HTCondor Administration Basics

John (TJ) Knoeller
Center for High Throughput Computing

Overview

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

Two Big HTCondor Abstractions

- Jobs
  - Request for resources
  - Description of the job components
  - Restriction on where job can run
- Machines (a.k.a. Slots)
  - Set of resources
  - Restrictions on what jobs are permitted

One Big Abstraction Holder

- ClassAd
  - Attribute = Value pairs
    - Attributes are case-insensitive, unique
    - Values have type: int, float, string …
    - Expressions!
  - Some attributes have meaning to HTCondor
  - Admins can make up new attributes
    - So can users...

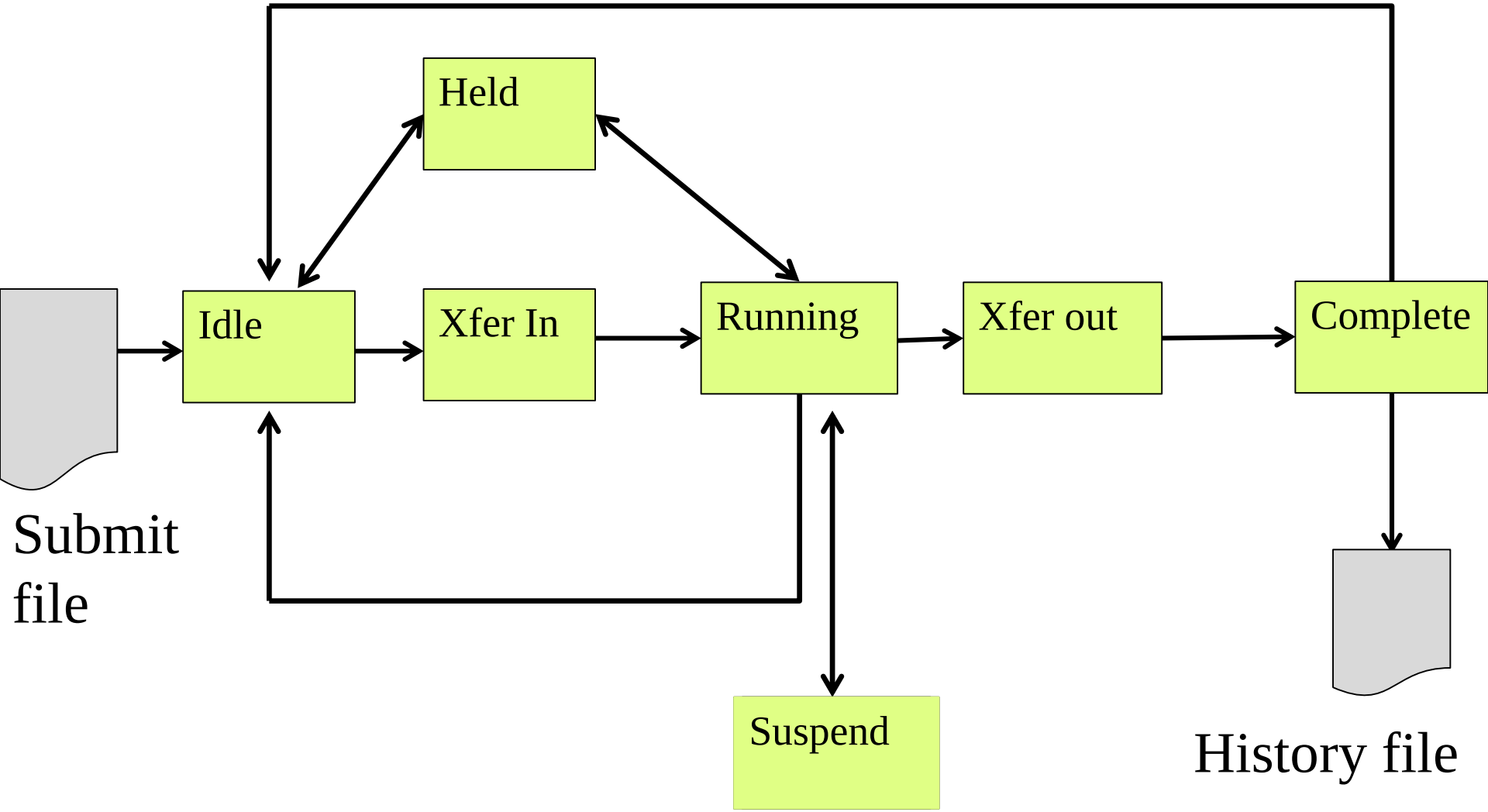
(Classads are used all over in HTCondor)

