Condor Administration

Alan De Smet
Computer Sciences Department
University of Wisconsin-Madison
condor-admin@cs.wisc.edu
http://www.cs.wisc.edu/condor

Outline

- Condor Daemons
  - Job Startup
- Configuration Files
- Policy Expressions
  - Startd (Machine)
  - Negotiator
- Priorities
- Security
- Administration
- Installation
  - “Full Installation”
- Condor-G & C

Condor Daemons

Condor Daemons

- `condor_master` - controls everything else
- `condor_startd` - executing jobs
  - `condor_starter` - helper for starting jobs
- `condor_schedd` - submitting jobs
  - condor_shadow - submit-side helper

Condor Daemons

- `condor_collector` - Collects system information; only on Central Manager
- `condor_negotiator` - Assigns jobs to machines; only on Central Manager

Condor Daemons

- You only have to run the daemons for the services you want to provide
- **DAEMON_LIST** is a comma separated list of daemons to start
  - **DAEMON_LIST=MASTER,SCHEDD,STARTD**

condor_master

- Starts up all other Condor daemons
- If a daemon exits unexpectedly, restarts deamon and emails administrator
- If a daemon binary is updated (timestamp changed), restarts the daemon

condor_master

- Provides access to many remote administration commands:
  - condor_reconfig, condor_restart, condor_off, condor_on, etc.
- Default server for many other commands:
  - condor_config_val, etc.

condor_master

- Periodically runs **condor_preen** to clean up any files Condor might have left on the machine
  - Backup behavior, the rest of the daemons clean up after themselves, as well

condor_startd

- Represents a machine to the Condor pool
- Should be run on any machine you want to run jobs
- Enforces the wishes of the machine owner (the owner's "policy")

condor_startd

- Starts, stops, suspends jobs
- Spawns the appropriate **condor_starter**, depending on the type of job
- Provides other administrative commands (for example, **condor_vacate**)

condor_starter

- Spawned by the **condor_startd** to handle all the details of starting and managing the job
  - Transfer job's binary to execute machine
  - Send back exit status
  - Etc.

condor_starter

- On multi-processor machines, you get one **condor_starter** per CPU
  - Actually one per running job
  - Can configure to run more (or less) jobs than CPUs
- For PVM jobs, the starter also spawns a PVM daemon (**condor_pvmd**)

condor_schedd

- Represents jobs to the Condor pool
- Maintains persistent queue of jobs
  - Queue is not strictly FIFO (priority based)
  - Each machine running **condor_schedd** maintains its own queue
- Should be run on any machine you want to submit jobs from

condor_schedd

- Responsible for contacting available machines and spawning waiting jobs
  - When told to by **condor_negotiator**
- Services most user commands:
  - **condor_submit, condor_rm, condor_q**

condor_shadow

- Represents job on the submit machine
- Services requests from standard universe jobs for remote system calls
  - including all file I/O
- Makes decisions on behalf of the job
  - for example: where to store the checkpoint file

`condor_shadow` Impact

- One **`condor_shadow`** running on submit machine for each actively running Condor job
- Minimal load on submit machine
  - Usually blocked waiting for requests from the job or doing I/O
  - Relatively small memory footprint

Limiting `condor_shadow`

- Still, you can limit the impact of the shadows on a given submit machine:
  - They can be started by Condor with a "nice-level" that you configure (**`SHADOW_RENICE_INCREMENT`**)
  - Can limit total number of shadows running on a machine (**`MAX_JOBS_RUNNING`**)

condor_collector

- Collects information from all other Condor daemons in the pool
- Each daemon sends a periodic update called a ClassAd to the collector
- Services queries for information:
  - Queries from other Condor daemons
  - Queries from users (**condor_status**)

condor_negotiator

- Performs matchmaking in Condor
  - Pulls list of available machines and job queues from `condor_collector`
  - Matches jobs with available machines
  - Both the job and the machine must satisfy each other's requirements (2-way matching)
- Handles user priorities

Central Manager

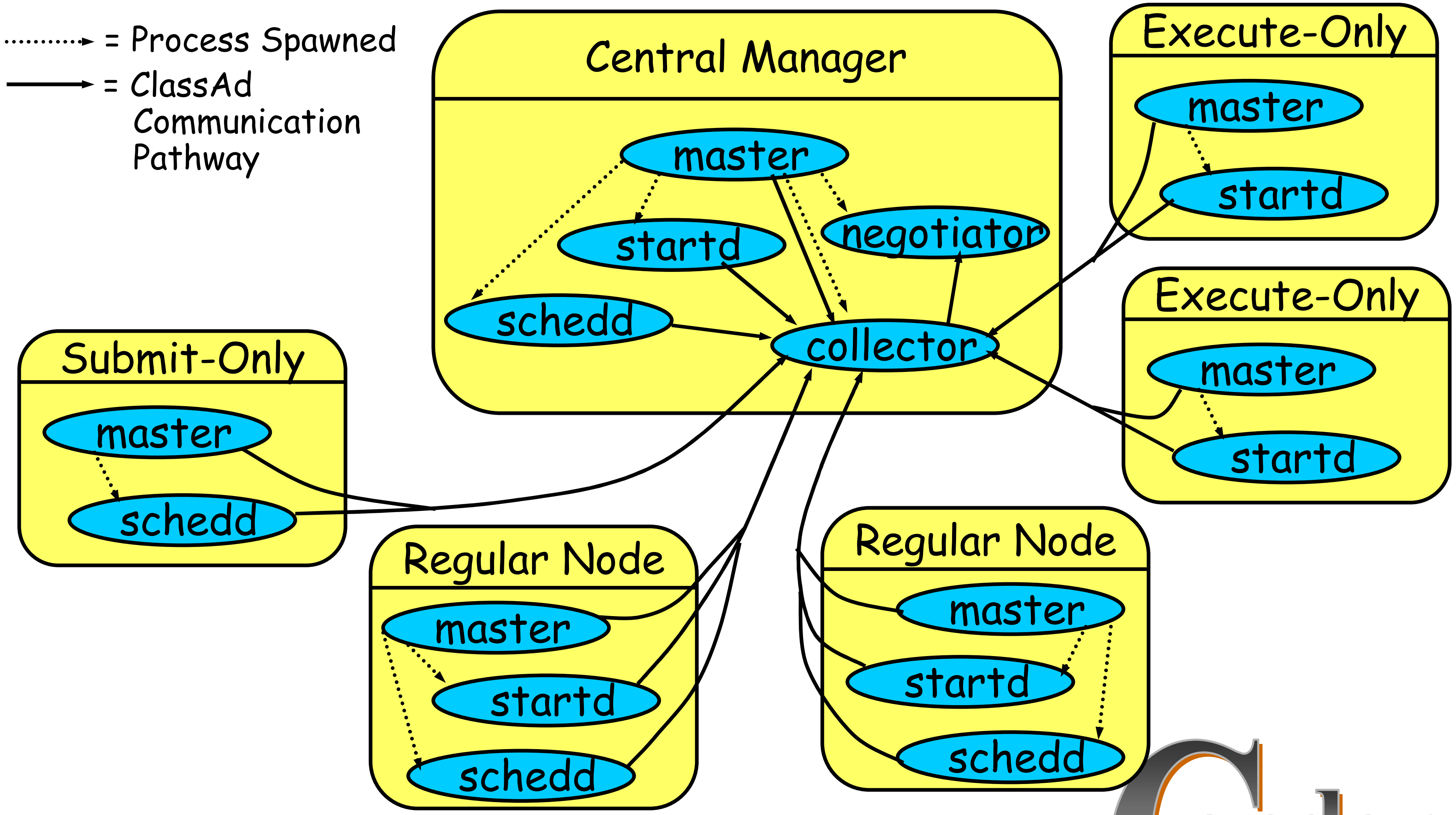
- The Central Manager is the machine running the collector and negotiator

```
DAEMON_LIST = MASTER,
  COLLECTOR, NEGOTIATOR
```

