



# HTCondor Administration Basics

Greg Thain  
Center for High Throughput Computing

# Overview

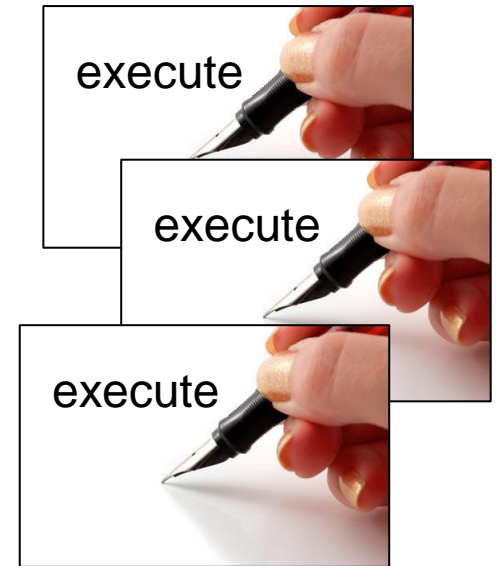
- › HTCondor Architecture Overview
- › Classads, briefly
- › Configuration and other nightmares
- › Setting up a personal condor
- › Setting up distributed condor
- › Minor topics

# Two Big HTCondor Abstractions

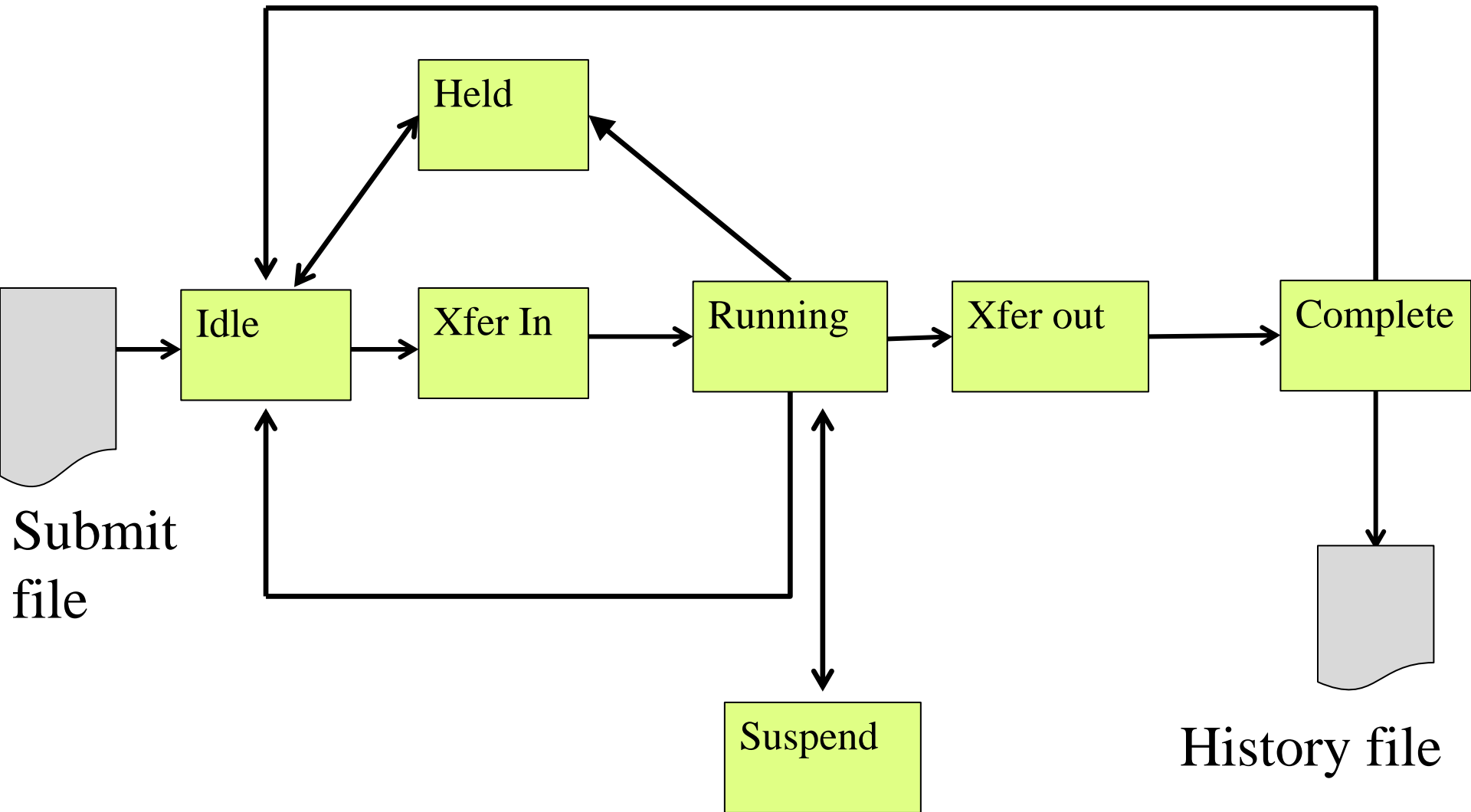
› Jobs



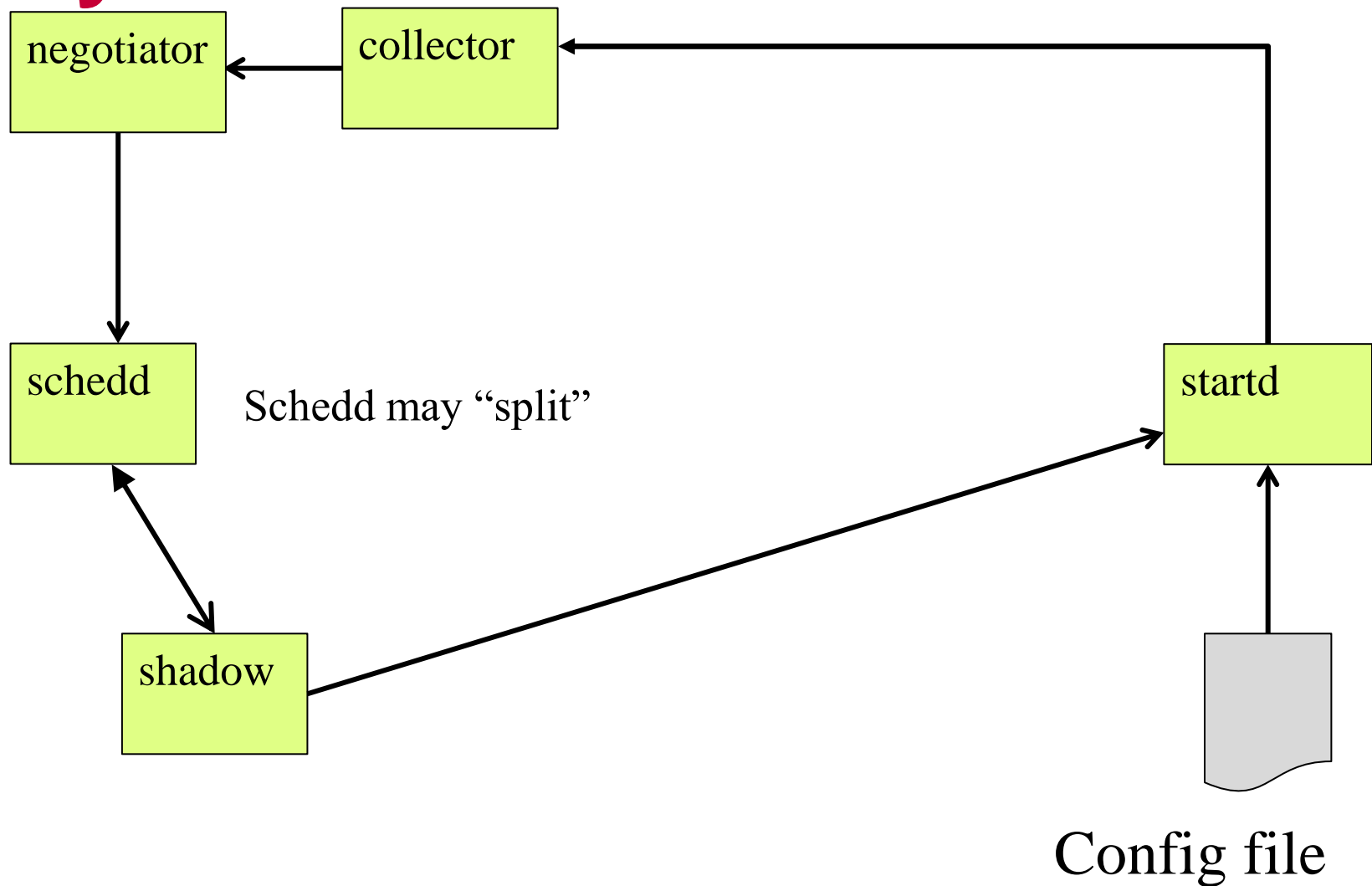
› Machines



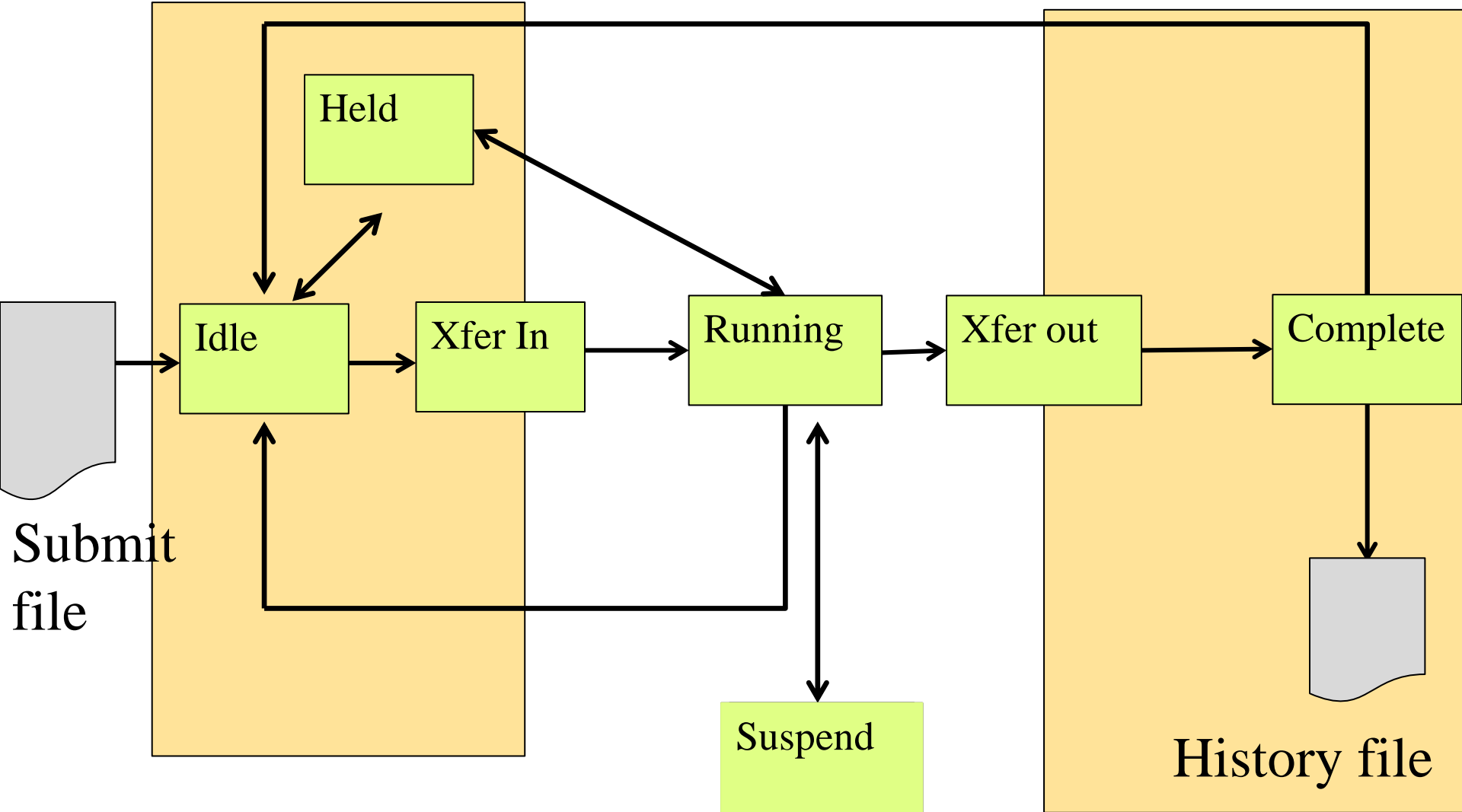
# Life cycle of HTCondor Job



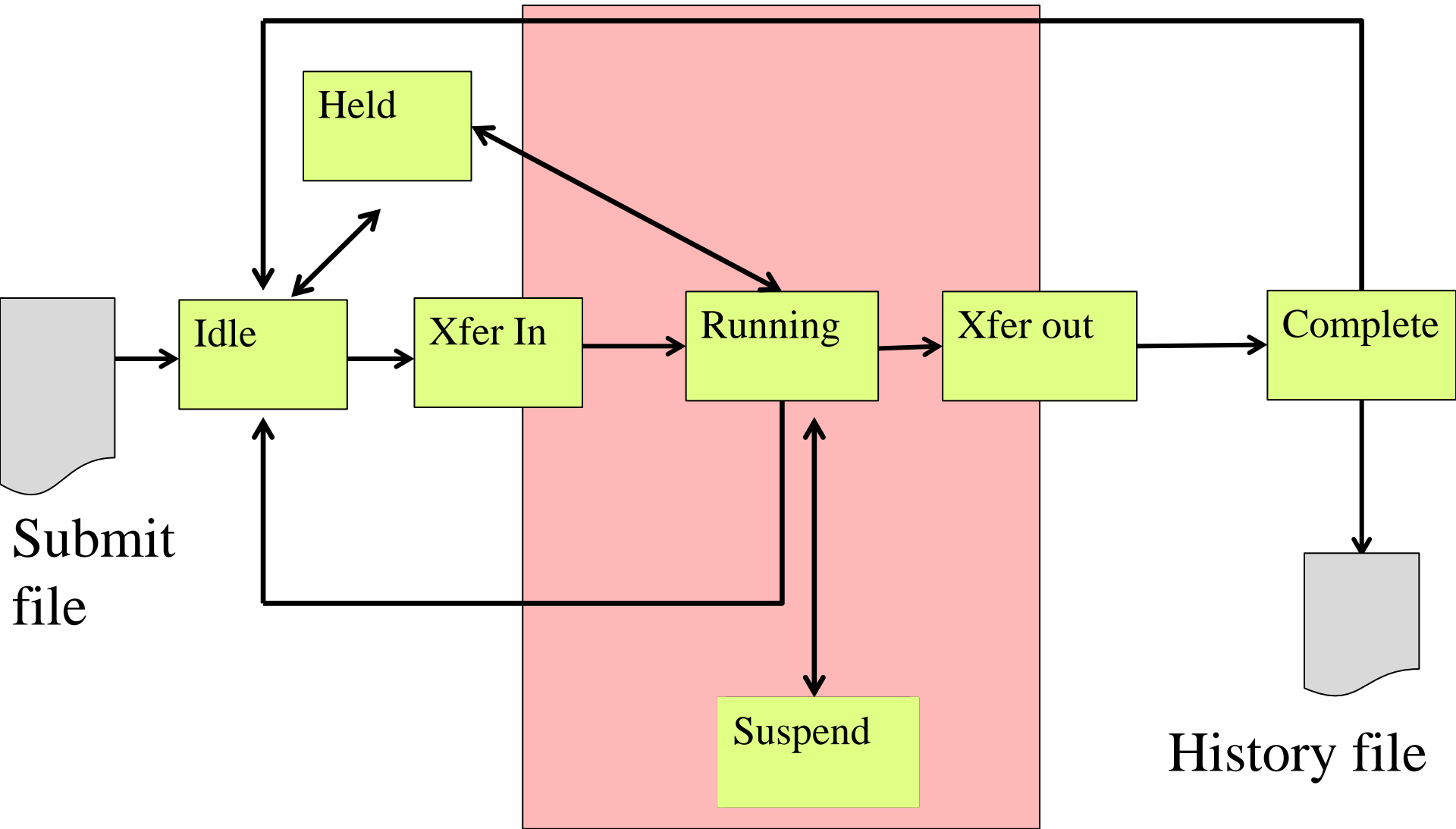
# Life cycle of HTCondor Machine



# “Submit Side”

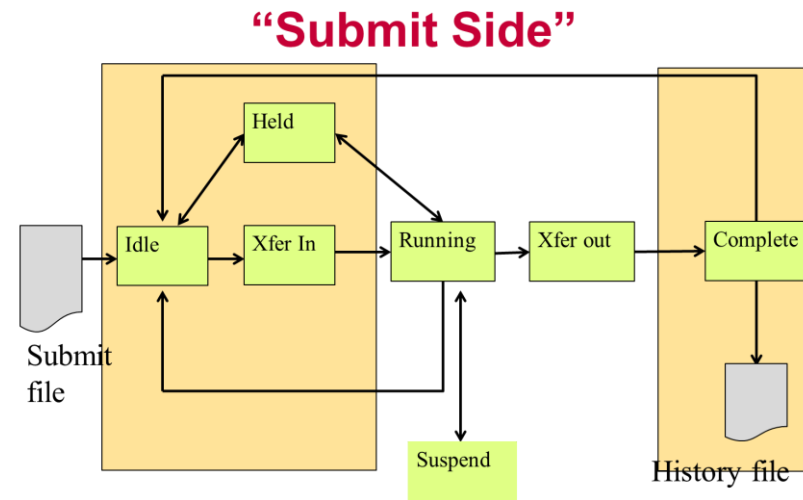


# “Execute Side”



# 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



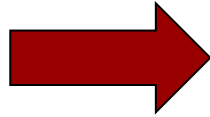
# In the Beginning...

```
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\_ATTRS
- › condor\_qedit
- › +Notation
- › Schedd transforms

# ClassAds: The *lingua franca* of HTCondor



# Classads for ~~people~~ admins

# What are ClassAds?

ClassAds is a language for objects (jobs and machines) to

- Express attributes about themselves
- Express what they require/desire in a “match” (similar to personal classified ads)