Classad Expressions

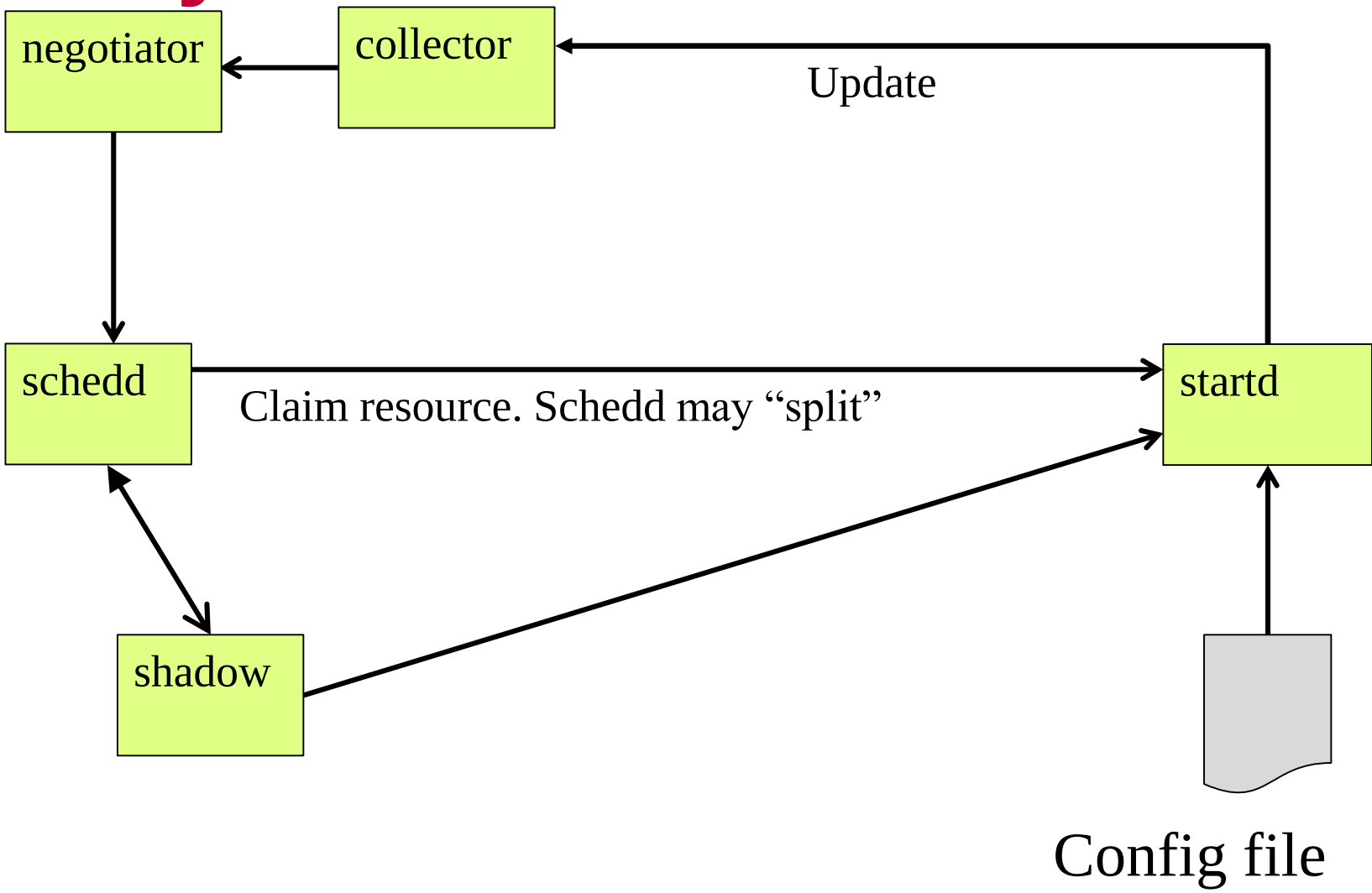
- Values can be expressions that refer to other classad attributes and can be
  - Math
  - Boolean logic
  - String manipulation
- Used extensively by HTCondor to set policy

```
Requirements = RequestCpus <= TARGET.Cpus &&
 TARGET.HasDocker && OpSysAndVer == "RedHat6"
```