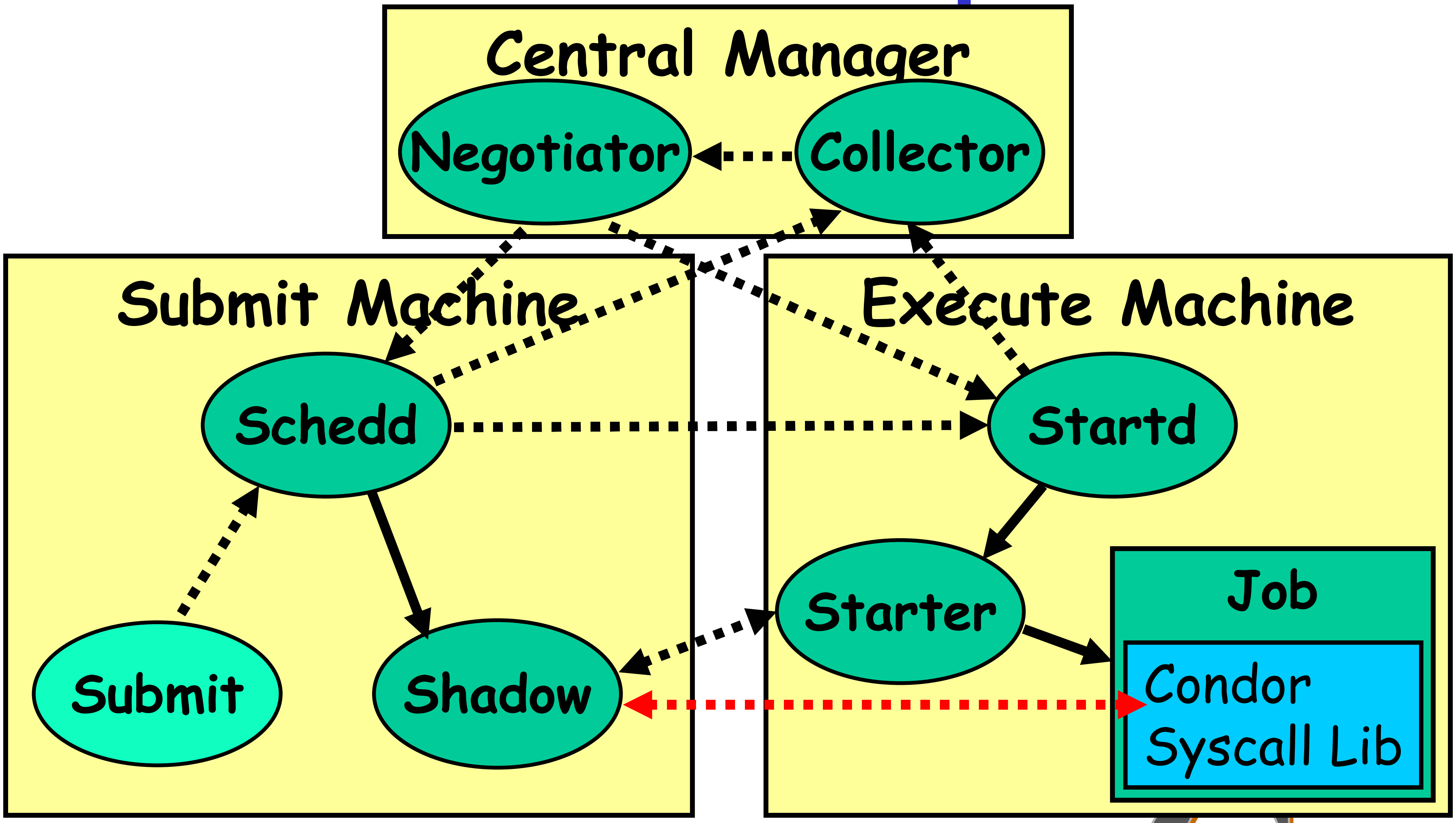
- Defines a Condor pool.

```
CONDOR_HOST =
  centralmanager.example.com
```

Typical Condor Pool



Job Startup


Central Manager
Negotiator
Collector
Submit Machine
Schedd
Submit
Shadow
Execute Machine
Startd
Starter
Job
Condor
Syscall Lib

Configuration Files

Configuration Files

- Multiple files concatenated
  - Definitions in later files overwrite previous definitions
- Order of files:
  - Global config file
  - Local config files, shared config files
  - Global and Local Root config file

Global Config File

- Found either in file pointed to with the **CONDOR_CONFIG** environment variable, **/etc/condor/condor_config**, or **~condor/condor_config**
- Most settings can be in this file
- Only works as a global file if it is on a shared file system

Other Shared Files

- **LOCAL_CONFIG_FILE** macro
  - Comma separated, processed in order
- You can configure a number of other shared config files:
  - Organize common settings (for example, all policy expressions)
  - platform-specific config files

Local Config File

- `LOCAL_CONFIG_FILE` macro (again)
  - Usually uses `$(HOSTNAME)`
- Machine-specific settings
  - local policy settings for a given owner
  - different daemons to run (for example, on the Central Manager!)

Local Config File

- Can be on local disk of each machine

```
/var/adm/condor/condor_config.local
```

- Can be in a shared directory

```
/shared/condor/condor_config.$(HOSTNAME)
/shared/condor/hosts/$(HOSTNAME)/
  condor_config.local
```

Root Config File (optional)

- Always processed last
- Allows root to specify settings which cannot be changed by other users
  - For example, the path to Condor daemons
- Useful if daemons are started as root but someone else has write access to config files

Root Config File (optional)

- **/etc/condor/condor_config.root** or **~condor/condor_config.root**
- Then loads any files specified in **ROOT_CONFIG_FILE_LOCAL**

Configuration File Syntax

- # at start of line is a comment
  - not allowed in names, confuses Condor.
- \ at the end of line is a line-continuation
  - Both lines are treated as one big entry
  - Works in comments!

Configuration File Macros

- Macros have the form:
  - `Attribute_Name = value`
    - Names are case insensitive
    - Values are case sensitive
- You reference other macros with:
  - `A = $(B)`
- 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
```

ClassAds

- Set of key-value pairs
- Can be matched against each other
  - Requirements and Rank
- This is old ClassAds
  - New, more expressive ClassAds exist
    - Not yet used in Condor

ClassAd Expressions

- Some configuration file macros specify expressions for the Machine's ClassAd
  - Notably START, RANK, SUSPEND, CONTINUE, PREEMPT, KILL
- Can contain a mixture of macros and ClassAd references
- Notable: UNDEFINED, ERROR

ClassAd Expressions

- +, -, *, /, <, <=,>, >=, ==, !=, &&, and || all work as expected
- TRUE==1 and FALSE==0 (guaranteed)

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

ClassAd Expressions: UNDEFINED and ERROR

- Special values
- Passed through most operators
  - Anything == UNDEFINED is UNDEFINED
- && and || eliminate if possible.
  - UNDEFINED && FALSE is FALSE
  - UNDEFINED && TRUE is UNDEFINED

ClassAd Expressions: =?= and =!=

- =?= and =!= are similar to == and !=
- =?= tests if operands have the same type and the same value.
  - `10 == UNDEFINED` -> UNDEFINED
  - `UNDEFINED == UNDEFINED` -> UNDEFINED
  - `10 =?= UNDEFINED` -> FALSE
  - `UNDEFINED =?= UNDEFINED` -> TRUE
- =!= inverts =?=

ClassAd Expressions

- Further information: Section 4.1, “Condor's ClassAd Mechanism,” in the Condor Manual.

Policy Expressions

Policy Expressions

- Allow machine owners to specify job priorities, restrict access, and implement local policies

Policy Expressions

- Specified in **`condor_config`**
- Policy evaluates both a machine ClassAd and a job ClassAd together
  - Policy can reference items in either ClassAd (See manual for list)
- Can reference **`condor_config`** macros: **`$(MACRONAME)`**

Machine (Startd) Policy Expression Summary

- **START** – When is this machine willing to start a job
  - Typically used to restrict access when the machine is being used directly
- **RANK** - Job preferences

Machine (Startd) Policy Expression Summary

- **SUSPEND** - When to suspend a job
- **CONTINUE** - When to continue a suspended job
- **PREEMPT** – When to nicely stop running a job
- **KILL** - When to immediately kill a preempting job

START

- START is the primary policy
- When FALSE the machine enters the Owner state and will not run jobs
- Acts as the Requirements expression for the machine, the job must satisfy START
  - Can reference job ClassAd values including Owner and ImageSize

RANK

- Indicates which jobs a machine prefers
  - Jobs can also specify a rank
- Floating point number
  - Larger numbers are higher ranked
  - Typically evaluate attributes in the Job ClassAd
  - Typically use + instead of &&

RANK

- Often used to give priority to owner of a particular group of machines
- Claimed machines still advertise looking for higher ranked job to preempt the current job

SUSPEND and CONTINUE

- When SUSPEND becomes true, the job is suspended
- When CONTINUE becomes true a suspended job is released

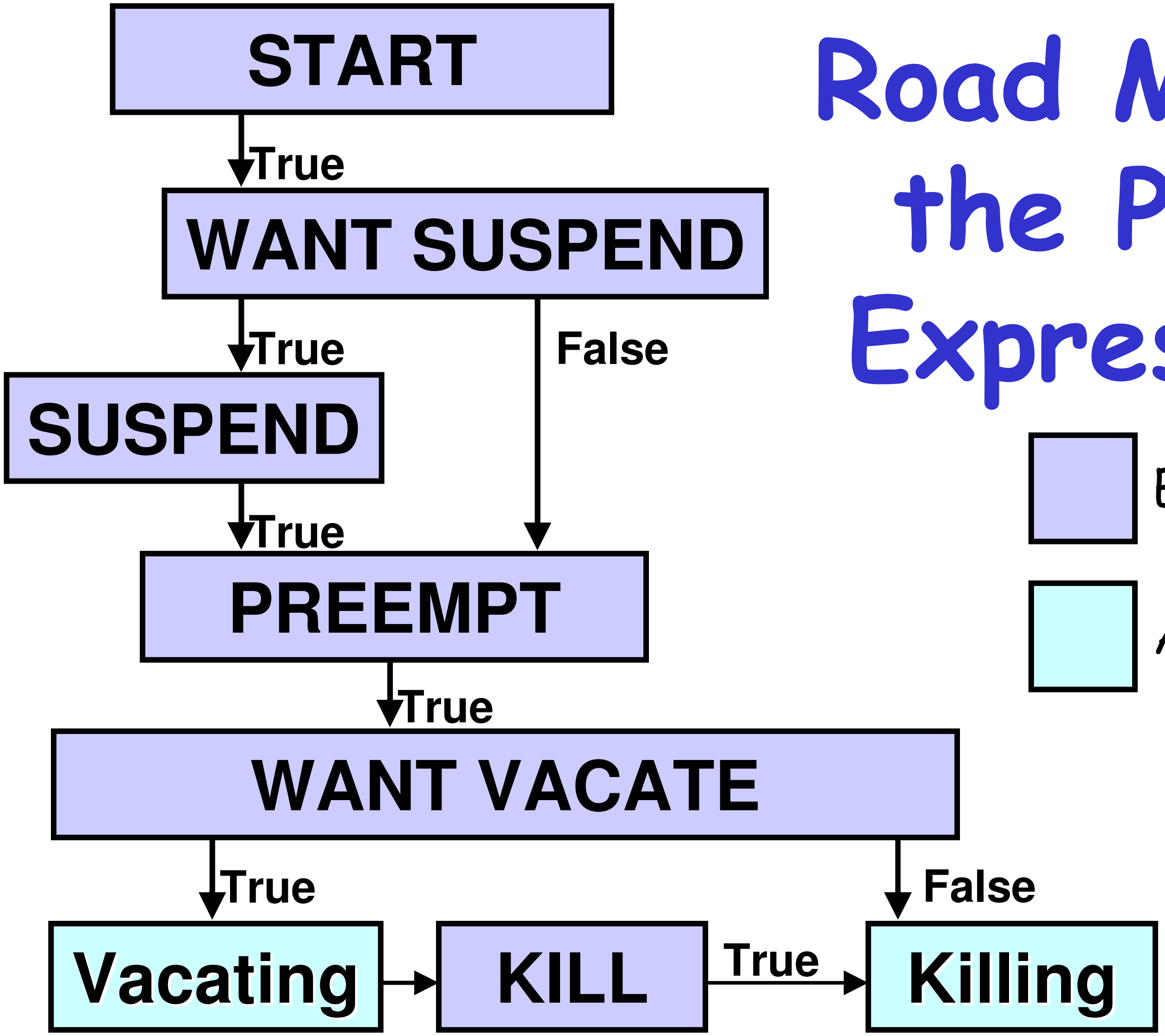
PREEMPT and KILL

- When PREEMPT becomes true, the job will be politely shut down
  - Vanilla universe jobs get SIGTERM
  - Standard universe jobs checkpoint
- When KILL becomes true, the job is SIGKILL
  - Checkpointing is aborted if started

WANT_SUSPEND and WANT_VACATE

- Typically leave both to TRUE
- **`WANT_SUSPEND`** - If false, skip **`SUSPEND test`**, jump to **`PREEMPT`**
- **`WANT_VACATE`**
  - If true, gives job time to vacate cleanly (until **`KILL`** becomes true)
  - If false, job is immediately killed (**`KILL`** is ignored)

Road Map of the Policy Expressions



START

True

WANT SUSPEND

True

False

SUSPEND

True

PREEMPT

True

WANT VACATE

True

False

Vacating

KILL

True

Killing

🟪 Expression

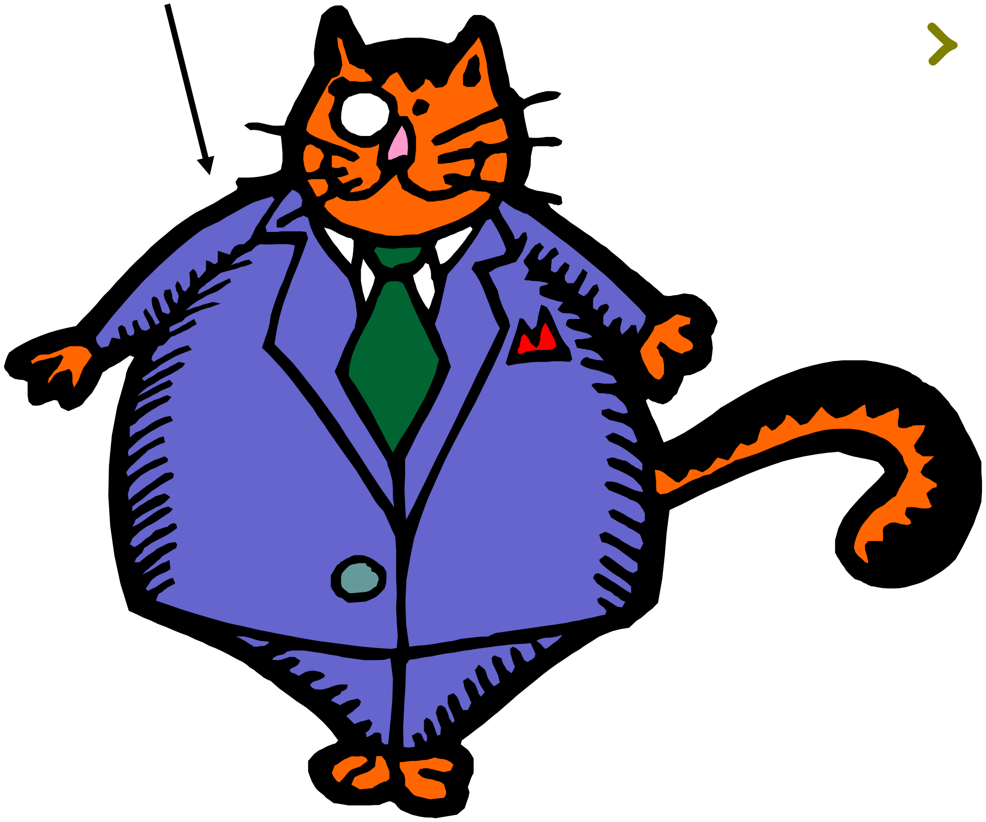
🟦 Activity

Minimal Settings

- Always runs jobs