**Structure** : Set of attribute name/value pairs, where the value can be a literal or an expression. Semi-structured, no fixed schema.



# Example

## Pet Ad

Type = "Dog"

**Requirements** =

DogLover =?= True

Color = "Brown"

Price = 75

Sex = "Male"

AgeWeeks = 8

Breed = "Saint Bernard"

Size = "Very Large"

Weight = 27

## Buyer Ad

AcctBalance = 100

DogLover = True

**Requirements** =

(Type == "Dog") &&

(TARGET.Price <=

MY.AcctBalance) &&

( Size == "Large" ||

Size == "Very Large" )

**Rank** =

100\* (Breed == "Saint  
Bernard") - Price

. . .

# ClassAd Values

## › Literals

- Strings ( “RedHat6” ), integers, floats, boolean (true/false), ...

## › Expressions

- Similar look to C/C++ or Java : operators, references, functions
- **References**: to other attributes in the same ad, or attributes in an ad that is a candidate for a match
- **Operators**: +, -, \*, /, <, <=, >, >=, ==, !=, &&, and || all work as expected
- **Built-in Functions**: if/then/else, string manipulation, regular expression pattern matching, list operations, dates, randomization, math (ceil, floor, quantize,...), time functions, eval, ...

# Four-valued logic

- › ClassAd Boolean expressions can return four values:
  - True
  - False
  - Undefined (a reference can't be found)
  - Error (Can't be evaluated)
- › Undefined enables explicit policy statements *in the absence of data* (common across administrative domains)
- › Special meta-equals ( `=?=` ) and meta-not-equals ( `!==` ) will never return Undefined

```
[  
  HasBeer = True  
  GoodPub1 = HasBeer == True  
  GoodPub2 = HasBeer =?= True  
]
```

```
[  
  GoodPub1 = HasBeer == True  
  GoodPub2 = HasBeer =?= True  
]
```

# ClassAd Types

- › HTCondor has many types of ClassAds
  - A "Job Ad" represents a job to Condor
  - A "Machine Ad" represents a computing resource
  - Others types of ads represent other instances of other services (daemons), users, accounting records.

# The Magic of Matchmaking

- › Two ClassAds can be matched via special attributes: Requirements and Rank
- › Two ads match if both their Requirements expressions evaluate to True
- › Rank evaluates to a float where higher is preferred; specifies which match is desired if several ads meet the Requirements.
- › Scoping of attribute references when matching
  - MY.name – Value for attribute “name” in local ClassAd
  - TARGET.name – Value for attribute “name” in match candidate ClassAd
  - Name – Looks for “name” in the local ClassAd, then the candidate ClassAd

# Example

## Pet Ad

Type = "Dog"

**Requirements** =

DogLover =?= True

Color = "Brown"

Price = 75

Sex = "Male"

AgeWeeks = 8

Breed = "Saint Bernard"

Size = "Very Large"

Weight = 27

## Buyer Ad

AcctBalance = 100

DogLover = True

**Requirements** =

(Type == "Dog") &&

(TARGET.Price <=

MY.AcctBalance) &&

( Size == "Large" ||

Size == "Very Large" )

**Rank** =

100\* (Breed == "Saint  
Bernard") - Price

. . .

# Back to configuration...

# Configuration File

- › (Almost) all configuration is in files, “root”  
`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
```

# 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 -dump`
- › Environment overrides:
- › `export _condor_KNOB_NAME=value`
  - Over rules all others (so be careful)

# condor\_reconfig

- › Daemons long-lived
  - Only re-read config files on condor\_reconfig command
  - Some knobs don't obey re-config, require restart
    - DAEMON\_LIST, NETWORK\_INTERFACE
- › condor\_restart