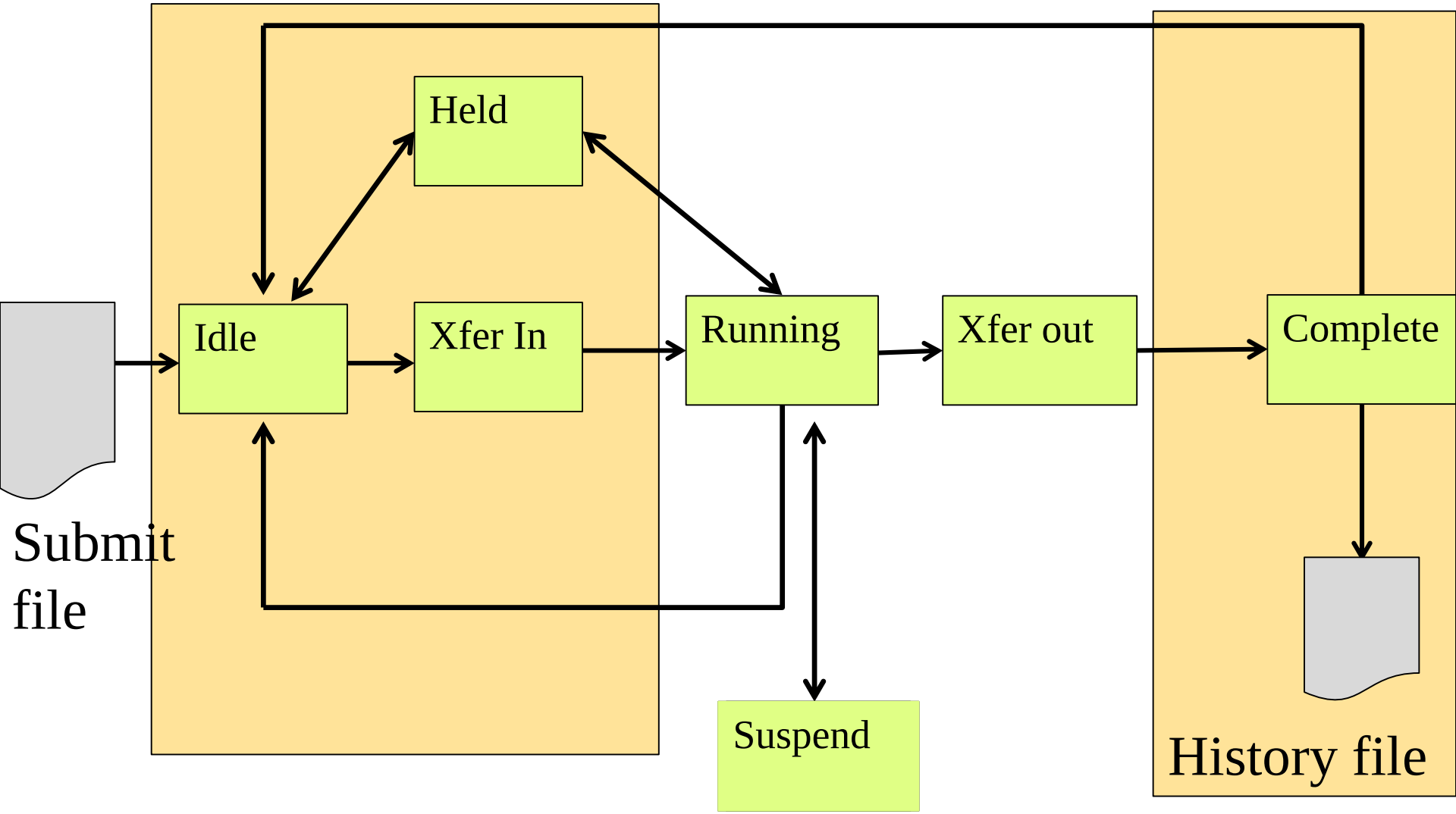
Life cycle of HTCondor Job



