HTCondor Administration Basics

Greg Thain
Center for High Throughput Computing

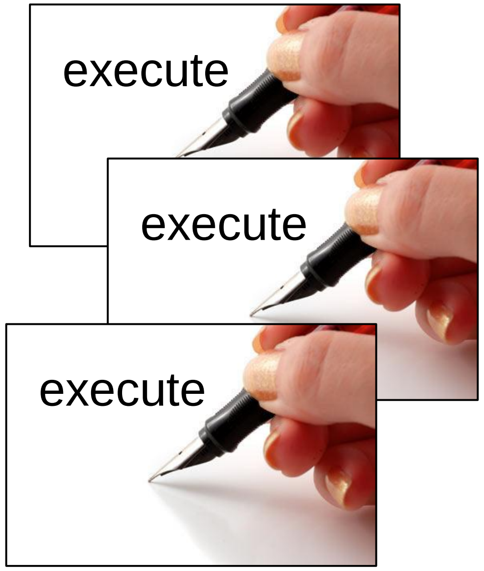
Overview

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

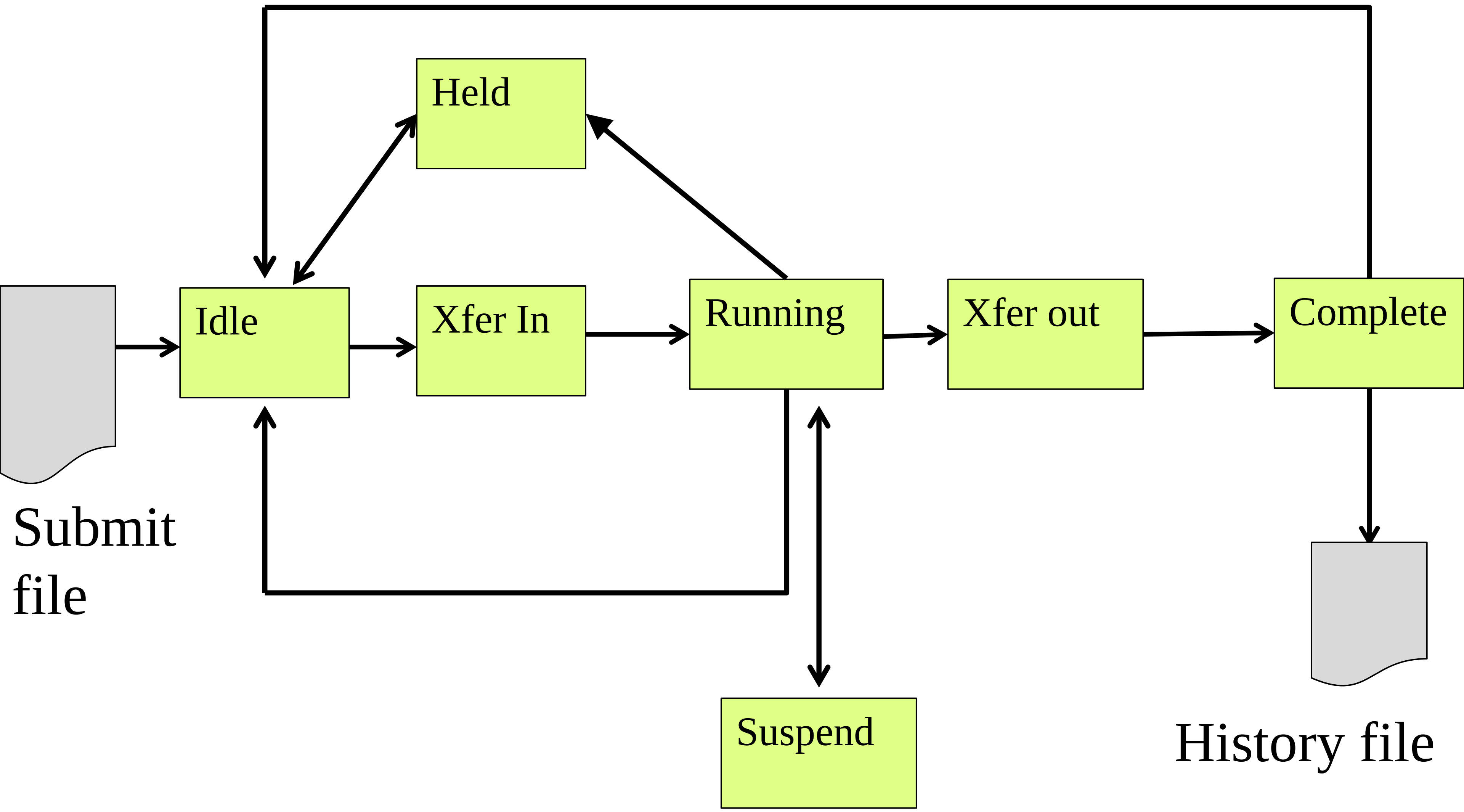