**Got all that?**

# 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

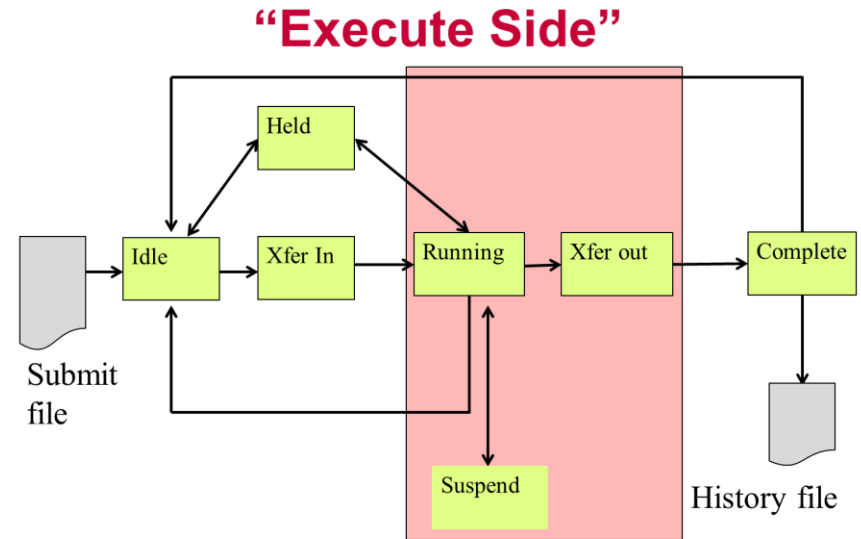
# The Execute Side

Primarily managed by  
condor\_startd process

With one condor\_starter  
per running jobs

Sandboxes the jobs

Usually many per pool  
(support 10s of thousands)



6

# Startd also has a classad

- › Condor creates it
  - 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
- › Slots: static vs. partitionable
- › Startd can be configured to:
  - Only run jobs based on machine state
  - Only run jobs based on Resources (GPUs)
  - Preempt or Evict jobs based on policy
  - ...

# 3 types of slots

- › Static (e.g. the usual kind)
- › Partitionable (e.g. leftovers)
- › Dynamic (usable ones)
  - Dynamically created
  - But once created, static

# How to configure

```
NUM_SLOTS = 1
```

```
NUM_SLOTS_TYPE_1 = 1
```

```
SLOT_TYPE_1 = cpus=100%
```

```
SLOT_TYPE_1_PARTITIONABLE = true
```



# Configuration of startd

- › Mostly policy,
- › Several directory parameters
- › EXECUTE – where the sandbox is
- › COLLECTOR\_HOST – where the cm 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
    - condor\_status queries this
- › Please don’t call this “Master node” or head
- › Not the bottleneck you may think: stateless

# Responsibilities of CM

- › Pool-wide scheduling policy resides here
- › Scheduling of one user vs another
- › Definition of groups of users
- › Definition of preemption
- › Whole talk on this – this pm.

# Defrag daemon

- › Optional, but usually on the central manager
  - One daemon defragments whole pool
- › Scan pool, try to fully defrag some startds
- › Only looks at partitionable machines
- › Admin picks some % of pool that can be “whole”