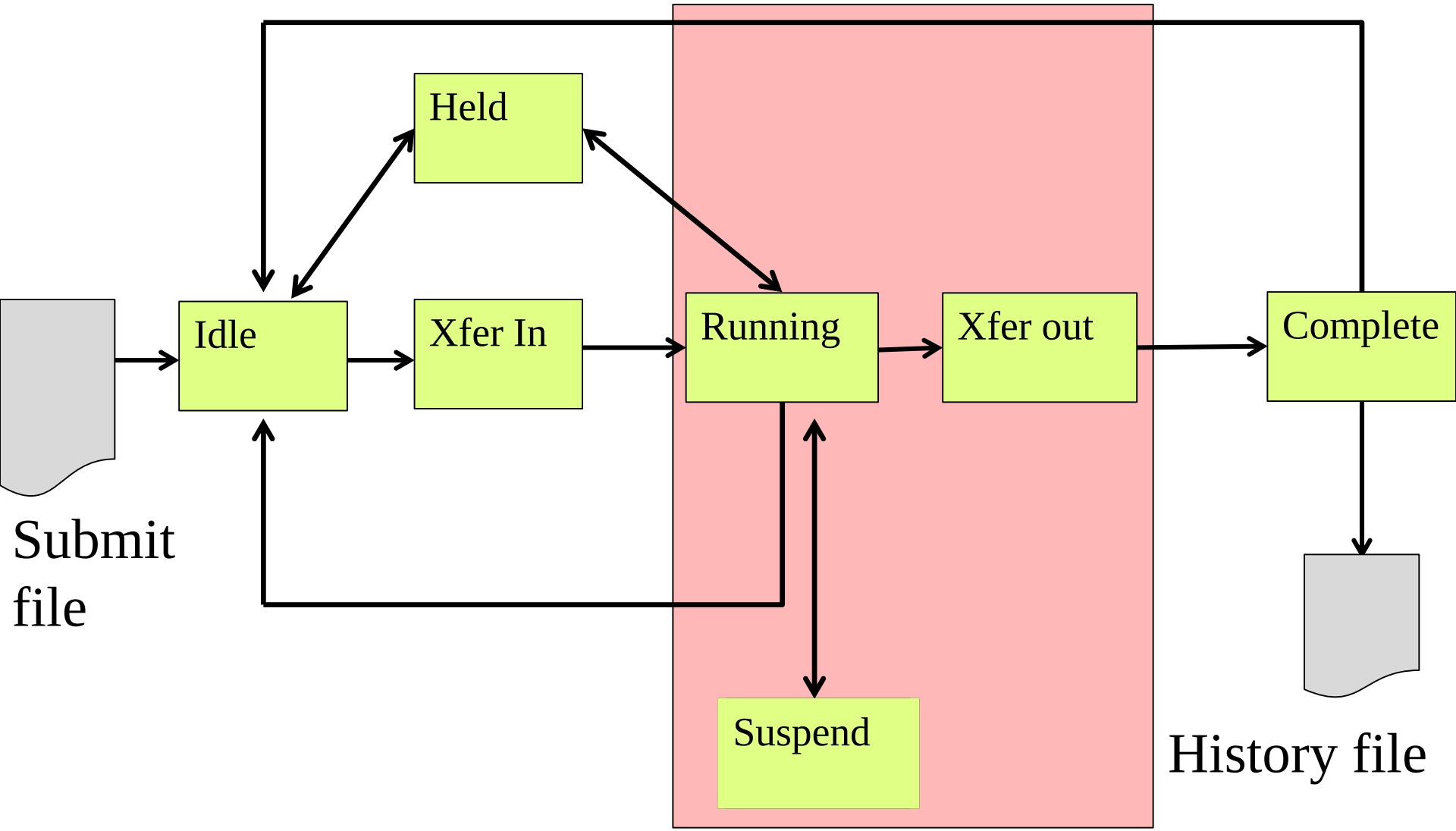
Life cycle of a Machine/Slot



“Submit Side”