Two Big HTCondor Abstractions

- Jobs

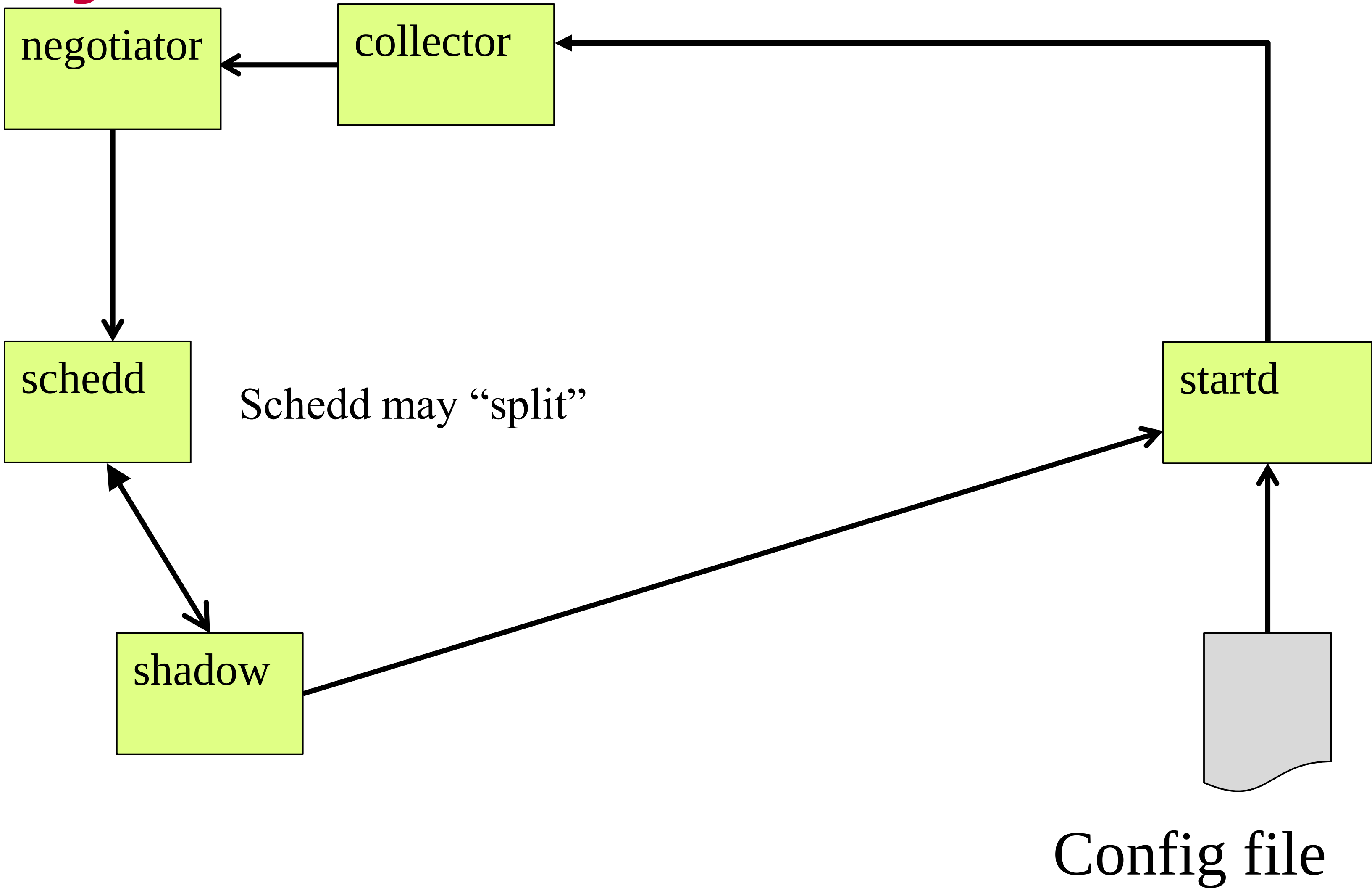
- Machines



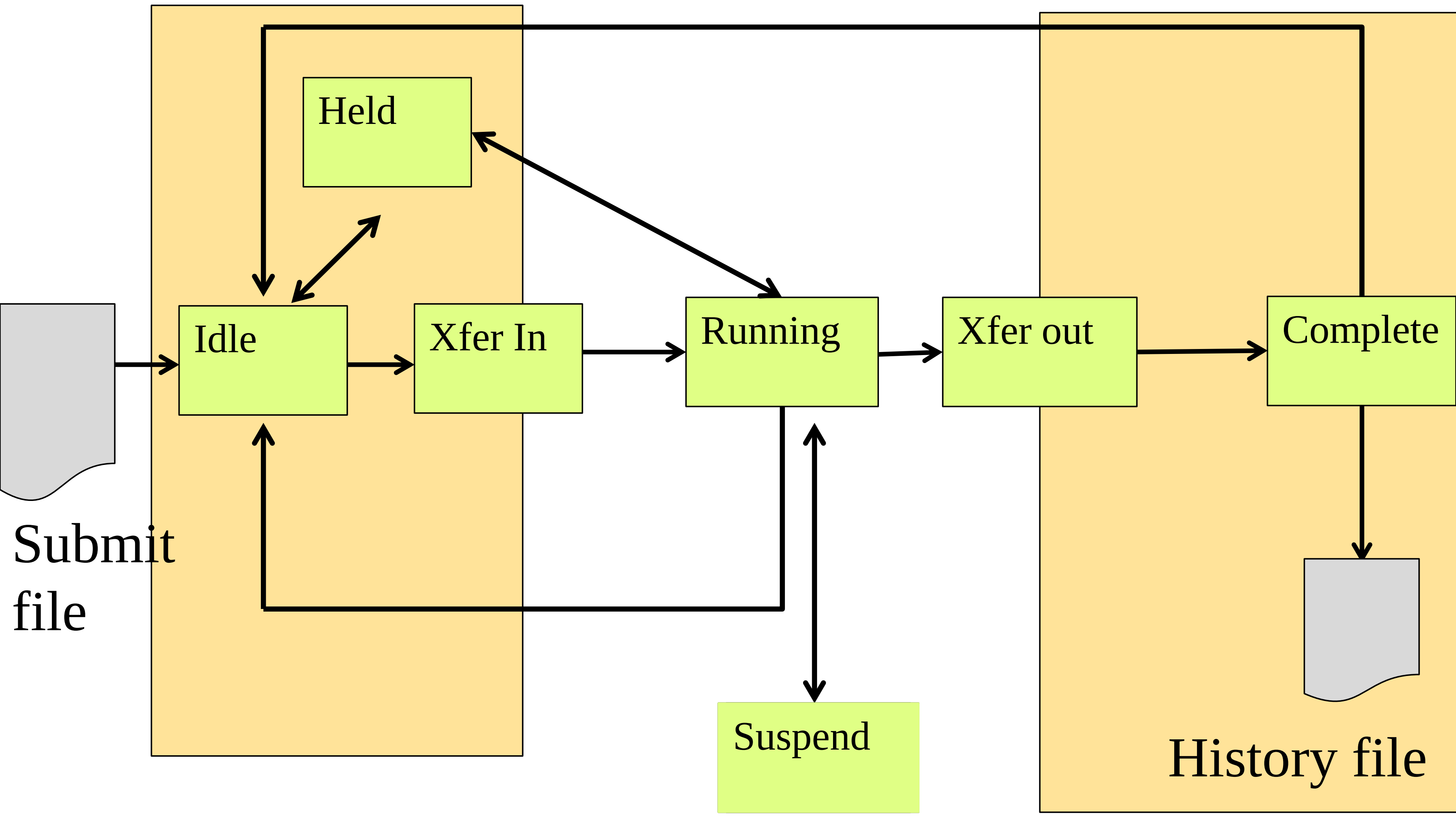
