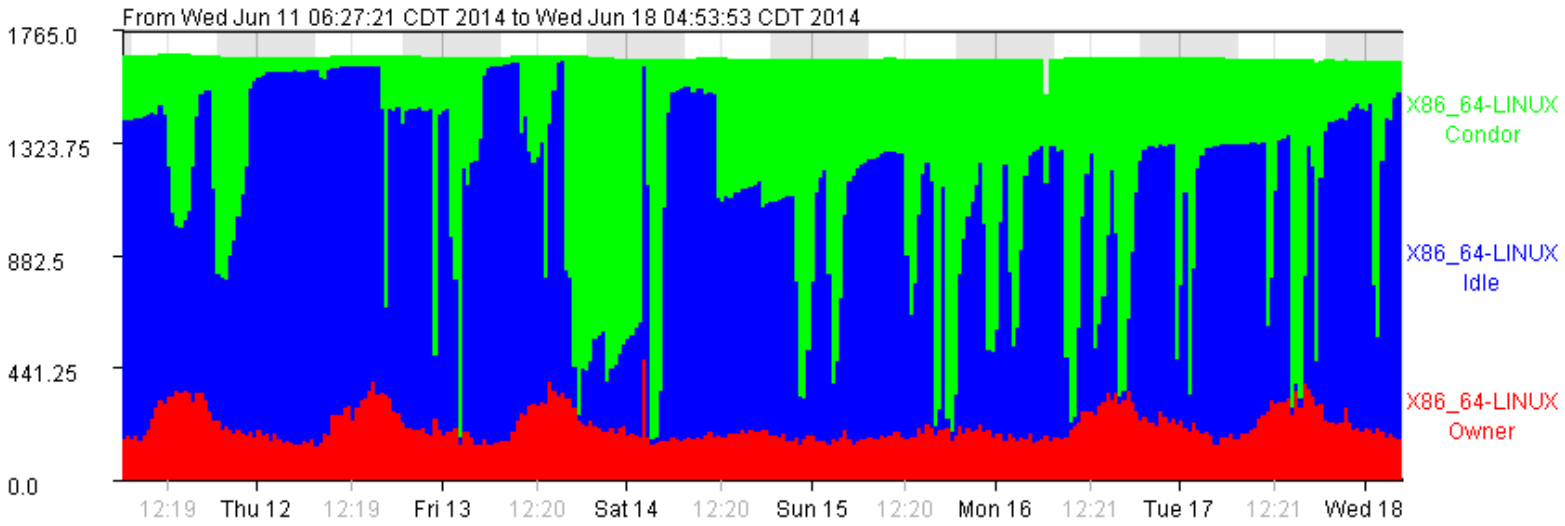
```
[
  Name    = "JobsStarted";
  Verbosity = 1;
  Desc    = "Number of job run attempts started";
  Units   = "jobs";
  TargetType = "Scheduler";
]
[
  Name    = "JobsSubmitted";
  Desc    = "Number of jobs submitted";
  Units   = "jobs";
  TargetType = "Scheduler";
]
```


Ganglia:: BATLAB Cluster Report - Google Chrome
cm.batlab.org/ganglia/
In
Out
Show Hosts: yes no | BATLAB load_one last hour sorted descending | Columns 4 Size small
exec-102.batlab.org
load_one: down
Last heartbeat 760s ago
exec-103.batlab.org
load_one last hour (now 4.87)
exec-114.batlab.org
load_one last hour (now 5.40)
exec-6.batlab.org
load_one last hour (now 6.91)
exec-105.batlab.org
load_one last hour (now 2.19)
exec-110.batlab.org
load_one last hour (now 1.99)
host-3.batlab.org
load_one last hour (now 4.89)
exec-2.batlab.org
load_one last hour (now 4.26)
host-4.batlab.org
load_one last hour (now 2.83)
exec-13.batlab.org
load_one last hour (now 2.74)
exec-1.batlab.org
load_one last hour (now 1.33)
exec-12.batlab.org
load_one last hour (now 1.18)
submit-3.batlab.org
load_one last hour (now 0.21)
submit-2.batlab.org
load_one last hour (now 0.39)
cm.batlab.org
load_one last hour (now 0.28)
exec-116.batlab.org
load_one last hour (now 0.15)