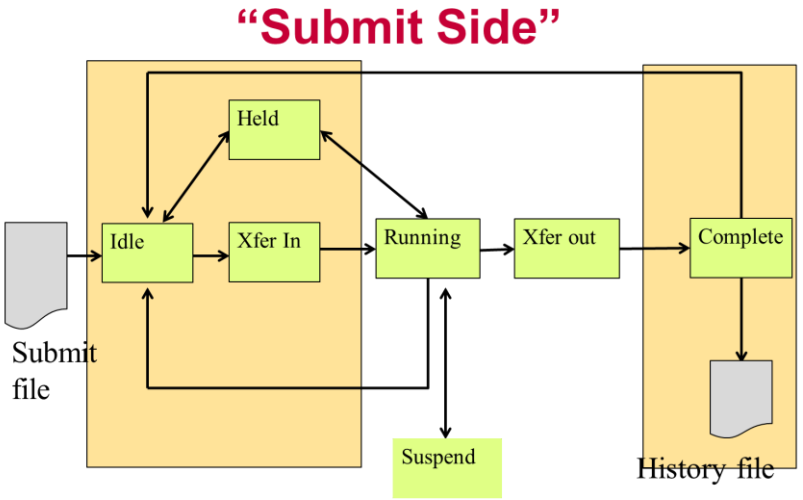

Held
Idle
Xfer In
Running
Xfer out
Complete
Submit file
Suspend
History file

“Execute Side”



The submit side

- Submit side managed by single condor_schedd process
- And one shadow per running job
  - condor_shadow process
- The Schedd is a database that holds Job classads



- Submit points can be performance bottleneck

- Usually a handful per pool

In the Beginning…

HTCondor Submit file (not a classad…)

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

Submit/Config “language”

- Submit files and config files
  - Statements (if, include, queue)
  - Key = Value pairs
    - Key are case-insensitive, Last definition wins
    - Values have no type. (all strings)
    - Values can refer to other values using $(key)
  - Some Keys have meaning to HTCondor
  - Admins/Users can make up new keys

(more on config syntax later)

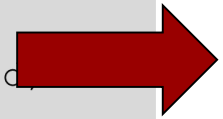
Submit file

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

Queue 1
```

From submit to schedd

```
universe = vanilla
executable = compute
arguments = -dataset $(ProcID).dat
request_memory = 70M
should_transfer_input = yes
transfer_input_files = $(ProcId
output = out.$(ProcID)
error = error.$(ProcId)
+IsVerySpecialJob = true
Queue 1
```



```
JobUniverse = 5
Cmd = "compute"
Args = "-dataset 0.dat"
RequestMemory = 70
Requirements = Opsys == "LI..
DiskUsage = 0
Output = "out.0"
IsVerySpecialJob = true
```

condor_submit *submit_file*

- Submit file in, Job classad(s) 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 = “johnkn”
JobStatus = 1
NumJobStarts = 5
Cmd = “compute”
Args = “0”
RequestMemory = 70
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 = "johnkn"
JobStatus = 1
NumJobStarts = 5
Cmd = "compute"
Args = "0"
RequestMemory = 70
Requirements = Opsys == "LI..
DiskUsage = 0
Output = "out.0"
IsVerySpecialJob = true
```

What if I don't like those Attributes?

- SUBMIT_EXPRS
  - Administrator can inject attributes into Jobs
- SUBMIT_REQUIREMENTS
  - Administrator can validate job attributes
- Write a wrapper to condor_submit
- condor_qedit
  - Modify the Job classad in the Schedd

Configuration of Submit side

- Not much policy to be configured in schedd
  - SUBMIT_REQUIREMENT_*
- Mainly scalability and security
  - MAX_JOBS_RUNNING
  - JOB_START_DELAY
  - MAX_CONCURRENT_DOWNLOADS
  - MAX_JOBS_SUBMITTED