Life cycle of HTCondor Job



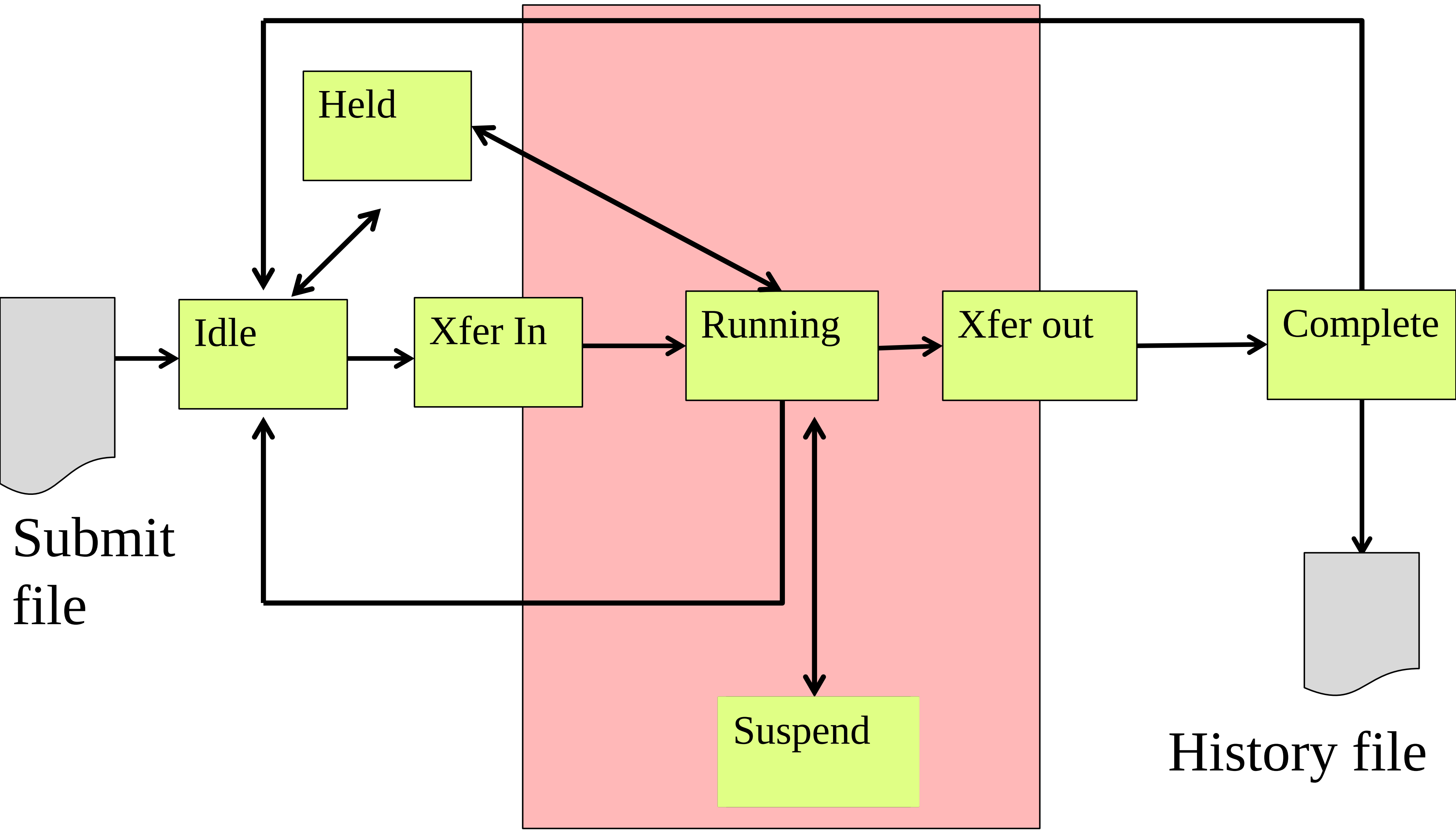
Life cycle of HTCondor Machine



"Submit Side"