```
START = True
RANK  =
SUSPEND  = False
CONTINUE  = True
PREEMPT = False
KILL  = False
```

Policy Configuration



(Boss Fat Cat)

- I am adding nodes to the Cluster… ***but the Chemistry Department has priority on these nodes***

New Settings for the Chemistry nodes

- Prefer Chemistry jobs

```
START = True
RANK = Department == "Chemistry"
SUSPEND = False
CONTINUE = True
PREEMPT = False
KILL = False
```

Submit file with Custom Attribute

- Prefix an entry with “+” to add to job ClassAd

```
Executable = charm-run
Universe = standard
+Department = Chemistry
queue
```

What if “Department” not specified?

```
START = True
RANK  = Department =!= UNDEFINED
  && Department == "Chemistry"
SUSPEND = False
CONTINUE = True
PREEMPT = False
KILL = False
```

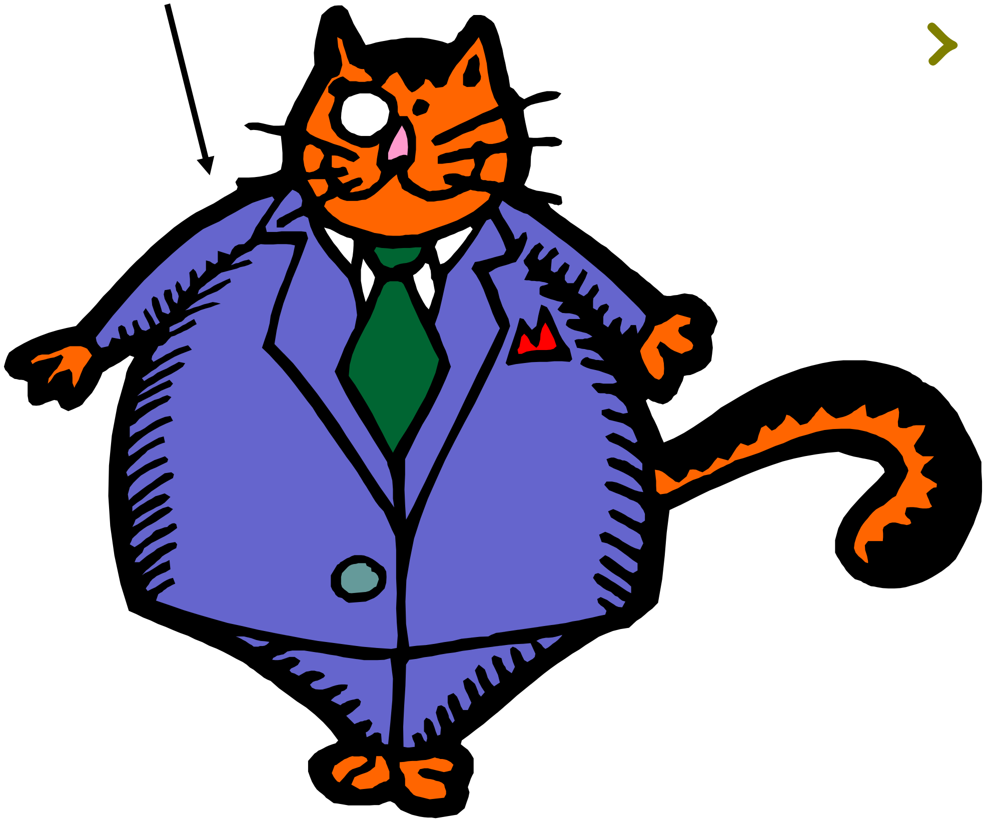
More Complex RANK

- Give the machine's owners (adesmet and roy) highest priority, followed by the Chemistry department, followed by the Physics department, followed by everyone else.

More Complex RANK