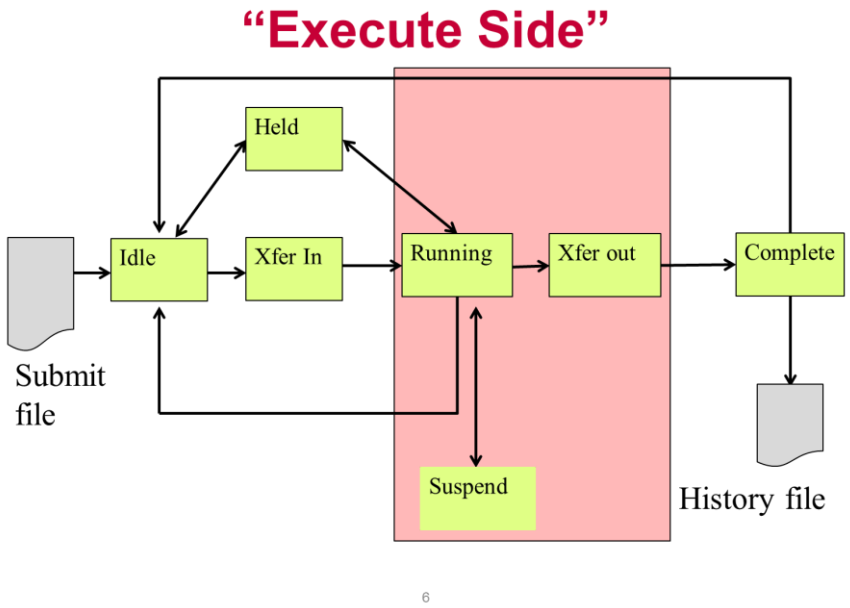
The Execute Side

Primarily managed by condor_startd process

With one condor_starter per running job

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(s)
- condor_status –direct *machine*
  - Goes to the startd

Condor_status –long *machine*

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

One Startd, Many slots

- HTCondor treats multicore as independent slots by default
- 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 policy
- A whole talk just on this

Configuration of startd

- Mostly policy, whole talk on that
- Several directory parameters
- EXECUTE – where the sandbox is
- START
  - When to run jobs, which jobs to run
- 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 Schedd(s)
  - 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