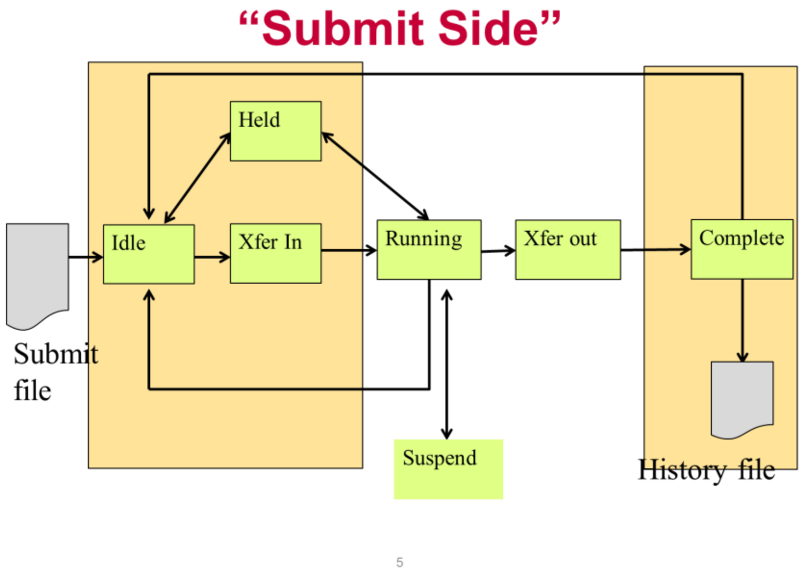

Held
Idle
Xfer In
Running
Xfer out
Complete
Suspend
Submit file
History file

“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



- 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

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

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

On to configuration…

Configuration File

- **`(Almost)`** all configure 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
```

Metaknobs

- One metaknob controls other knobs
- use ROLE : Personal