```
IsOwner = (Owner == "adesmet" ||
  Owner == "roy")
IsChem =(Department =!= UNDEFINED
  && Department == "Chemistry")
IsPhys =(Department =!= UNDEFINED
  && Department == "Physics")
RANK = $(IsOwner)*20 +
  $(IsChem)*10 + $(IsPhys)
```

Policy Configuration



(Boss Fat Cat)

- Cluster is okay, but… ***Condor can only use the desktops when they would otherwise be idle***

Defining Idle

- One possible definition:
  - No keyboard or mouse activity for 5 minutes
  - Load average below 0.3

Desktops should

- **START** jobs when the machine becomes idle
- **SUSPEND** jobs as soon as activity is detected
- **PREEMPT** jobs if the activity continues for 5 minutes or more
- **KILL** jobs if they take more than 5 minutes to preempt

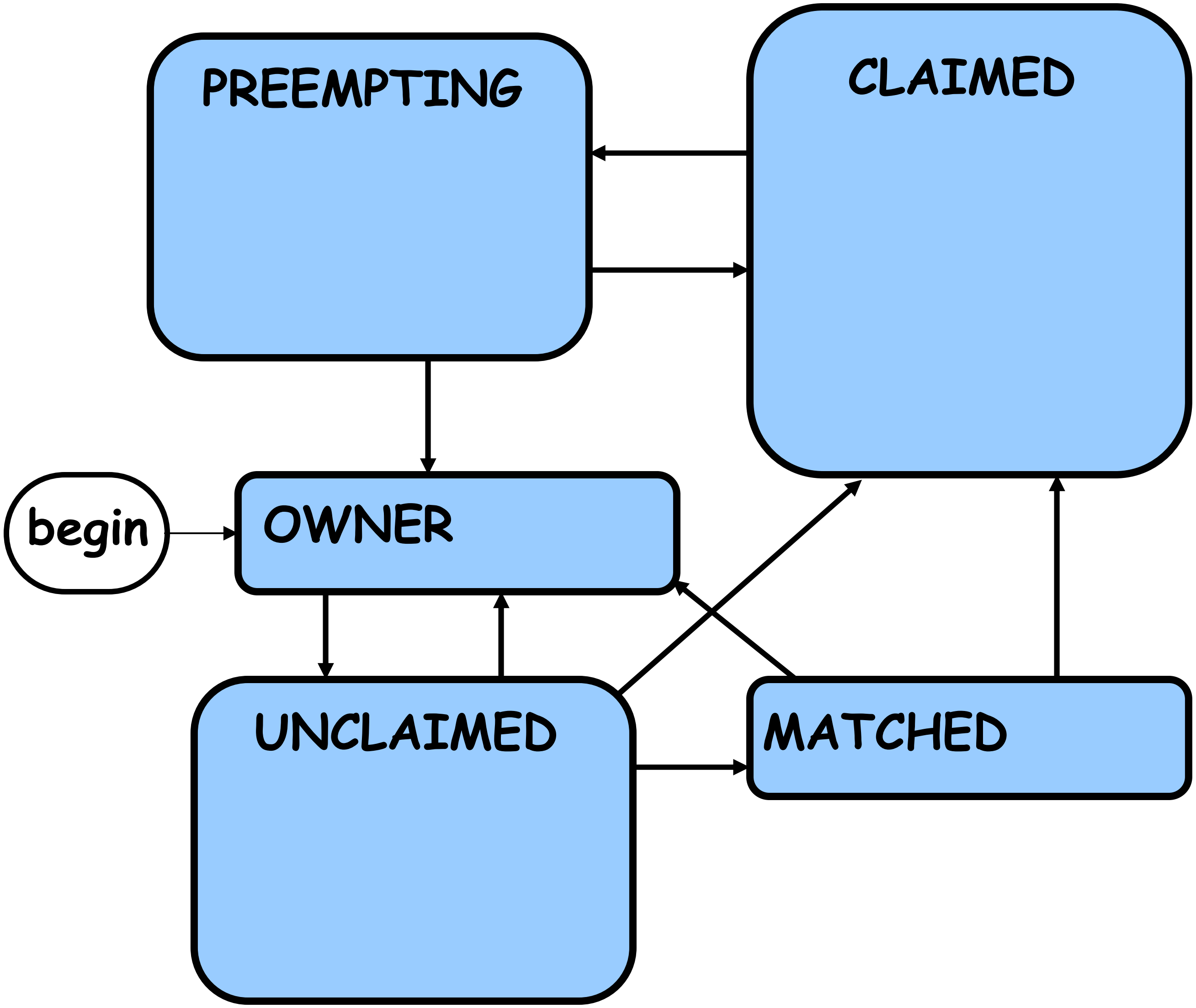
Macros in the Config File

```
NonCondorLoadAvg = (LoadAvg - CondorLoadAvg)
HighLoad = 0.5
BgndLoad = 0.3
CPU_Busy = ($(NonCondorLoadAvg) >=
  $(HighLoad))
CPU_Idle = ($(NonCondorLoadAvg) <=
  $(BgndLoad))
KeyboardBusy = (KeyboardIdle < 10)
MachineBusy = ($(CPU_Busy) ||
  $(KeyboardBusy))
ActivityTimer = \
    (CurrentTime - EnteredCurrentActivity)
```

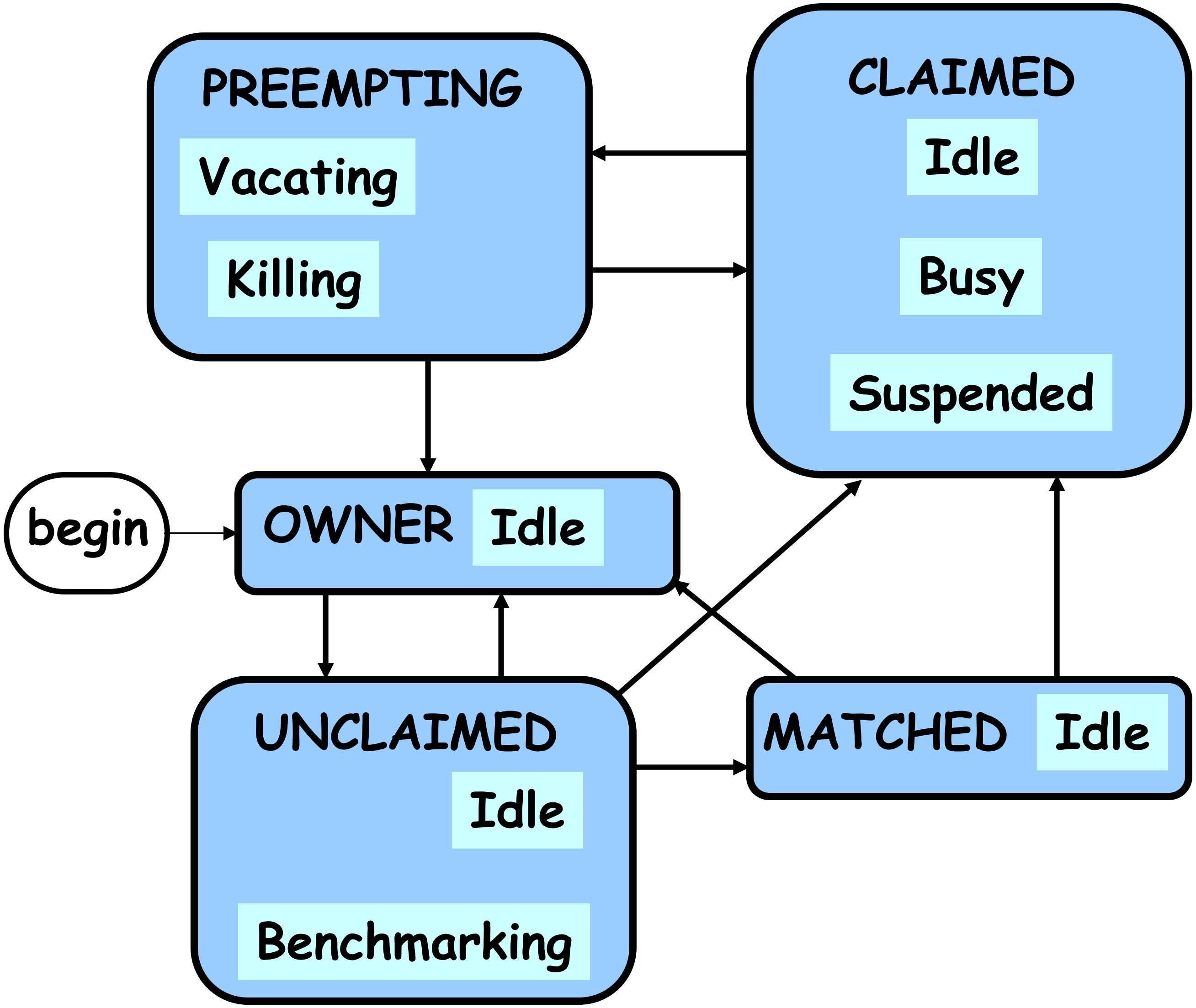
Desktop Machine Policy

