



# Advanced Pool Management

John (TJ) Knoeller  
Condor Week 2016

# Overview

- › Two pools, both alike in dignity
- › Gotchas
- › Advanced configuration tips and tricks
- › Did I mention gotchas?
- › High Availability

# Best Practices

- › No real 'best'  
(But there are *Worse Practices*)
- › Two example pools
  - The CS Pool
  - The CHTC Pool

# The CS Pool

- › The Oldest HTCondor pool
  - Still mostly cycle scavenging
  - Runs every HTCondor stable build
- › HTCondor on shared file system
  - condor\_master on local disk
- › Uses the tarball/zip release
- › Upgrade is changing a symlink
  - Master restarts when it sees new binaries

# The CS Pool config

## › Root config is on local machine

```
ETC = /shr/condor/etc
```

```
GLOBAL = $(ETC)/condor_config.global
```

```
PLATFORM = $(ETC)/condor_config.$(OPSYS)
```

```
LOCAL = $(ETC)/hosts/$(HOSTNAME).local
```

```
LOCAL_CONFIG_FILE = $(GLOBAL), $(PLATFORM), $(LOCAL)
```

## › All other config files on shared file system

- Each machine has a private config file at  
`/shr/condor/etc/hosts/<hostname>.local`
- Each OS has an OS config file at\*  
`/shr/condor/etc/condor_config.<os>`

# The CHTC Pool

- › Puppet Installs HTCondor from RPM
  - Uses development release candidates
- › *We used to* push the config using Puppet
  - Slow to push out changes
  - Complicated puppet rules to vary config
- › *Now* Puppet pushes only the base config
  - Creates a git clone of the config repo
  - base config uses a script to git clean/pull

(Still working out the scaling problems)

# Config central

- › Keep your config files in source control
- › One set of config files for the whole pool
- › Fetch via git

```
LOCAL_CONFIG_FILE = \  
    git_script -s $(subsys) -h $(hostname) |
```

- › Fetch via condor\_urlfetch

```
LOCAL_CONFIG_FILE = condor_urlfetch \  
    -$(subsys) http://my.com?h=$(hostname) \  
    $(LOCALDIR)/urlconfig.cache |
```

# CHTC Pool upgrade cycle

- › Upgrades deployed gradually over 3 days
  1. A few execute nodes
  2. 1/3 of execute nodes
  3. A non-essential Schedd
  4. The Collector & Negotiator
  5. Most other Schedds
  6. The remaining execute nodes
  7. Repeat monthly (ish)

# An aside on upgrading

- › Upgrade execute nodes
  - Gracefully to maximize throughput
  - Peacefully to minimize badput
- › Upgrade Collector/Negotiator
  - Gracefully or Fast (there is no peaceful)
- › Upgrade Schedd
  - Fast to keep jobs running
  - Gracefully for extended shutdown

# Gotcha #1

- › There is no setting that will both
  - Shut down an Startd gracefully
  - Shut down a Schedd Fast

# 8.2 Power config

- › 8.2+ configuration language constructs
  - `$ (<param>:<default>)`
  - `include`
  - `use` (aka meta-knobs)
  - `if, else, elif, endif`
- › Have “backward parseable” flavors
  - `use, include, :if`
- › Have “backward fail” flavors
  - `@use, @include, if`

# 8.4 Power config

- > **\$INT** (*knob*, *format*)
- > **\$REAL** (*knob*, *format*)
  - Evaluate *knob* and printf with *format*
- > **\$CHOICE** (*knob*, *list*)
- > **\$CHOICE** (*knob*, *item*, *item*, *item*)
  - Evaluate *knob* as index into item list
- > **\$Fpdxq** (*file*)
  - Extract filename parts and strip/add quotes

# Substitution defaults

`$ (<param>:<default>)`

- › Is the value of `<param>` if it is defined, otherwise it is `<default>`

example:

```
NUM_SLOTS = $(NUM_CPUS:2)/2
```

Number of slots will be either half the number of cpus or it will be 1.

# Include :

- › Like LOCAL\_CONFIG\_FILE except
  - As many as you want
  - Nested
  - Read and parsed inline
- › Can include the output of a command
- › Macros on the include line substitute the current value, not the final one.