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 LEXIGRAPHIC 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 -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

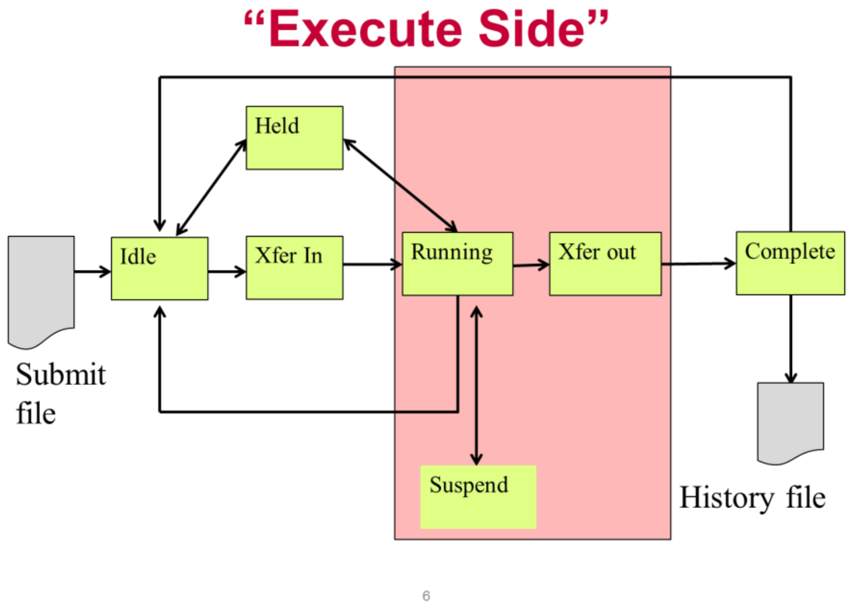
The Execute Side

Primarily managed by condor_startd process

With one condor_starter per running jobs

Sandboxes the jobs

Usually many per pool (support 10s of thousands)



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 (usableable 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

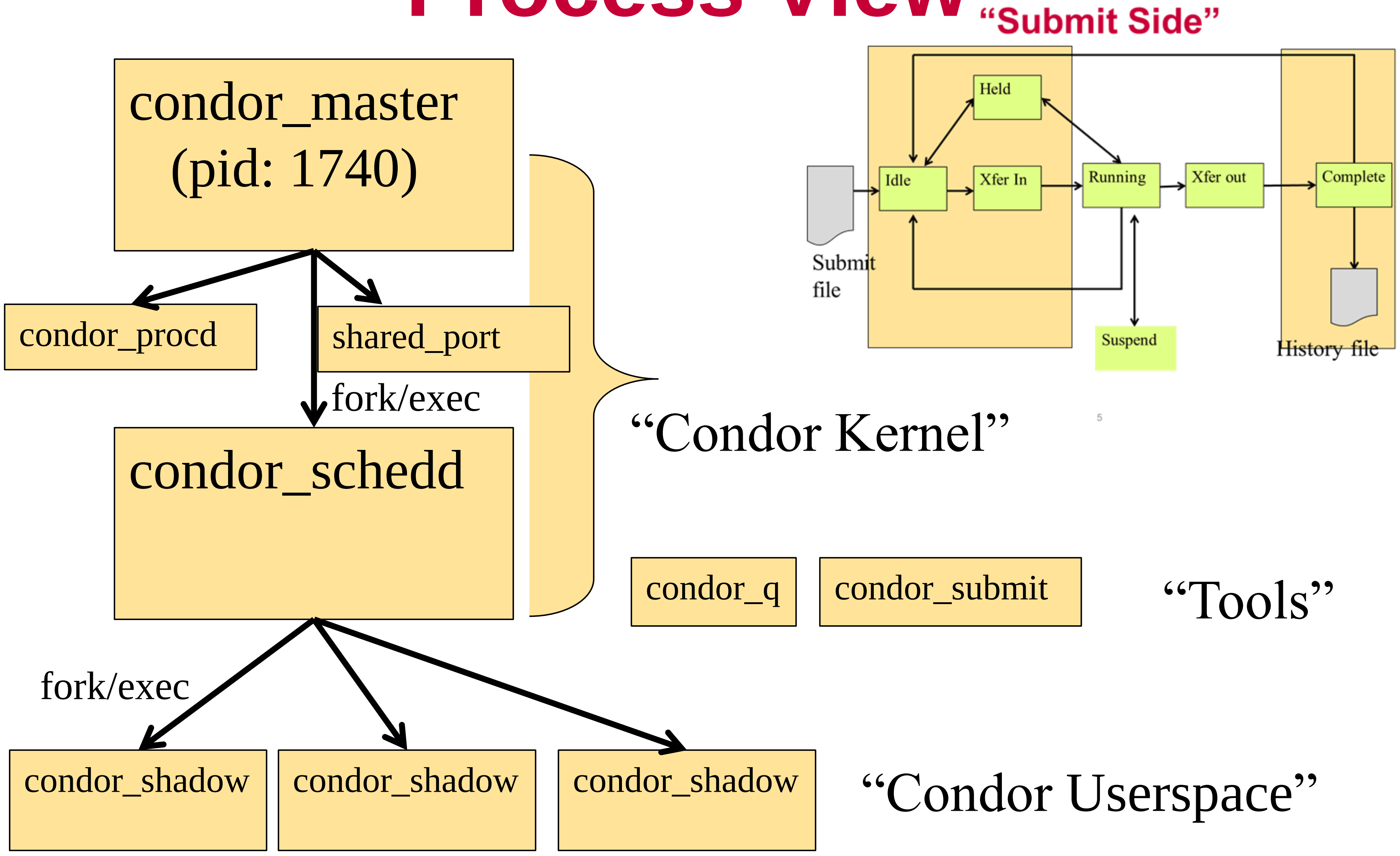
condor_startd: runs on execute machine

  condor_starter: one per job