```
START = $(CPU_Idle) && KeyboardIdle > 300
SUSPEND   = $(MachineBusy)
CONTINUE = $(CPU_Idle) && KeyboardIdle > 120
PREEMPT   = (Activity == "Suspended") && \
            $(ActivityTimer) > 300
KILL  = $(ActivityTimer) > 300
```

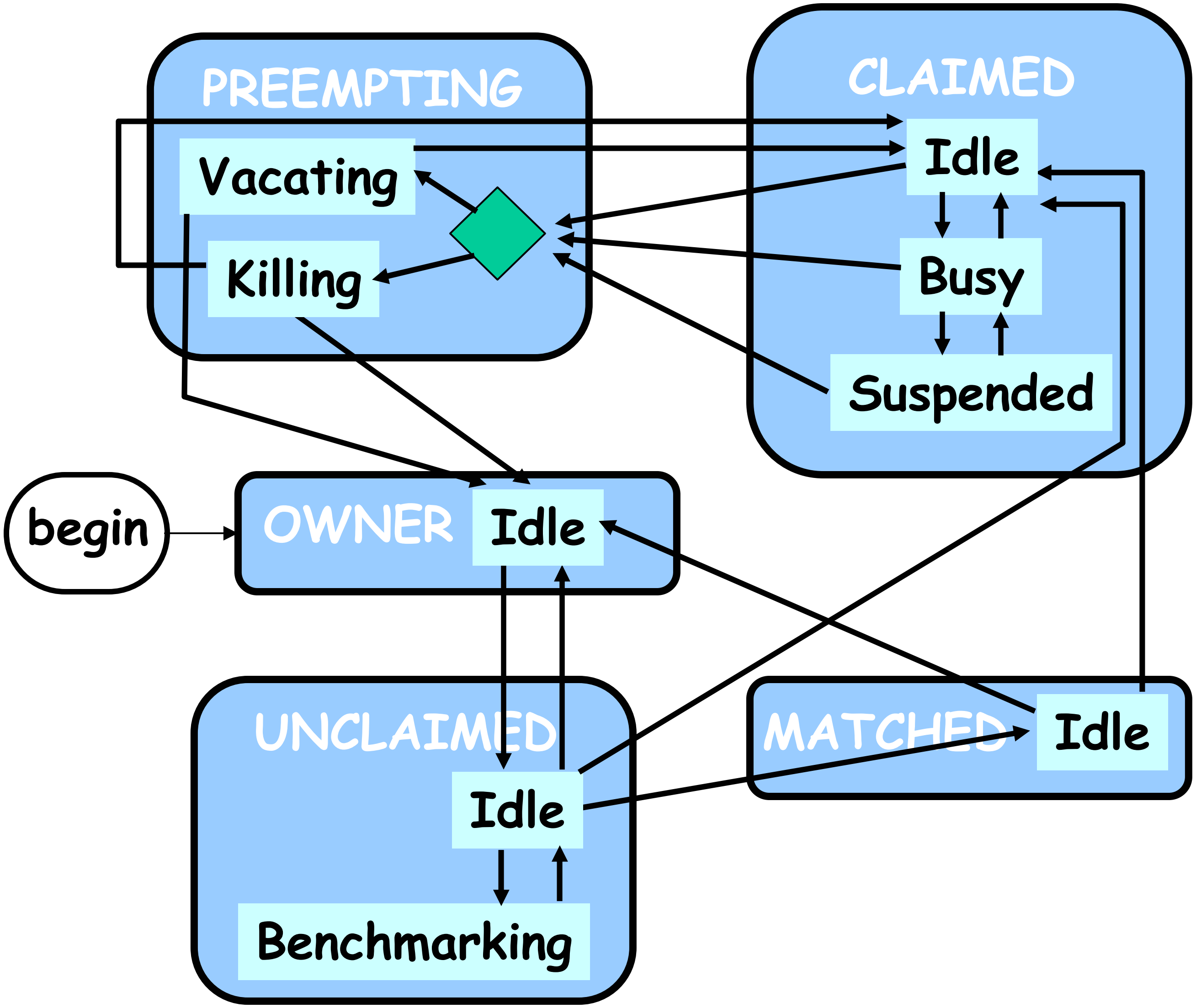
Machine States



Machine Activities



Machine Activities



See the manual for the gory details

(Section 3.6: Configuring the Startd Policy)

Custom Machine Attributes

- Can add attributes to a machine's ClassAd, typically done in the local config file

```
 INSTRUCTIONAL=TRUE
NETWORK_SPEED=100
STARTD_EXPRS=INSTRUCTIONAL,
  NETWORK_SPEED
```

Custom Machine Attributes

- Jobs can now specify Rank and Requirements using new attributes:

```
Requirements =
  (INSTRUCTIONAL=?=UNDEFINED ||
  INSTRUCTIONAL==FALSE)
Rank = NETWORK_SPEED =!=
  UNDEFINED && NETWORK_SPEED
```

Policy Review

- Users submitting jobs can specify Requirements and Rank expressions
- Administrators can specify Startd policy expressions individually for each machine
- Custom attributes easily added
- You can enforce almost any policy!

Further Machine Policy Information

- For further information, see section 3.6 “Startd Policy Configuration” in the Condor manual
- condor-users mailing list

  http://www.cs.wisc.edu/condor/mail-lists/
- condor-admin@cs.wisc.edu

Priorities

Job Priority

- Set with `condor_prio`
- Range from -20 to 20
- Only impacts order between jobs for a single user

User Priority

- Determines allocation of machines to waiting users
- View with **`condor_userprio`**
- Inversely related to machines allocated
  - A user with priority of 10 will be able to claim twice as many machines as a user with priority 20

User Priority

- Effective User Priority is determined by multiplying two factors
  - Real Priority
  - Priority Factor

Real Priority

- Based on actual usage
- Defaults to 0.5
- Approaches actual number of machines used over time
  - Configuration setting **PRIORITY_HALFLIFE**

Priority Factor

- Assigned by administrator
  - Set with **condor_userprio**
- Defaults to 1 (**DEFAULT_PRIO_FACTOR**)
- Nice users default to 1,000,000 (**NICE_USER_PRIO_FACTOR**)
  - Used for true bottom feeding jobs
  - Add "**nice_user=true**" to your submit file

Negotiator Policy Expressions

- **`PREEMPTION_REQUIREMENTS`** and **`PREEMPTION_RANK`**
- Evaluated when **`condor_negotiator`** considers replacing a lower priority job with a higher priority job
- Completely unrelated to the **`PREEMPT`** expression

PREEMPTION_REQUIREMENTS

- If false will not preempt machine
  - Typically used to avoid pool thrashing

```
PREEMPTION_REQUIREMENTS = \
 $(StateTimer) > (1 * $(HOUR)) \
 && RemoteUserPrio > SubmittorPrio * 1.2
```

  - Only replace jobs running for at least one hour and 20% lower priority

PREEMPTION_RANK

- Picks which already claimed machine to reclaim

```
PREEMPTION_RANK = \
  (RemoteUserPrio * 1000000)\
  - ImageSize
```

  - Strongly prefers preempting jobs with a large (bad) priority and a small image size

Security

Host/IP Address Security

- The basic security model in Condor
  - Stronger security available (Encrypted communications, cryptographic authentication)
- Can configure each machine in your pool to allow or deny certain actions from different groups of machines

Security Levels

- READ access - querying information
  - **`condor_status`**, **`condor_q`**, etc
- WRITE access - updating information
  - Does *not* include READ access!
  - **`condor_submit`**, adding nodes to a pool, etc

Security Levels

- ADMINISTRATOR access
  - `condor_on`, `condor_off`, `condor_reconfig`, `condor_restart`, etc.
- OWNER access
  - Things a machine owner can do (notably `condor_vacate`)

Setting Up Security

- List what hosts are allowed or denied to perform each action
  - If you list allowed hosts, everything else is denied
  - If you list denied hosts, everything else is allowed
  - If you list both, only allow hosts that are listed in “allow” but not in “deny”

Specifying Hosts

- There are many possibilities for specifying which hosts are allowed or denied:
  - Host names, domain names
  - IP addresses, subnets

Wildcards

- '*' can be used anywhere (once) in a host name
  - for example, "infn-corsi*.corsi.infn.it"
- '*' can be used at the end of any IP address
  - for example "128.105.101.*" or "128.105.*"

Setting up Host/IP Address Security

- Can define values that effect all daemons:
  - **HOSTALLOW_WRITE, HOSTDENY_READ, HOSTALLOW_ADMINISTRATOR**, etc.
- Can define daemon-specific settings:
  - **HOSTALLOW_READ_SCHEDD, HOSTDENY_WRITE_COLLECTOR**, etc.

Example Security Settings

```
HOSTALLOW_WRITE = *.infn.it
HOSTALLOW_ADMINISTRATOR= infn-corsi1*,\
  $(CONDOR_HOST), axpb07.bo.infn.it, \
  $(FULL_HOSTNAME)
HOSTDENY_ADMINISTRATOR = infn-corsi15
HOSTDENY_READ = *.gov, *.mil
HOSTDENY_ADMINISTRATOR_NEGOTIATOR = *
```

Default Security Settings

```
HOSTALLOW_ADMINISTRATOR =
  $(CONDOR_HOST)
HOSTALLOW_OWNER = $(FULL_HOSTNAME),
  $(HOSTALLOW_ADMINISTRATOR)
HOSTALLOW_READ = *
HOSTALLOW_WRITE = *
```

- Make write restrictive

```
HOSTALLOW_WRITE=*.site.uk
```

Advanced Security Features

- **AUTHENTICATION** – Who is allowed
- **ENCRYPTION** - Private communications, requires **AUTHENTICATION**.
- **INTEGRITY** - Checksums
- **NEGOTIATION** - Required for all others

Security Features

- Features individually set as **REQUIRED**, **PREFERRED**, **OPTIONAL**, or **NEVER**
- Can set default and for each level (**READ**, **WRITE**, etc)
- All default to **OPTIONAL**
- Leave **NEGOTIATION** at **OPTIONAL**

Authentication Complexity

- Authentication comes at a price: complexity
- Authentication between machines requires an authentication system
- Condor supports several existing authentication systems
  - We don't want to create yet another one

AUTHENTICATION_METHODS

- Authentication requires one or more methods:
  - FS
  - FS_REMOTE
  - GSI
  - Kerberos
  - NTSSPI
  - CLAIMTOBE

FS and FS_REMOTE Filesystem Tests

- FS checks that the user can create a file owned by the user.
  - Only works on local machine
  - Assumes the filesystem is trustworthy
- FS_REMOTE works remotely
  - Allows test file to be on NFS, AFS, or other shared file system

GSI
Globus Security Infrastructure

- Daemons and users have X.509 certs
- All Condor daemons in pool can share one certificate
- Map file maps from X.509 distinguished names to identities.

Kerberos and NTSSPI

- Kerberos
  - Complex to set up
  - If you are already using, easy to add to Condor
- NTSSPI - Windows NT
  - Only works on Windows

CLAIMTOBE

- Trust any claims about user identity
  - If used, encryption's secret password passed in clear!
  - Use with care

Additional Security Levels

- `CONFIG`
  - Dynamically change config settings
- `IMMEDIATE_FAMILY`
  - Daemon to daemon communications
- `NEGOTIATOR`
  - `condor_negotiator` to other daemons

ALLOW and DENY

- When authentication is enabled you can filter based on user identifier
- Use ALLOW and DENY instead of HOSTALLOW and HOSTDENY
- Can specify hostnames and IPs as before

Specifying User Identities

- username@site.example.com/hostname
- Can use * wildcard
- Hostname can be hostname or IP address with optional netmask

Example Filters

- Allow anyone from wisc.edu:

```
ALLOW_READ=*@wisc.edu/*.wisc.edu
```

- Allow any authorized local user:

```
ALLOW_READ=*/*.wisc.edu
```

- Allow specific user/machine

```
ALLOW_NEGOTIATOR=
  daemon@wisc.edu/condor.wisc.edu
```

Example Advanced Security Configuration

- Enable authentication, encryption, and integrity
- Use GSI authentication for between machine connections
- Use GSI or FS authentication on a single machine

Example Advanced Security Configuration

```
# Turn on all security:
SEC_DEFAULT_AUTHENTICATION=REQUIRED
SEC_DEFAULT_ENCRYPTION=REQUIRED
SEC_DEFAULT_INTEGRITY=REQUIRED
```

Example Advanced Security Configuration

```
# Require authentication
SEC_DEFAULT_AUTHENTICATION_METHODS
  = FS, GSI
```

Example Advanced Security Configuration

```
ALLOW_READ = *
ALLOW_WRITE= *@wisc.edu/*.wisc.edu
DENY_WRITE = abuser@*.wisc.edu/*
ALLOW_ADMINISTRATOR =
  admin@wisc.edu/*.wisc.edu,
  *@wisc.edu/$(CONDOR_HOST)
```

Example Advanced Security Configuration

```
ALLOW_CONFIG =
  $(ALLOW_ADMINISTRATOR)
ALLOW_IMMEDIATE_FAMILY =
  daemon@wisc.edu/*.wisc.edu
```

Example Advanced Security Configuration

```
ALLOW_OWNER =
  $(ALLOW_ADMINISTRATOR),
  $(FULL_HOSTNAME)
ALLOW_NEGOTIATOR =
  daemon@wisc.edu/
  $(CONDOR_HOST)
```

Users without Certs

- Using **`FS`** authentication users can submit jobs and check the local queue
- **`condor_status`** won't work for normal users without an X.509 Cert
  - Requires **`READ`** access to **`condor_collector`**
- Can let anyone read any daemon!

Allow Any User Read Access

```
# Using dreaded CLAIMTOBE
SEC_READ_AUTHENTIATION_METHODS =
  FS, GSI, CLAIMTOBE
```

Advanced Security Features

- For further details
  - Chapter 3.7, “Security in Condor” in the Condor Manual
  - condor-admin@cs.wisc.edu

Administration

condor_config_val

- Find current configuration values

```
% condor_config_val MASTER_LOG
/var/condor/logs/MasterLog
```

condor_config_val -v

- Can identify source

```
% condor_config_val -v CONDOR_HOST
CONDOR_HOST: condor.cs.wisc.edu
  Defined in
  ‘/etc/condor_config.hosts’, line 6
```

condor_fetchlog

- Retrieve logs remotely

```
condor_fetchlog
  beak.cs.wisc.edu Master
```

Querying daemons `condor_status`

- Queries the collector for information about daemons in your pool
- Defaults to finding `condor_startds`
  - `condor_status -schedd` summarizes all job queues
  - `condor_status -master` returns list of all `condor_masters`

condor_status

- `-long` displays the full ClassAd
- Specifiy a machine name to limit results to a single host

```
condor_status -l
  node4.cs.wisc.edu
```

condor_status -constraint

- Only return ClassAds that match an expression you specify
- Show me idle machines with 1GB or more memory

  -
    ```
    condor_status -constraint 'Memory >= 1024 && Activity == "Idle"`
    ```

condor_status -format

- Controls format of output
- Useful for writing scripts
- Uses C printf style formats
  - One field per argument

condor_status -format

- Census of systems in your pool:

```
% condor_status -format '%s '
  Arch -format '%s\n' OpSys |
  sort | uniq -c
     797 INTEL LINUX
     118 INTEL WINNT50
     108 SUN4u SOLARIS28
       6 SUN4x SOLARIS28
```

Examinging Queues
`condor_q`

- View the job queue
- The "**`-long`**" option is useful to see the entire ClassAd for a given job
- supports **`-constraint`** and **`-format`**
- Can view job queues on remote machines with the "**`-name`**" option

condor_q -format

- Census of jobs per user

```
% condor_q -format '%8s ' Owner
  -format '%s\n' Cmd | sort |
  uniq -c
   64 adesmet /scratch/submit/a.out
    2 adesmet /home/bin/run_events
    4   smith /nfs/sim1/em2d3d
    4   smith /nfs/sim2/em2d3d
```

`condor_q -analyze`

- `condor_q` will try to figure out why the job isn't running
- Good at determining that no machine matches the job Requirements expressions

condor_q -analyze

- Typical results:

```
471216.000:  Run analysis summary.  Of 820 machines,
    458 are rejected by your job's requirements
     25 reject your job because of their own requirements
      0 match, but are serving users with a better priority in the pool
      4 match, but prefer another specific job despite its worse user-
   priority
      6 match, but will not currently preempt their existing job
    327 are available to run your job
```

condor_analyze

- Available in Condor 6.5 and beyond
- Breaks down the job's requirements and suggests modifications

condor_analyze

- (Heavily truncated output)

```
The Requirements expression for your job is:
( ( target.Arch == "SUN4u" ) && ( target.OpSys ==
    "WINNT50" ) && [snip]
Condition                        Machines  Suggestion
1 (target.Disk > 100000000)    0      MODIFY TO 14223201
2 (target.Memory > 10000)      0      MODIFY TO 2047
3 (target.Arch == "SUN4u")     106
4 (target.OpSys == "WINNT50") 110    MOD TO "SOLARIS28"

Conflicts: conditions: 3, 4
```

Condor's Log Files

- Condor maintains one log file per daemon

Condor's Log Files

- Can increase verbosity of logs on a per daemon basis
  - SHADOW_DEBUG, SCHEDD_DEBUG, and others
  - Space separated list

Useful Debug Levels

- **D_FULLDEBUG** dramatically increases information logged
- **D_COMMAND** adds information about about commands received

```
 SHADOW_DEBUG = \
      D_FULLDEBUG D_COMMAND
```

Condor's Log Files

- Log files are automatically rolled over when a size limit is reached
  - Defaults to 64000 bytes, you will probably want to increase.
  - Rolls over quickly with **`D_FULLDEBUG`**
  - **`MAX_*_LOG`**, one setting per daemon
    - **`MAX_SHADOW_LOG`**, **`MAX_SCHEDD_LOG`**, and others

Condor's Log Files

- Many log files entries primarily useful to Condor developers
  - Especially if D_FULLDEBUG is on
  - Minor errors are often logged but corrected
  - Take them with a grain of salt
  - `condor-admin@cs.wisc.edu`

Debugging Jobs: condor_q

- Examine the job with `condor_q`
  - especially `-long` and `-analyze`
  - Compare with `condor_status -long`

Debugging Jobs: User Log

- Examine the job's user log
  - Can find with:

```
condor_q -format '%s\n' UserLog 17.0
```

  - Set with "log" in the submit file
- Contains the life history of the job
- Often contains details on problems
  - Condor 6.6 includes improved messages

Debugging Jobs: ShadowLog

- Examine `ShadowLog` on the submit machine
  - Note any machines the job tried to execute on
  - There is often an "ERROR" entry that can give a good indication of what failed

Debugging Jobs: Matching Problems

- No **ShadowLog** entries? Possible problem matching the job.
  - Examine **ScheddLog** on the submit machine
  - Examine **NegotiatorLog** on the central manager

Debugging Jobs: Local Problems

- **ShadowLog** entries suggest an error but aren't specific?
  - Examine **StartLog** and **StarterLog** on the execute machine

Debugging Jobs: Reading Log Files

- Condor logs will note the job ID each entry is for
  - Useful if multiple jobs are being processed simultaneously
  - grepping for the job ID will make it easy to find relavent entries

Debugging Jobs: What Next?

- If necessary add "**D_FULLDEBUG D_COMMAND**" to **DEBUG_*DAEMONNAME*** setting for additional log information
- Increase **MAX_*DAEMONNAME*_LOG** if logs are rolling over too quickly
- If all else fails, email us
  - condor-admin@cs.wisc.edu

Installation

Considerations for Installing a Condor Pool

- What machine should be your central manager?
- Does your pool have a shared file system?
- Where to install Condor binaries and configuration files?
- Where should you put each machine's local directories?
- Start the daemons as root or as some other user?

What machine should be your central manager?

- The central manager is very important for the proper functioning of your pool
- If the central manager crashes, jobs that are currently matched will continue to run, but new jobs will not be matched

Central Manager

- Want assurances of high uptime or prompt reboots
- A good network connection helps

Does your pool have a shared file system?