# The condor\_master

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

# Quick Review of Daemons

condor\_master: runs on all machine, always

condor\_schedd: runs on submit machine

condor\_shadow: one per job

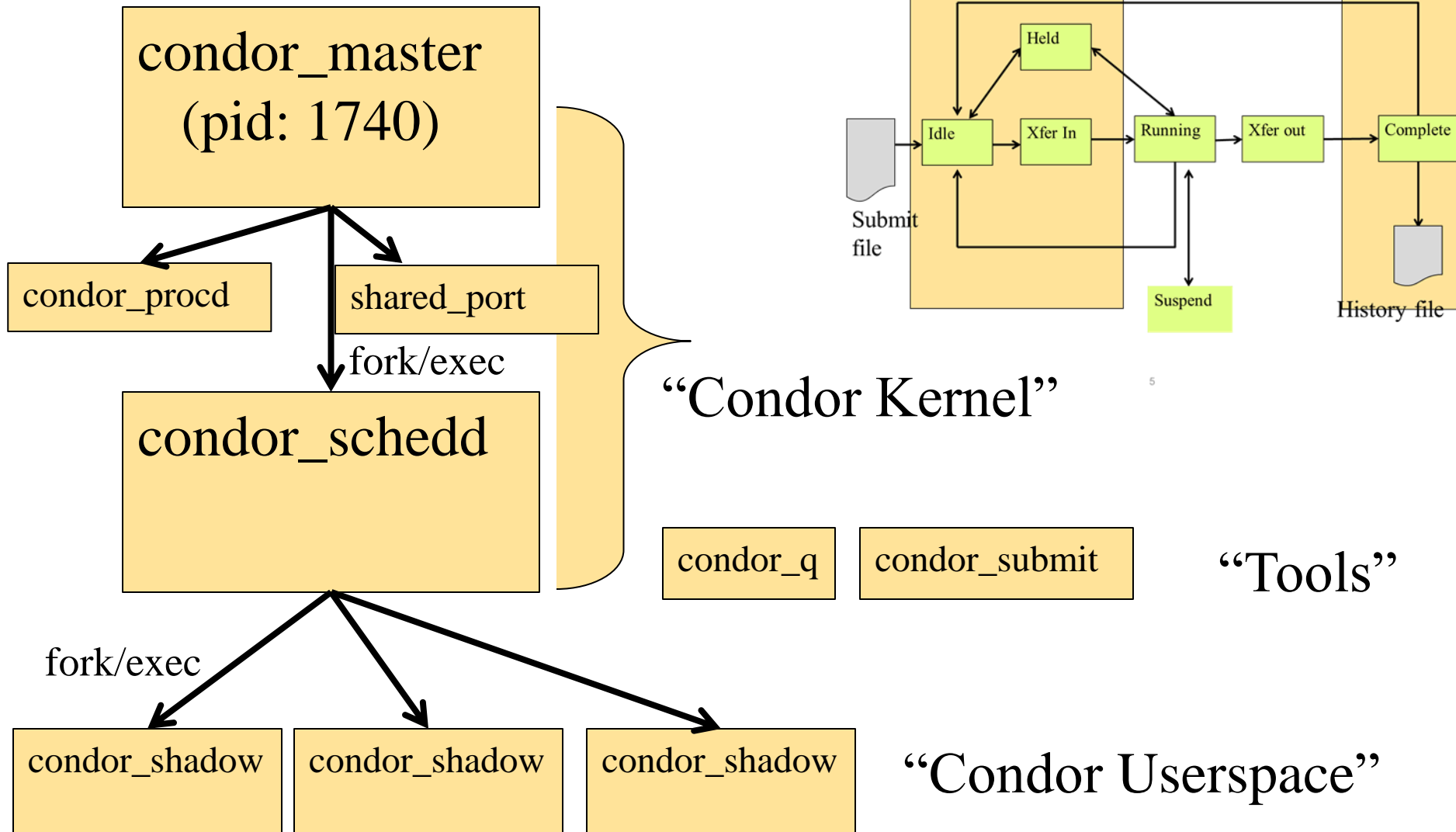
condor\_startd: runs on execute machine

condor\_starter: one per job

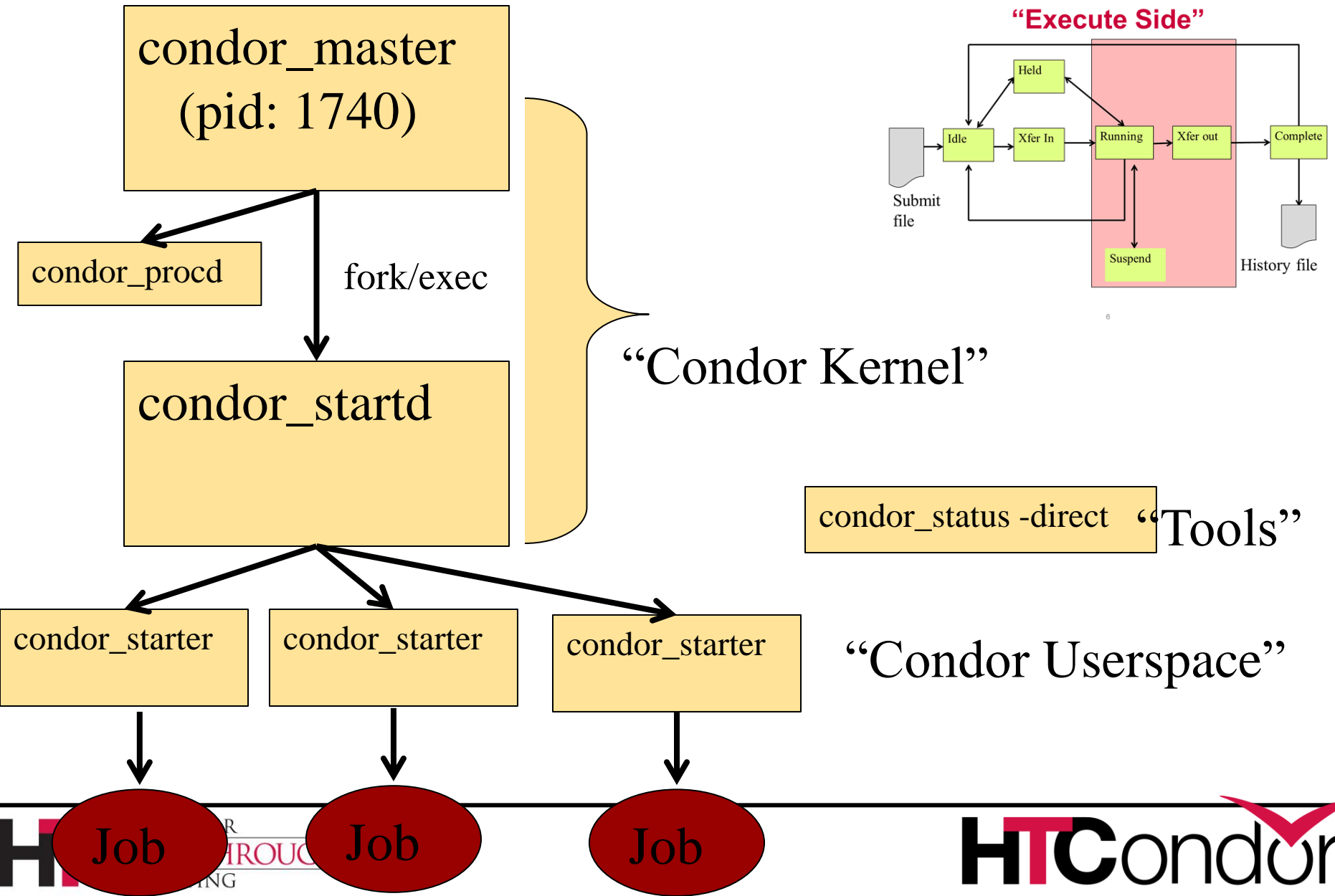
condor\_negotiator/condor\_collector

one per pool

# Process View “Submit Side”

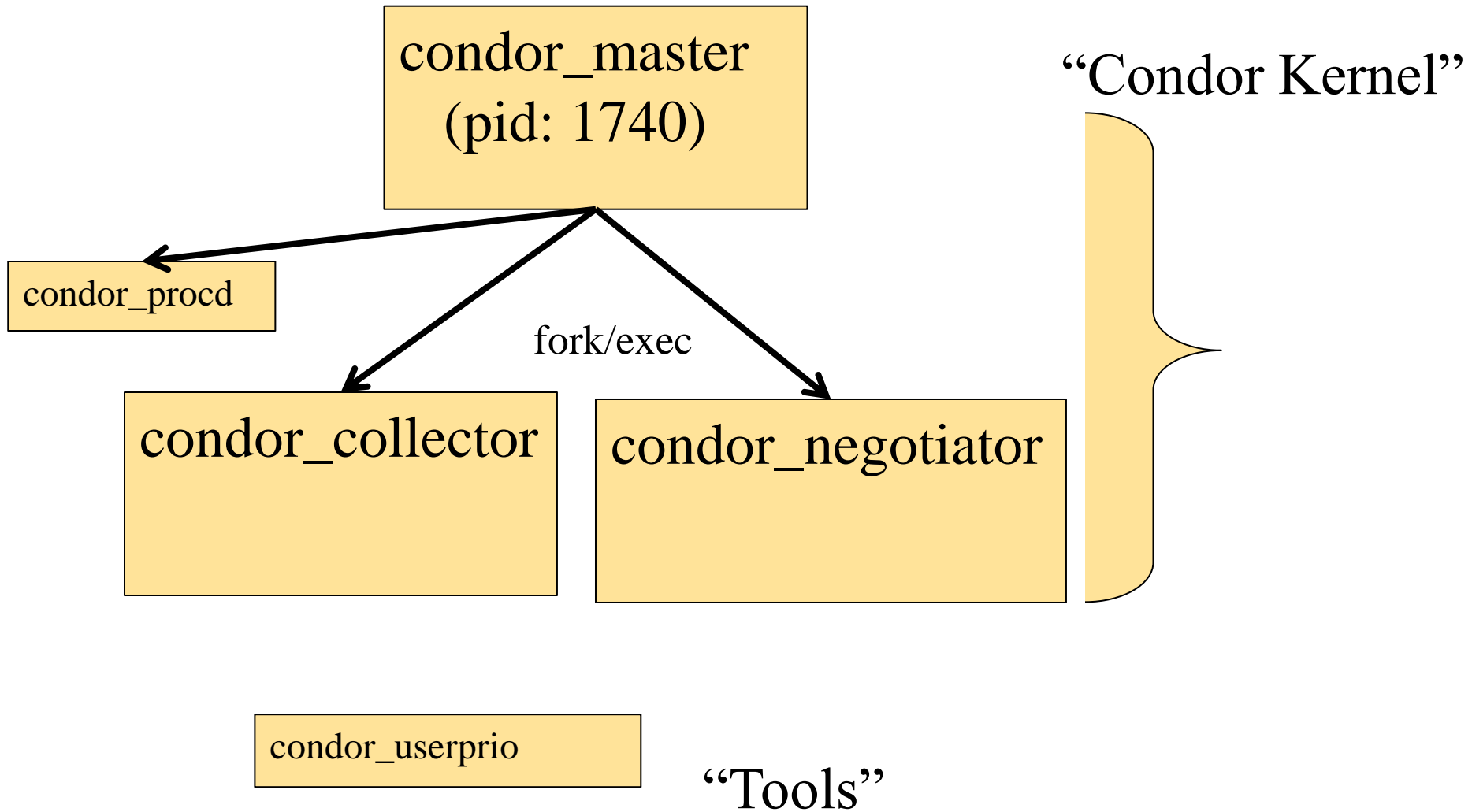