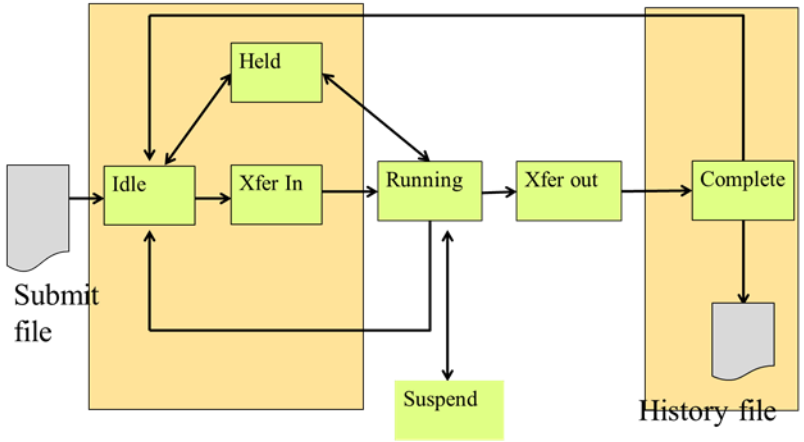
  condor_shadow: one per job

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

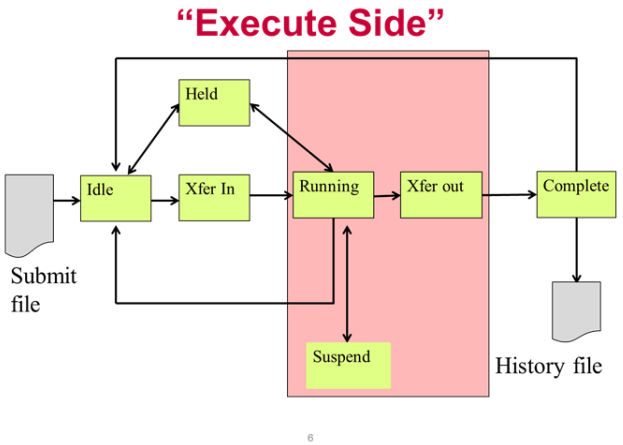
condor_shadow

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

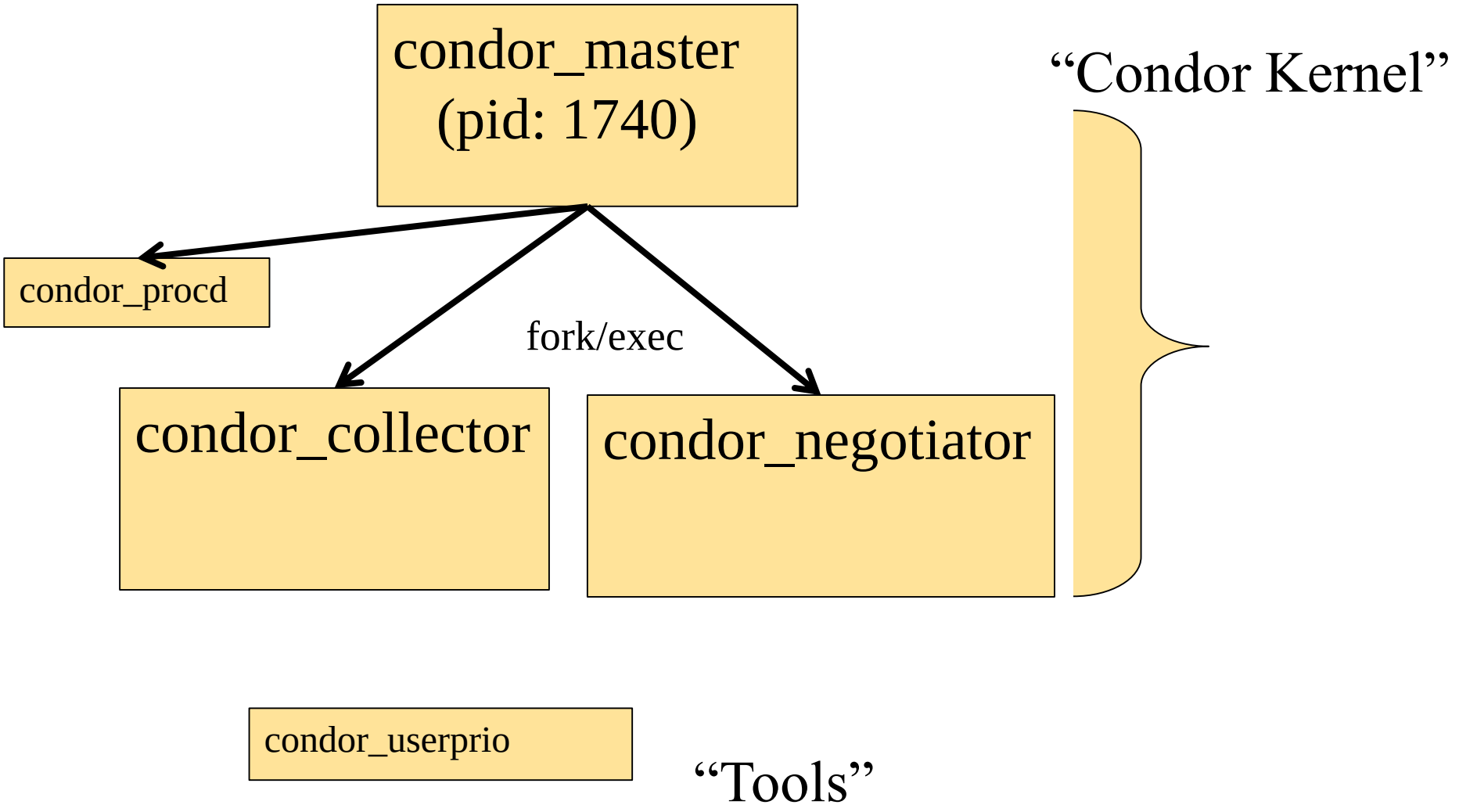
"Condor Userspace"

Job

Job

Job

Process View: Central Manager



Condor Admin Basics

Install HTCondor

- Either with tarball
  - tar xvf htcondor-8.2.3-redhat6
  - Run condor_install script
- Or native packages (rpm, deb, msi)
  - yum install htcondor

Basic HTCondor Commands

```
condor_on                condor_off
condor_config_val        condor_reconfig
condor_status            condor_advertise
condor_submit            condor_rm
condor_q                 condor_qedit
condor_hold              condor_release
condor_vacate            condor_vacate_job
condor_prio              condor_userprio
condor_drain             condor_who
```

Let's Make a Pool

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

Configuration File

(Almost) all configure is in files, “base” is **CONDOR_CONFIG** environment variable

- **/etc/condor/condor_config**
- This file points to others
- All daemons share same configuration
- Might want to share between all machines (NFS, HTTP, rsync, puppet, etc)

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`
- Macro reference is **text** substitution
- Can create additional macros for organizational purposes

Configuration File Macros

- Self references allowed

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

  Equivalent to: `A=abc,def`

- Circular references are not!!