- It is easier to run vanilla universe jobs if so, but one is not required
- Shared location for configuration files can ease administration of a pool
- AFS can work, but Condor does not yet manage AFS tokens

Where to install binaries and configuration files?

- Shared location for configuration files can ease administration of a pool
- Binaries on a shared file system makes upgrading easier, but can be less stable if there are network problems
  - `condor_master` on the local disk is a good compromise

Where should you put each machine's local directories?

- You need a fair amount of disk space in the spool directory for each **`condor_schedd`** (holds job queue and binaries for each job submitted)
- The execute directory is used by the **`condor_starter`** to hold the binary for any Condor job running on a machine

Where should you put each machine's local directories?

- The log directory is used by all daemons
  - More space means more saved info

Hostnames

- Any two machines that will be communicating must know each others names
- You can't have nameless machines

Start the daemons as root or some other user?

- If possible, we recommend starting the daemons as root
  - Jobs run as the user that submitted them
    - More secure
    - Less confusion for users
  - Condor will try to run as the user "condor" whenever possible

Running Daemons as Non-Root

- Condor will still work, users just have to take some extra steps to submit jobs
- Can have "personal Condor" installed - only you can submit jobs

Basic Installation Procedure

- 1. Decide what version and parts of Condor to install and download them
- 2. Install the "release directory" - all the Condor binaries and libraries
- 3. Setup the Central Manager
- 4. (optional) Setup Condor on any other machines you wish to add to the pool
- 5. Spawn the Condor daemons

Condor Version Series

- We distribute two versions of Condor
  - Stable Series
  - Development Series

Stable Series

- Heavily tested
- Recommended for general use
- 2nd number of version string is even (6.6.3)

Development Series

- Latest features, not necessarily well-tested
- Not recommended unless you're willing to work with beta code or need new features
- 2nd number of version string is odd (6.7.0)

Condor Versions

- What am I running?
- All daemons advertise a `CondorVersion` attribute in the ClassAd they publish
- You can also view the version string by running `ident` on any Condor binary

Condor Versions

- All parts of Condor on a single machine should run the same version!
- Machines in a pool can usually run different versions and communicate with each other
- Documentation will specify when a version is incompatible with older versions

Downloading Condor

- Go to http://www.cs.wisc.edu/condor/
- Fill out the form and download the different pieces you need
  - Normally, you want the full stable release
- There are also "contrib" modules for non-standard parts of Condor
  - For example, the View Server

Downloading Condor

- Distributed as compressed “tar” files
- Once you download, unpack them

Install the Release Directory

- In the directory where you unpacked the tar file, you'll find a **`release.tar`** file with all the binaries and libraries
- **`condor_configure`** can help manage the installation

condor_configure

- Handles installation and reconfiguration

```
condor_configure --install
 --install-dir=/nfs/opt/condor
 --local-dir=/var/condor
 --owner=condor
```

Install the Release Directory

- In a pool with a shared release directory, you should run **`condor_configure`** somewhere with write access to the shared directory
- You need a *separate* release directory for *each platform*!

Setup the Central Manager

- Central manager needs specific configuration to start the `condor_collector` and `condor_negotiator`
  - `condor_configure --type=manager`
    - or
  - `DAEMON_LIST = master, collector, negotiator`

Setup Additional Machines

- If you have a shared file system, just run **`condor_init`** on any other machine you wish to add to your pool
  - Created local directories
- Without a shared file system, you must run **`condor_configure`** on each host

Start the Condor daemons

- Run **condor_master** to start Condor
  - Remember to start as root if desired
- Start Condor on the central manager first
- Add Condor to your boot scripts?
  - We provide a “SysV-style” init script (**<release>/etc/examples/condor.boot**)

Shared Release Directory

- Simplifies administration

Shared Release Directory

- Unifies configuration files, simplifying changes
  - Same shared global config file for all machines
  - All local config files visible in one place
    - Can symlink local files for multiple machines to a single file

Shared Release Directory

- Keep all of your binaries in one place
  - Prevents having different versions accidentally left on different machines
  - Easier to upgrade

"Full Installation" of `condor_compile`

- `condor_compile` re-links user jobs with Condor libraries to create "standard" universe jobs.
- By default, only works with certain commands (`gcc, g++, g77, cc, CC, f77, f90, ld`)
- With a "full-installation", works with any command (notably, `make`)

"Full Installation" of `condor_compile`

- Move real **`ld`** binary, the linker, to **`ld.real`**
  - Location of **`ld`** varies between systems, typically **`/bin/ld`**
- Install Condor's **`ld`** script in its place
- Transparently passes to **`ld.real`** by default; during **`condor_compile`** hooks in Condor libraries.

Other Installation Options

- VDT – Virtual Data Toolkit
  - PacMan installer
  - Includes other Grid software
  - http://www.lsc-group.phys.uwm.edu/vdt/
- RPM

Condor-G and Condor-C

Condor-G

- Transfers jobs to other systems (typically Grids)
  - Globus Toolkit
    - GT2 in Condor 6.6
    - GT2, 3, and 4 in Condor 6.7
  - Experimental: Unicore, Oracle, Nordugrid, other batch systems like PBS or LSF

Condor-G

- By default: Immediately runs job at specified site / resource
- Can use matchmaking
  - Complex, contact us

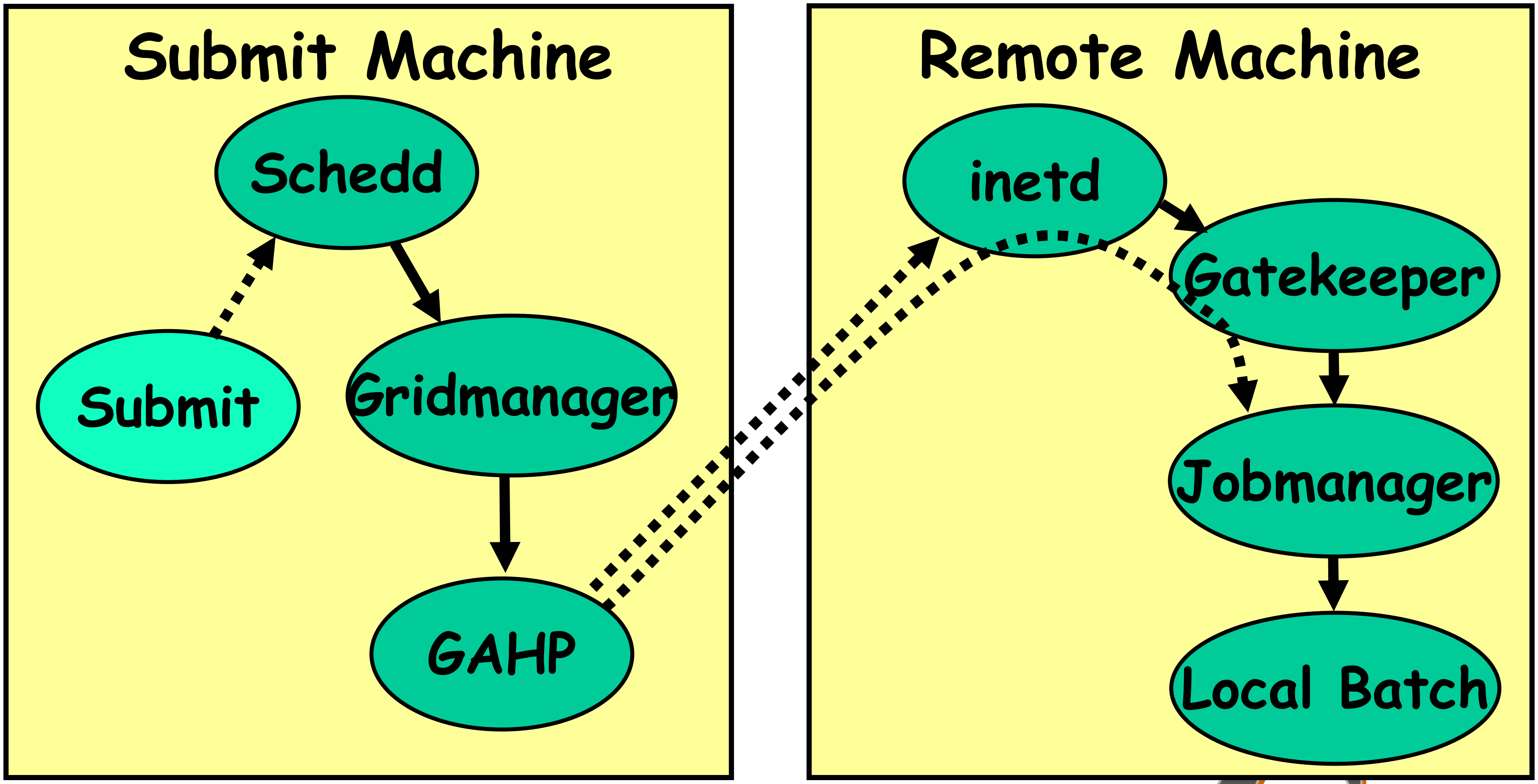
Condor-G: Gridmanager

- **condor_gridmanager** replaces **condor_shadow**.
  - One per user (not per job)
  - Runs as user (not root or condor)
    - Logs must user writable
      - **GRIDMANGER_LOG=/tmp/GridmanagerLog.$(USERNAME)**

Condor-G: GAHP

- Grid ASCII Helper Protocol
- Interface to various systems
- Typically one GAHP per protocol
- Gridmanager will spawn one GAHP per protocol/grid type