# Process View: Execute





# Process View: Central Manager



# Condor Installation Basics

# Let's Install HTCondor

## › Either with tarball

- `tar xvf htcondor-8.6.11-redhat6`

## › Or native packages

```
wget
```

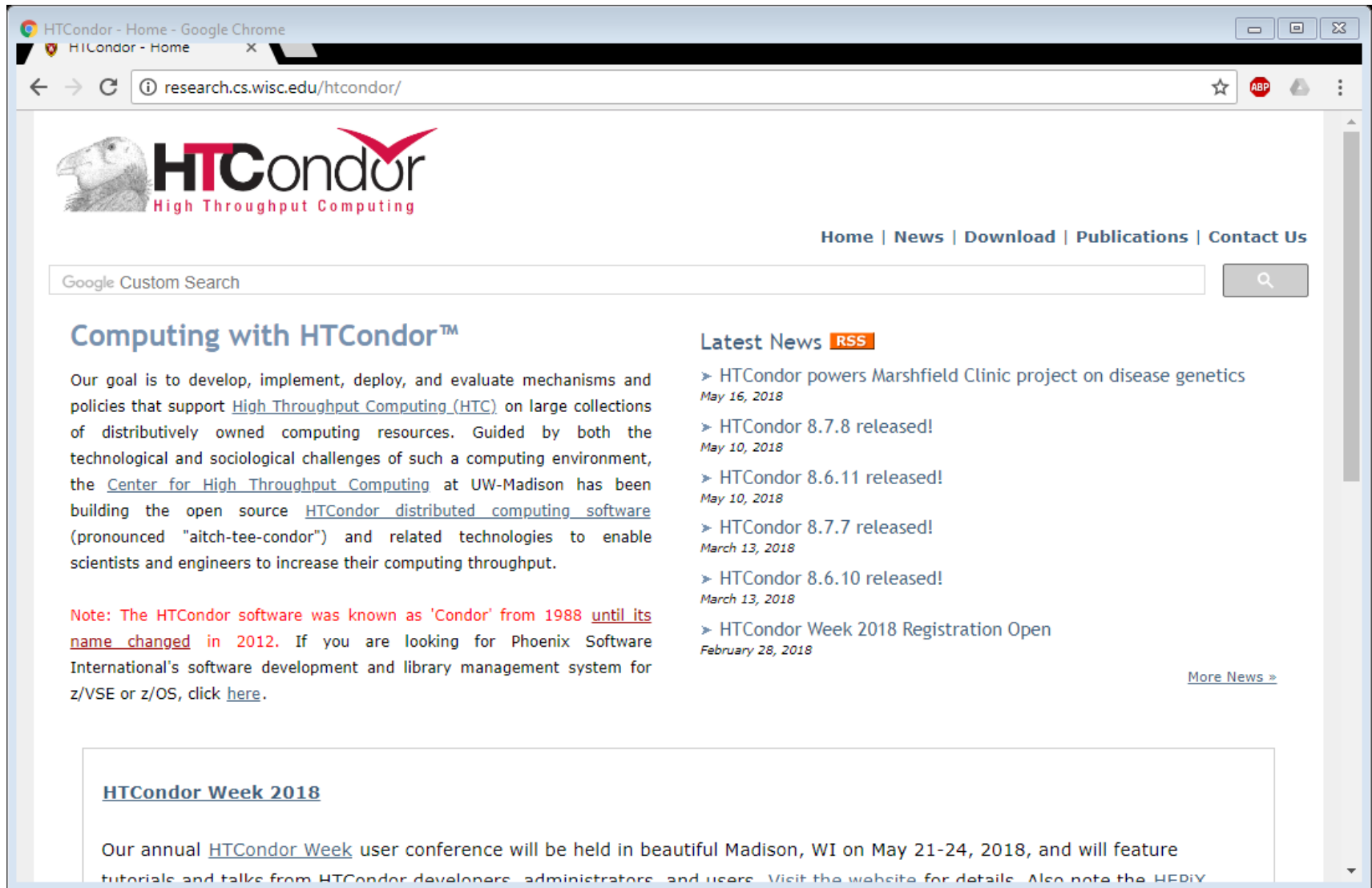
```
http://research.cs.wisc.edu/htcondor/yum/repo.d/htcondor-stable-rhel6.repo
```

```
get http://research.cs.wisc.edu/htcondor/yum/RPM-GPG-KEY-HTCondor
```

```
rpm -import RPM_GPG-KEY-HTCondor
```


```
Yum install htcondor
```