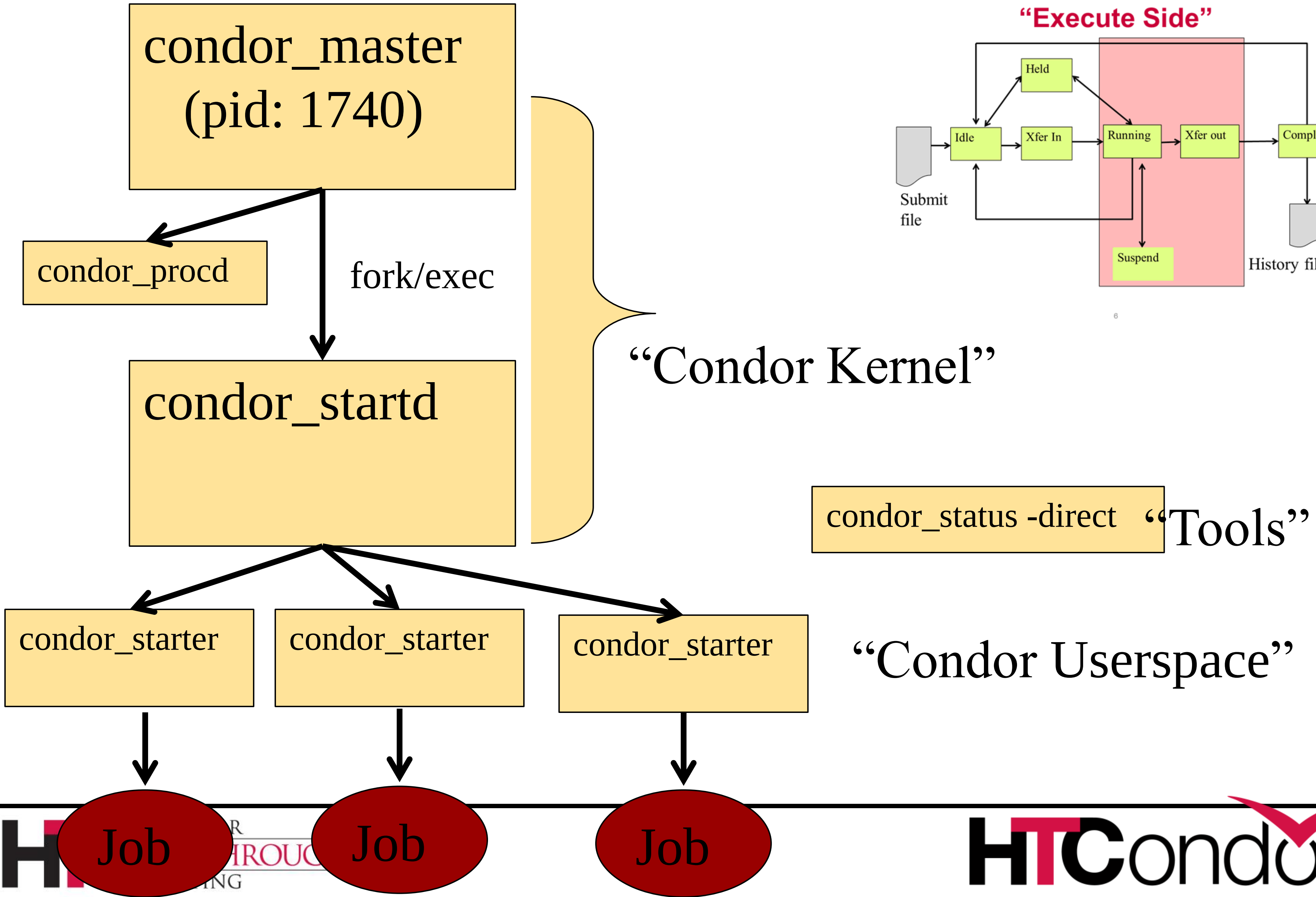
condor_negotiator/condor_collector

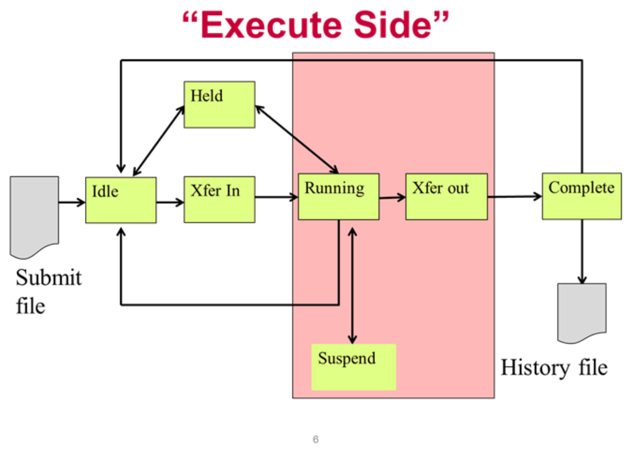
    one per pool

Process View

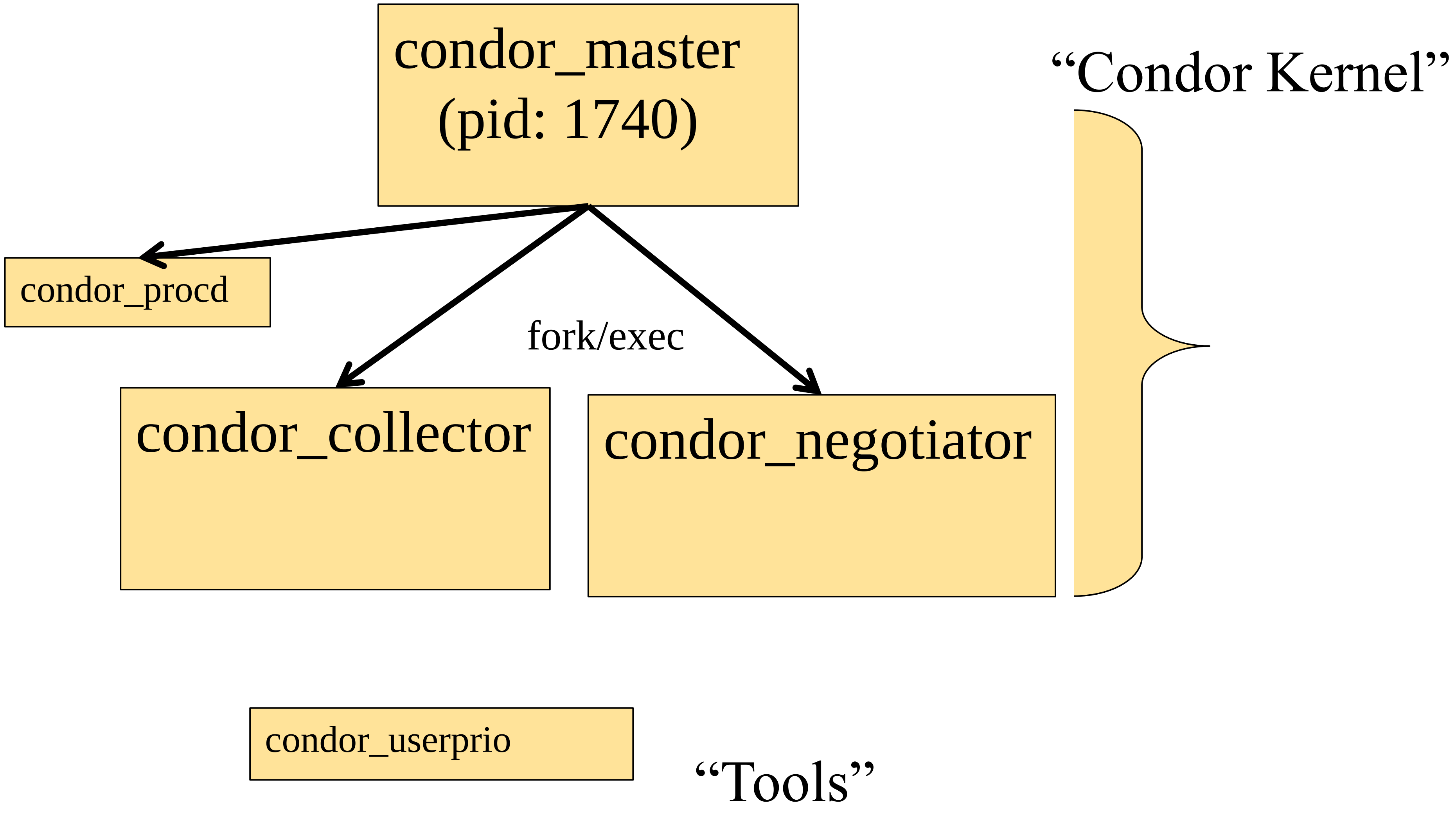


Process View: Execute





Process View: Central Manager



Condor Installation Basics

Let’s Install HTCondor

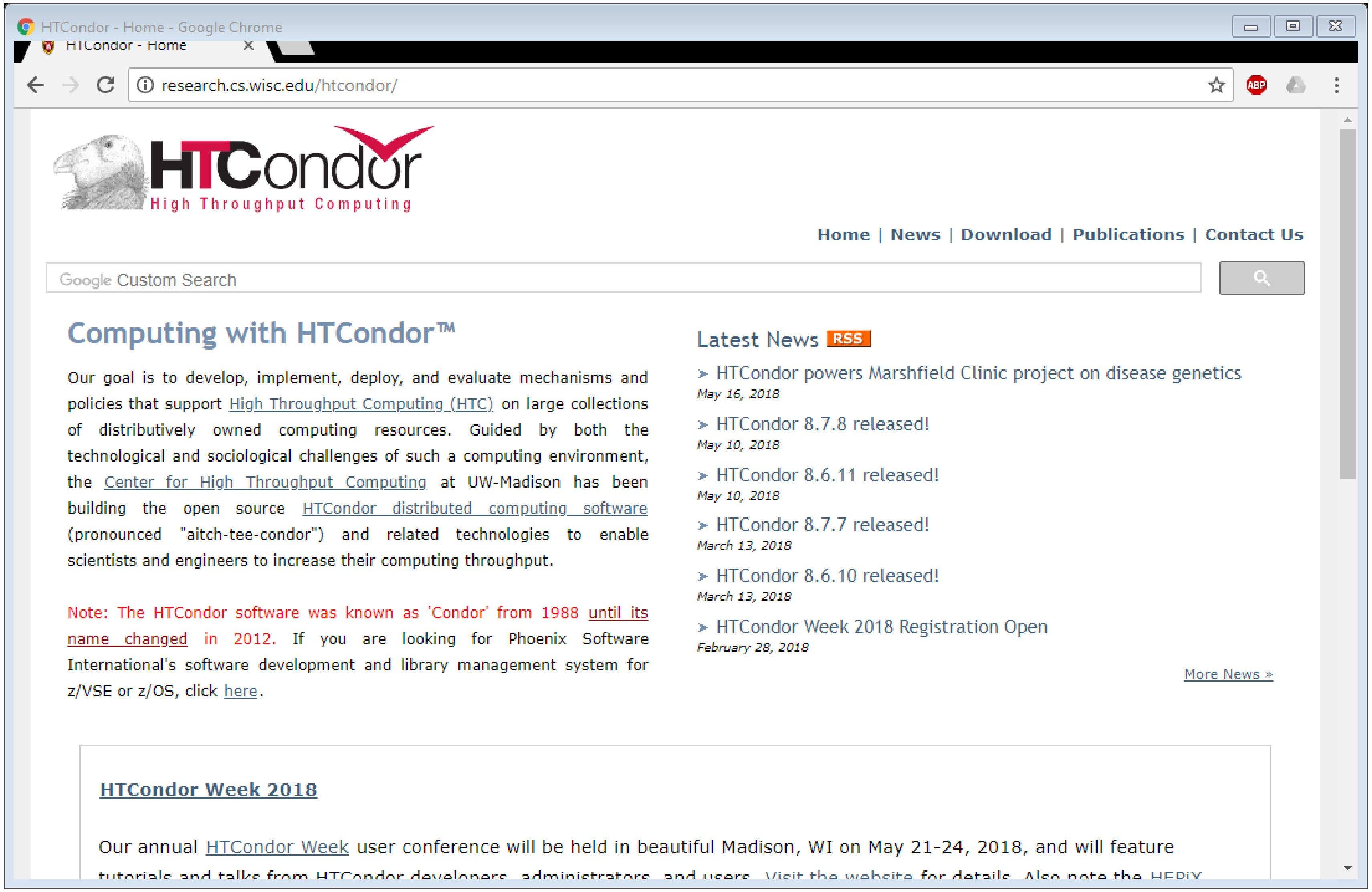
- Either with tarball
  - tar xvf htcondor-8.6.11-redhat6
- Or native packages

```
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
```

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

http://htcondorproject.org



HTCondor - Home - Google Chrome

HTCondor - Home

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

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 8.6.0, 8.6.5, 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_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

```
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 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!

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

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

- More detail in following talks…
- htcondor-users email list

- Talk to us!