Condor-G to Globus 2



Condor-G

- Condor-G should work out of the box
- Globus can push several limits, consider increasing:
  - /proc/sys/fs/file-max
  - /proc/sys/net/ipv4/ip_local_port_range
  - Per process file descriptor limits

http://www.cs.wisc.edu/condor/condorg/linux_scalability.html

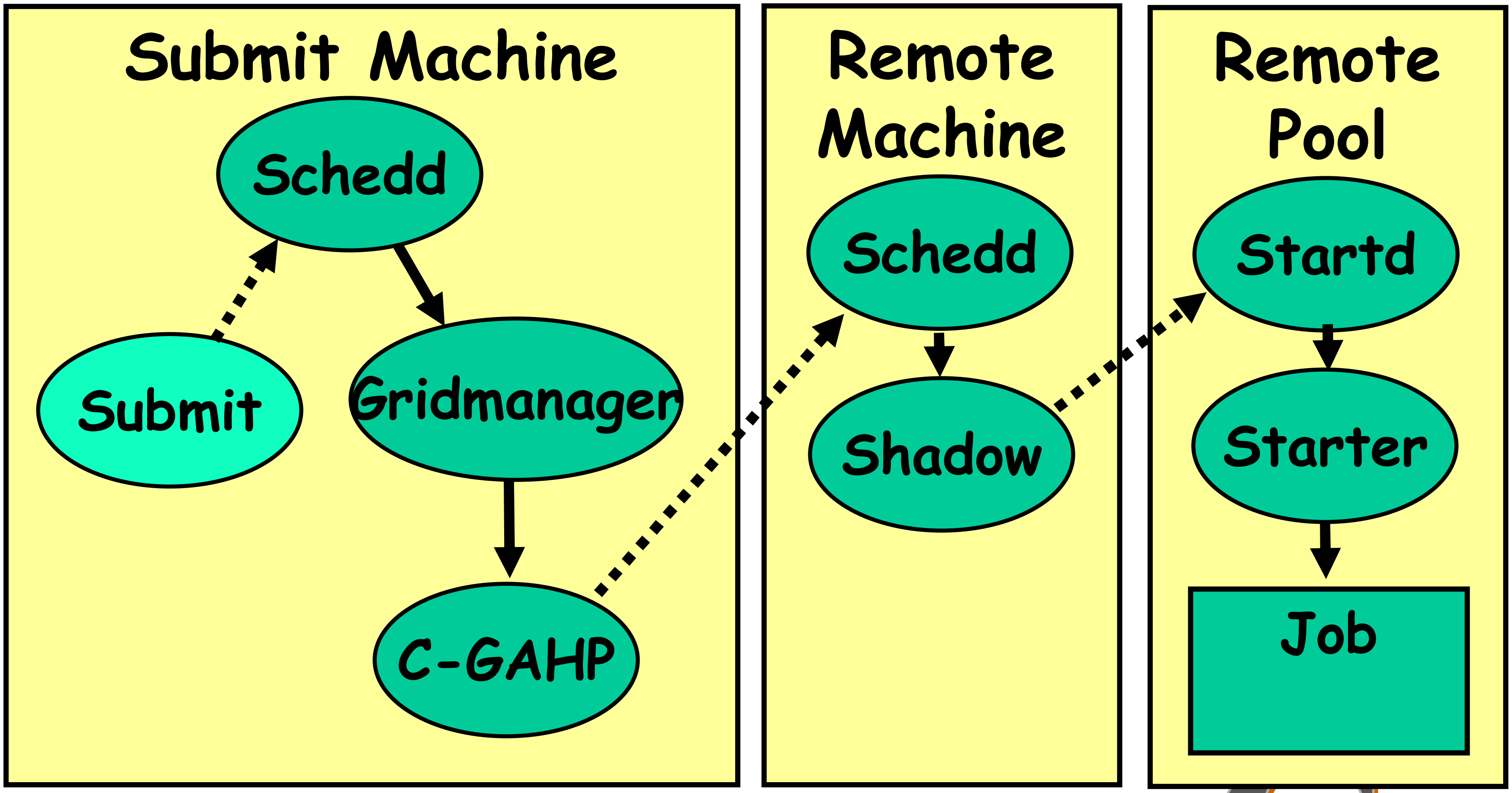
Condor-C

- Condor-G, but remote side is Condor
- Schedd to schedd communications
- Job appears in both queues
- Condor 6.7.3 and later

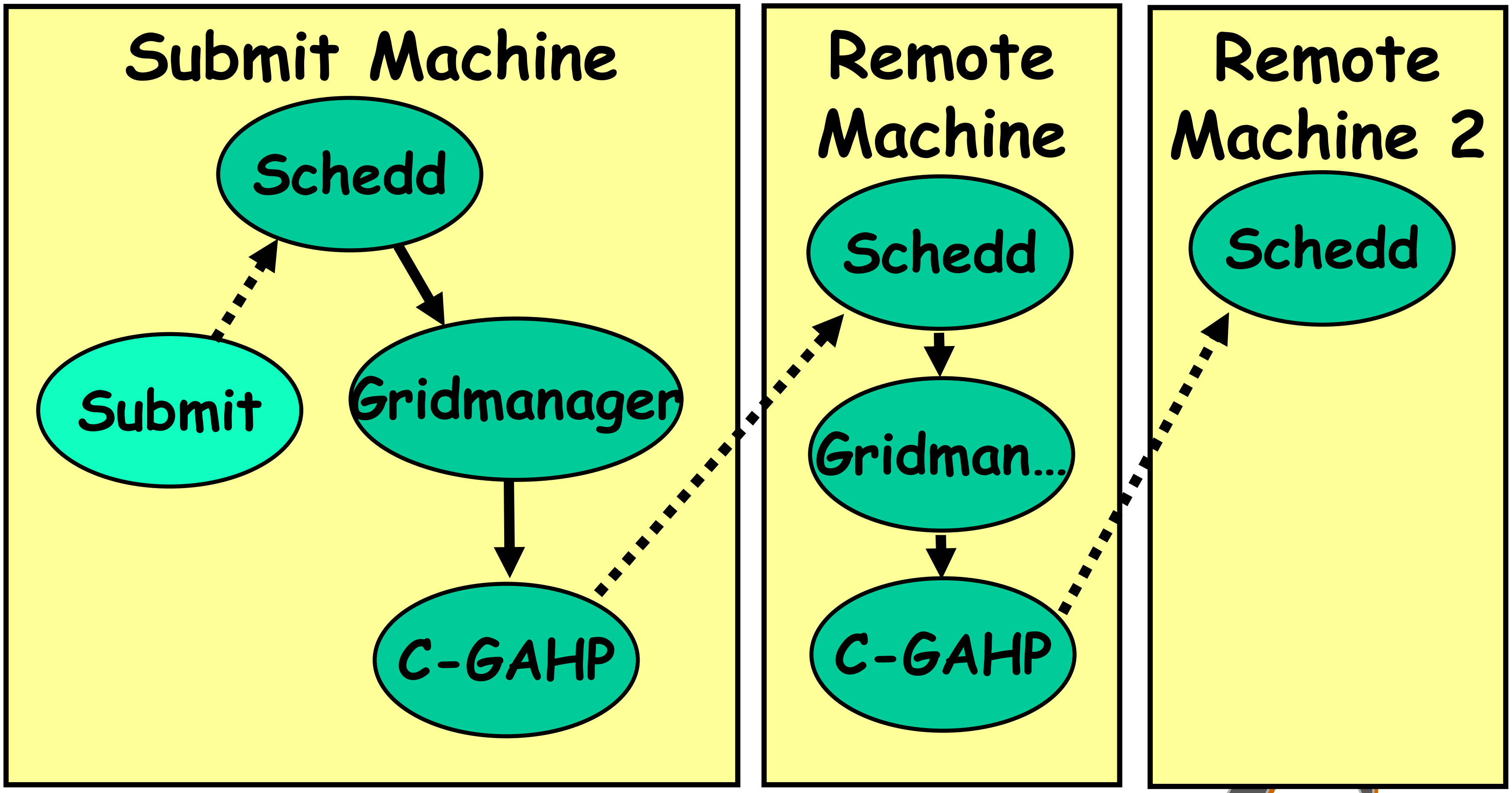
Condor-C

- Can chain
- Condor-C to Condor
- Condor-C to Condor-G to Globus
- Condor-C to Condor-C to Condor-C …

Condor-C


Submit Machine
Schedd
Submit
Gridmanager
C-GAHP
Remote Machine
Schedd
Shadow
Remote Pool
Startd
Starter
Job

Condor-C


Submit Machine
Schedd
Submit
Gridmanager
C-GAHP
Remote Machine
Schedd
Gridman…
C-GAHP
Remote Machine 2
Schedd

Condor-C Configuration

- Basically Just Works
- User requires permissions
- User on Submit machine requires:
  - Read access to Remote collector
  - Read and write access to Remote schedd

Condor-C Limitations

- Still under development
- Limited security support: CLAIMTOBE only
- Remote Schedd can not run as root or condor
- Various limits on remote universes supported

Condor-C's Near Future

- GSI authentication
- Root/condor schedd
- Usability refinements

Condor-C Caveats

- Under development
- Tricky to specify jobs
  - condor-admin@cs.wisc.edu

Other Sources

- Condor Manual
- Condor Web Site
- condor-users mailing list

  http://www.cs.wisc.edu/condor/mail-lists/
- condor-admin@cs.wisc.edu

Publications

- "Condor - A Distributed Job Scheduler," *Beowulf Cluster Computing with Linux*, MIT Press, 2002
- "Condor and the Grid," *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, 2003
- These chapters and other publications available online at our web site

Thank you!

http://www.cs.wisc.edu/condor

condor-admin@cs.wisc.edu