# Gotcha #2

- › Every daemon and every tool will
  - Read every config file
  - Run every config script (if any)
- › Sometimes several at the same time!
  - Scripts should have NO side effects
- › Config is read as root on startup but as condor on reconfig
  - All config files should be owned by root
  - World readable, root writable

# Example of Include

```
FILE = config.$(FULL_HOSTNAME)
Include : $(LOCAL_DIR)/$(FILE)
FILE = script.$(IP_ADDRESS)
Include : $(RELEASE_DIR)/$(FILE) |
Foo = bar
```

- › HTCondor 8.2+ Includes a file and the output of a script before parsing `Foo = bar`
- › HTCondor 8.0 sees

```
FILE = script.$(IP_ADDRESS)
Include = $(RELEASE_DIR)/$(FILE) |
Foo = bar
```



# Use (meta-knobs)

**use ROLE : Submit, Execute**

**use POLICY : Always\_Run\_Jobs**

**use SECURITY : User\_Based**

**use SECURITY : Strong**

- › Each keyword after colon expands inline to one or more configuration statements.
- › Defined when HTCCondor is built
  - See param\_info.in (mentioned earlier)

# Explore the meta-knobs

- › Categories are currently

`ROLE, FEATURE, POLICY, SECURITY`

- › Find out what options are available with

`condor_config_val use <category>`

- › Examine contents of a meta-knob with

`condor_config_val use <category>:<option>`

# If / Else

- › **If**, **Elif** support only basic conditionals
  - `[!] <boolean-or-number>`
  - `[!] defined <name>`
  - `[!] version [ $>$  $<$  $=$ ] = x.y[.z]`
- › No comparison or complex conditionals
  - **If** `version` is a special case
- › Conditional `$ (knob : 0)` is false when knob is not defined.

# Example of If / Else

```
If version >= 8.1.6
    use feature : gpus
else
    MACHINE_RESOURCE_GPUS = 0
endif
```

- › HTCondor 8.0 reports a syntax error!
  - `else` and `endif` lines have no operator



# Pre 8.2 compatible If / Else

```
:If version >= 8.1.6
:  use feature : gpus
:else
    MACHINE_RESOURCE_GPUS = 0
:endif
```

› HTCondor 8.0 only sees

```
MACHINE_RESOURCE_GPUS = 0
```

(because 8.0 ignores everything after the colon)\*

# Special macros for If

- › Magic “knobs” that are set based on who is parsing config

`$ (IsMaster)`

`$ (IsNegotiator)`

`$ (IsSchedd)`

`$ (IsShadow)`

`$ (IsStartd)`

`$ (IsStarter)`

`$ (IsTool)`

`$ (IsWindows)`

# Gotcha #3

condor\_config\_val output can differ from what the daemon sees if you use the \$(IsXXX) macros. You must use

```
condor_config_val -<daemon>
```

```
condor_config_val -subsys <daemon>
```

To see the effective config

# 8.4+ if tricks

```
HAVE_SCHEDD_DAEMON = \  
    stringListMember ("SCHEDD", "$ (DAEMON_LIST) ")  
  
if $INT (HAVE_SCHEDD_DAEMON)  
    MASTER_NEW_BINARY_RESTART = FAST  
else  
    MASTER_NEW_BINARY_RESTART = GRACEFUL  
endif
```

# Gotcha #4

`If` and `include` evaluate arguments inline  
So the previous example only works if it is  
after the last `DAEMON_LIST` in your config

# Gotcha #5

Line continuation behavior changed in 8.2

# Line continuation after comment

```
# We want to frob the bobulator \  
FROB_BOBULATOR = true
```

- › In 8.0 \ at the end of a comment line ‘eats’ the next line, so FROB\_BOBULATOR is not set
- › In 8.2+ \ at the end of a comment line is ignored, so every comment line needs its own #

# Comment after line continuation

```
ALLOW_WRITE = a.b.c.d \  
a.b.c.e \  
# a.b.c.x \  
a.b.c.z
```

- › In 8.0 you end up with # as a list member
- › In 8.2+ a.b.c.x is commented out.

# condor\_config\_val tricks

```
condor_config_val -schedd -verbose
```

- Ask the Schedd about it's config

```
condor_config_val -subsys schedd -verbose
```

- Parse the config as the schedd would

```
condor_config_val -writeconfig:upgrade -
```

- Write an 'upgrade' file containing *only* the knobs that you've changed

**HT**  
CENTER FOR  
HIGH THROUGHPUT  
COMPUTING

**HTC**Condor



**Any Questions?**