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

  Daemon (or tool) will abort

Configuration File Macros

- Substitions happen **after** config is parsed
  - Self references are exception to this
  - So are config statements (more later)
- So last definition wins

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

A and B are the same, both are `1,2`

Environment overrides

- Environment overrides are read LAST
  - Trumps all others (so be careful)

export _condor_KNOB_NAME=value

Config file recommendations

- For “system” condor, use default
  - Global config file. minimal, read-only
    - /etc/condor/condor_config
  - Central config management via HTTP or GIT
  - Or use small snippets in config.d
    - All files begin with 2 digit numbers

    /etc/condor/config.d/05some_example
- Personal condors elsewhere
  - CONDOR_CONFIG environment variable

condor_reconfig

- Daemons long-lived
  - Only re-read config files on condor_reconfig command
  - Some knobs don't obey reconfig, require restart
    - DAEMON_LIST, NETWORK_INTERFACE
    - Most STARTD Slot configuration
- condor_restart
- condor_off / condor_on

Got all that?

Let's make a pool!

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

Default file locations for tar

```
BIN = tardir/bin
SBIN = tardir/sbin
CONDOR_CONFIG = \
tardir/etc/condor_config.generic
LOG = $(LOCAL_DIR)/log
# script to setup htcondor
tardir/condor_install
```

Default file locations for rpm

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

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

`condor_config_val LOG`

Where daemons write debugging info

`condor_config_val SPOOL`

Where the schedd stores jobs and data

`condor_config_val EXECUTE`

Where the startd runs jobs

Minimum knobs for personal Condor

```
RELEASE_DIR = <where you untar'd>

LOCAL_DIR = <where to put LOG, SPOOL>

Use ROLE : Personal
     CONDOR_HOST=127.0.0.1
     COLLECTOR_HOST=$(CONDOR_HOST):0
     DAEMON_LIST=MASTER COLLECTOR NEGOTIATOR STARTD SCHEDD
     RunBenchmarks=0

Use SECURITY : HOST_BASED

ALLOW_WRITE=$(ALLOW_WRITE) $(IP_ADDRESS)
```

Starting Condor

- condor_master –f
  - For any condor
  - Run as root for full HTCondor capabilities
- service start condor
  - For rpm “root” condor

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
$
```

```
$ 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
- `START = <classad-expression>`
  - Which jobs are allowed to run (and when)
- `STARTD_ATTRS = <knob-list>`
  - Advertise <knob-list> in Startd ClassAd

Brief Diversion into daemon logs

- Each daemon logs mysterious info to file
- $(LOG)/DaemonNameLog
- Default:
  - /var/log/condor/MasterLog
  - /var/log/condor/SchedLog
  - /var/log/condor/StarterLog.slotX
- Experts-only view of condor
  - When daemons don't start, won't talk

Let's make a "real" pool

- Distributed machines makes it hard
  - Different policies on each machines
  - Different owners
  - Scale
- Each machine has it's own config files

  (Unless you arrange otherwise)

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 (the default)
    - 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`
- Mismatched domains can prevent jobs from matching – jobs stay idle.

Typical real HTCondor pool

- One Central Manager
- At least one Execute Machine
  - Usually many
- At least one Submit Machine
  - One, a few, or many

Central Manager

```
Use ROLE : CentralManager
CONDOR_HOST = cm.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
# to use a non-default port
# default is 9618
#COLLECTOR_HOST=$(CONDOR_HOST):1234
# ^- set this for ALL machines…
```

Submit Machine

```
Use ROLE : submit
CONDOR_HOST = cm.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
UID_DOMAIN = cs.wisc.edu
FILESYSTEM_DOMAIN = cs.wisc.edu
```

Execute Machine

```
Use ROLE : Execute
CONDOR_HOST = cm.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
UID_DOMAIN = cs.wisc.edu
FILESYSTEM_DOMAIN = cs.wisc.edu
# default is
#FILESYSTEM_DOMAIN=$(FULL_HOSTNAME)
```

Now Start them all up

- Does order matter?
  - Somewhat: starting CM first is best
- How to check:
- Every Daemon has classad in collector
  - condor_status -master
  - condor_status -schedd
  - condor_status -any

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?

- Search NegotiatorLog, ScheddLog and StartLog for refused connections
  - Fix ALLOW_NEGOTIATOR
- Check userlog – may be preempted often
  - Check StarterLog.slotN for reasons
- run condor_q –better-analyze <job_id>
  - Unmatched FILESYSTEM_DOMAIN?

How big can I go?

- 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

Thank you!