# http://htcondorproject.org



The screenshot shows the HTCondor website homepage. The browser window title is "HTCondor - Home - Google Chrome" and the address bar shows "research.cs.wisc.edu/htcondor/". The page features the HTCondor logo (a condor head) and the text "HTCondor High Throughput Computing". Navigation links include "Home | News | Download | Publications | Contact Us". A search bar is present with the text "Google Custom Search". The main content area is titled "Computing with HTCondor™" and contains a paragraph about the project's goals and a note about the software's history. A "Latest News" section lists several releases and events with dates. A "More News >" link is at the bottom of the news section. A "HTCondor Week 2018" section is partially visible at the bottom.

HTCondor - Home - Google Chrome  
research.cs.wisc.edu/htcondor/

 HTCondor  
High Throughput Computing

Home | News | Download | Publications | Contact Us

Google Custom Search

## Computing with HTCondor™

Our goal is to develop, implement, deploy, and evaluate mechanisms and policies that support [High Throughput Computing \(HTC\)](#) on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the [Center for High Throughput Computing](#) at UW-Madison has been building the open source [HTCondor distributed computing software](#) (pronounced "aitch-tee-condor") and related technologies to enable scientists and engineers to increase their computing throughput.

**Note:** The HTCondor software was known as 'Condor' from 1988 [until its name changed](#) in 2012. If you are looking for Phoenix Software International's software development and library management system for z/VSE or z/OS, click [here](#).

### Latest News [RSS](#)

- HTCondor powers Marshfield Clinic project on disease genetics  
*May 16, 2018*
- HTCondor 8.7.8 released!  
*May 10, 2018*
- HTCondor 8.6.11 released!  
*May 10, 2018*
- HTCondor 8.7.7 released!  
*March 13, 2018*
- HTCondor 8.6.10 released!  
*March 13, 2018*
- HTCondor Week 2018 Registration Open  
*February 28, 2018*

[More News >](#)

### HTCondor Week 2018

Our annual [HTCondor Week](#) user conference will be held in beautiful Madison, WI on May 21-24, 2018, and will feature tutorials and talks from HTCondor developers, administrators, and users. Visit the website for details. Also note the HEDiX

# Version Number Scheme

## > Major.minor.release

- If minor is even (a.b.c): Stable series
  - Very stable, mostly bug fixes
  - Current: 8.6
  - Examples: 8.2.5, 8.0.3
    - 8.6.0 coming soon to a repo near you
- If minor is odd (a.b.c): Developer series
  - New features, may have some bugs
  - Current: 8.7
  - Examples: 8.3.2,
    - 8.5.5 almost released

# The Guarantee

- › All minor releases in a stable series interoperate
  - E.g. can have pool with 8.4.0, 8.4.1, etc.
  - But not WITHIN A MACHINE:
    - Only across machines
- › The Reality
  - We work really hard to do better
    - 8.4 with 8.2 with 8.5, etc.
    - Part of HTC ideal: can never upgrade in lock-step

# 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`



# 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

## Role

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`

```
# All daemons local
```

```
Use ROLE : Personal
```

```
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 ...
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_ALLOW_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

```
Use ROLE : CentralManager  
CONDOR_HOST = cm.cs.wisc.edu  
ALLOW_WRITE = *.cs.wisc.edu
```

# 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: start CM first
- › How to check:
- › Every Daemon has classad in collector
  - condor\_status -schedd
  - condor\_status -negotiator
  - condor\_status -any

# condor\_status -any

MyType	TargetType	Name
Collector	None	Test <u><a href="mailto:Pool@cm.cs.wisc.edu">Pool@cm.cs.wisc.edu</a></u>
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!

# 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 File System 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

# Tools for admins

# 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

# condor\_restart

- › Restarts all daemons on a given machine
- › Can be run remotely – if admin priv allows

# condor\_status

- › -collector
- › -submitter
- › -negotiator
- › -schedd
- › -master



# condor\_userprio

- › Condor\_userprio –allusers
  - Whole talk on this,

# condor\_fetchlog

- › Remotely pulls a log file from remote machine
- › `condor_fetchlog execute_machine STARTD`

# Thank you -- For more info

- › <http://htcondorproject.org>
- › Detail talks today...
- › htcondor-users email list
- › Talk to us!