Condor View

- Contrib “plug in”
- See manual for installation instructions
- Requires java plugin
- Graphs state of machine in pool over time:

UW-Madison Comp Sci Condor Pool Machine Statistics for Week



[Graph Hints: The Y-axis is number of machines, the X-axis is time. When graph finishes updating, press "Configure.." to view different Architecture or State data. Also, you can use the mouse to draw a rectangle on the graph and then press "Zoom In". Press "Reset" to center/resize the data after Configure or when done zooming. Nighttime shows up on graph background as grey.]

Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak		
Total	228.2 (12.2%)	535.6 (28.9%)	1090.9 (58.9%)	0.0 (0.0%)	0.0 (0.0%)	522 (27%)	1506 (84%)
X86_64/WINDOWS	18.7 (10.2%)	16.0 (8.4%)	131.3 (81.4%)	0.0 (0.0%)	0.0 (0.0%)	247 (70%)	41 (12%)
X86_64/LINUX	207.8 (12.5%)	518.2 (31.3%)	929.8 (56.1%)	0.0 (0.0%)	0.0 (0.0%)	471 (28%)	1492 (90%)
[illegible]	1.7	1.4	29.9	0.0	0.0	9	15

Speeds, Feeds, Rules of Thumb

- HTCondor scales to 100,000s of machines
  - With a lot of work
  - Contact us, see wiki page

Without Heroics:

- Your Mileage may vary:
  - Shared FileSystem vs. File Transfer
  - WAN vs. LAN
  - Strong encryption vs none
  - Good autoclustering
- A single schedd can run at 50 Hz
- Schedd needs 500k RAM for running job
  - 50k per idle jobs
- Collector can hold tens of thousands of ads

Upgrading to new HTCondor

Version Number Scheme

- Major.minor.release
  - If minor is even (a.b.c): Stable series
    - Very stable, mostly bug fixes
    - Current: 8.2
    - Examples: 8.2.5, 8.0.3
      - 8.2.5 just released
  - If minor is odd (a.b.c): Developer series
    - New features, may have some bugs
    - Current: 8.3
    - Examples: 8.3.2,
      - 8.3.2 just released

The Guarantee

- All minor releases in a stable series interoperate
  - E.g. can have pool with 8.2.0 8.2.1, etc.
  - But not WITHIN A MACHINE:
    - Only across machines
- The Reality
  - We work really hard to to better
    - 8.2 with 8.0 with 8.3, etc.
    - Part of HTC ideal: can never upgrade in lock-step

Condor_master will restart children

- Master periodically stats executables
  - But not their shared libraries

- Will restart them when they change

Upgrading the Central Manager

- Central Manager is Stateless
- If not running, jobs keep running
  - But no new matches made
- Easy to upgrade:
  - Shutdown & Restart
- Condor_status won't show full state for a while

Upgrading the workers

- No State, but jobs
- No way to restart startd w/o killing jobs
- Hard choices: Drain vs. Evict
  - Cost of draining may be high than eviction

Upgrading the schedds

- All the important state is here
- If schedd restarts quickly, jobs stay running
- How Quickly:
  - Job_lease_duration (20 minute default)
- Local / Scheduler universe jobs restart
  - Impacts DAGman

Basic plan for upgrading

- Plan to migrate/upgrade configuration
- Shutdown old daemons
- Install new binaries
- Startup again

  - But not all all machines at once.

Config changes

- Condor_config_val –writeconfig:upgrade x
  - Writes a new config file with just the diffs
- Condor manual release notes

- You did put all changes in local file, right?

condor_off

- Three kinds for submit and execute
- -fast:
  - Kill all jobs immediate, and exit
- -gracefull
  - Give all jobs 10 minutes to leave, then kill
- -peaceful
  - Wait forever for all jobs to exit

Recommended Strategy for a big pool

- Upgrade a few startds,
  - let them run for a few days
- Upgrade/create one small (test?) schedd
  - Let it run for a few days
- Upgrade the central manager
- Rolling upgrade the rest of the startds
- Upgrade the rest of the schedds

Thank you!