Building a Campus Grid

Mats Rynge – rynge@renci.org

Renaissance Computing Institute

University of North Carolina, Chapel Hill

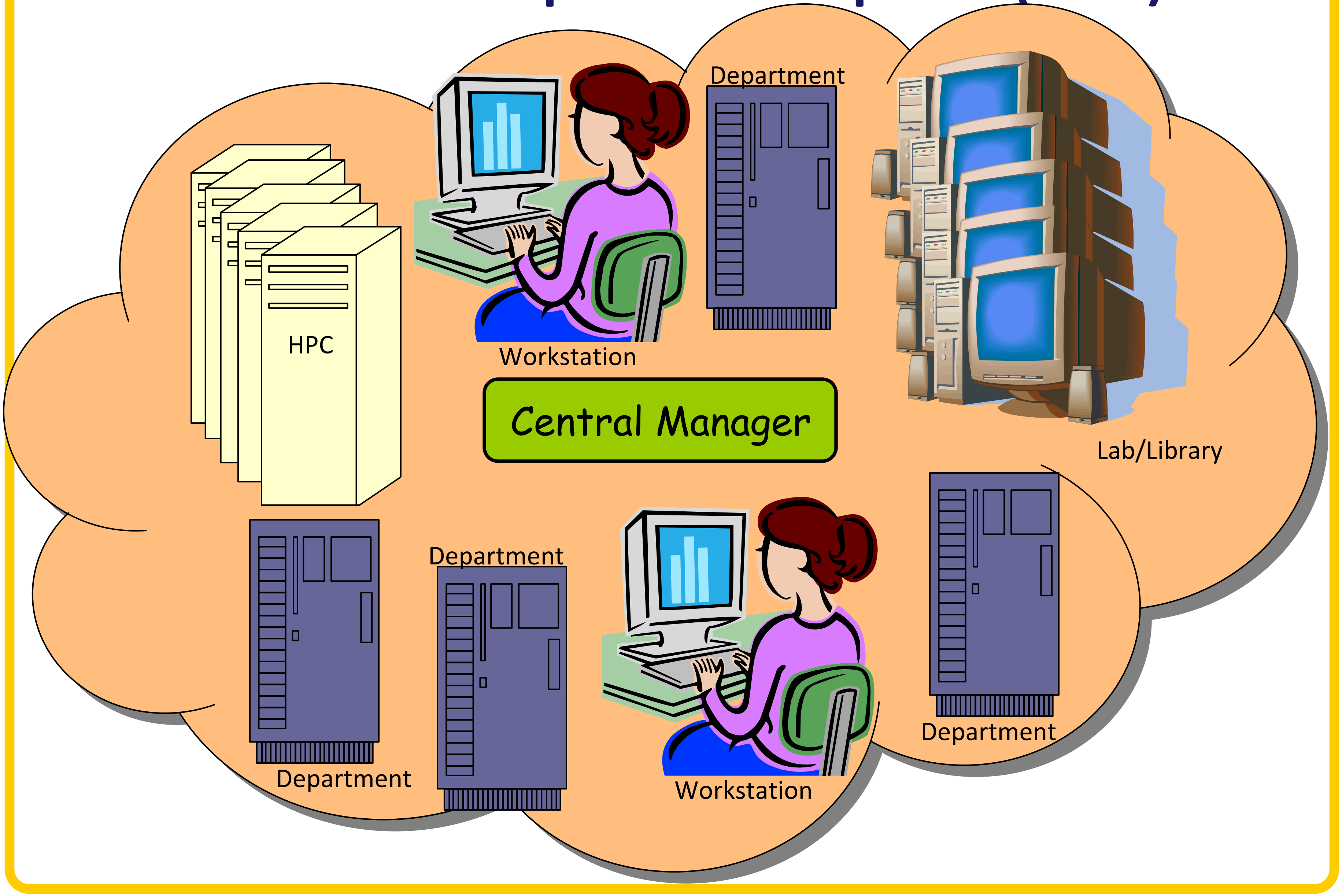
Outline

- Recipe
- Condor Daemons
- ClassAds
- Configuration Files
- Configuration Examples
  - Creating a central manager
  - Joining a dedicated processing machine
  - Joining a interactive workstation
- Troubleshooting
- Security

Why a Campus Grid?

- Existing, under-utilized, resources all over campus!
- Enable departments and researchers on campus to share their *existing* computers
- The same departments and researchers will have access to a larger pool of resources when they need to do large computations
- Community driven project
  - Enabled by Central IT
- Corner stone to join national cyber infrastructure

What is a campus Condor pool? (cont)



Recipe

- Central IT will provide central services (collector, negotiator, one submit node, …)
- Campus departments / researches can join their existing compute resources to the pool, and share with the community
- Resource owners have full control over the policy for their resources

Recipe - Required Tools

- Condor
- A central IT champion
  - Configure and maintain the central services
  - Outreach to departments and users (remember, this is about distributed resources!)
  - Documentation and support

Condor Daemons

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

Condor Daemons

- `condor_master` - controls everything else
- `condor_startd` - executing jobs
- `condor_schedd` - submitting jobs
- `condor_collector` - Collects system information; only on Central Manager
- `condor_negotiator` - Assigns jobs to machines; only on Central Manager

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_startd

- Represents a machine willing to run jobs to the Condor pool
- Run on any machine you want to run jobs on
- Enforces the wishes of the machine owner (the owner’s “policy”)

condor_schedd

- Represents jobs to the Condor pool
- Maintains persistent queue of jobs
  - Queue is not strictly first-in-first-out (priority based)
  - Each machine running **condor_schedd** maintains its own independent queue
- Run on any machine you want to submit jobs from

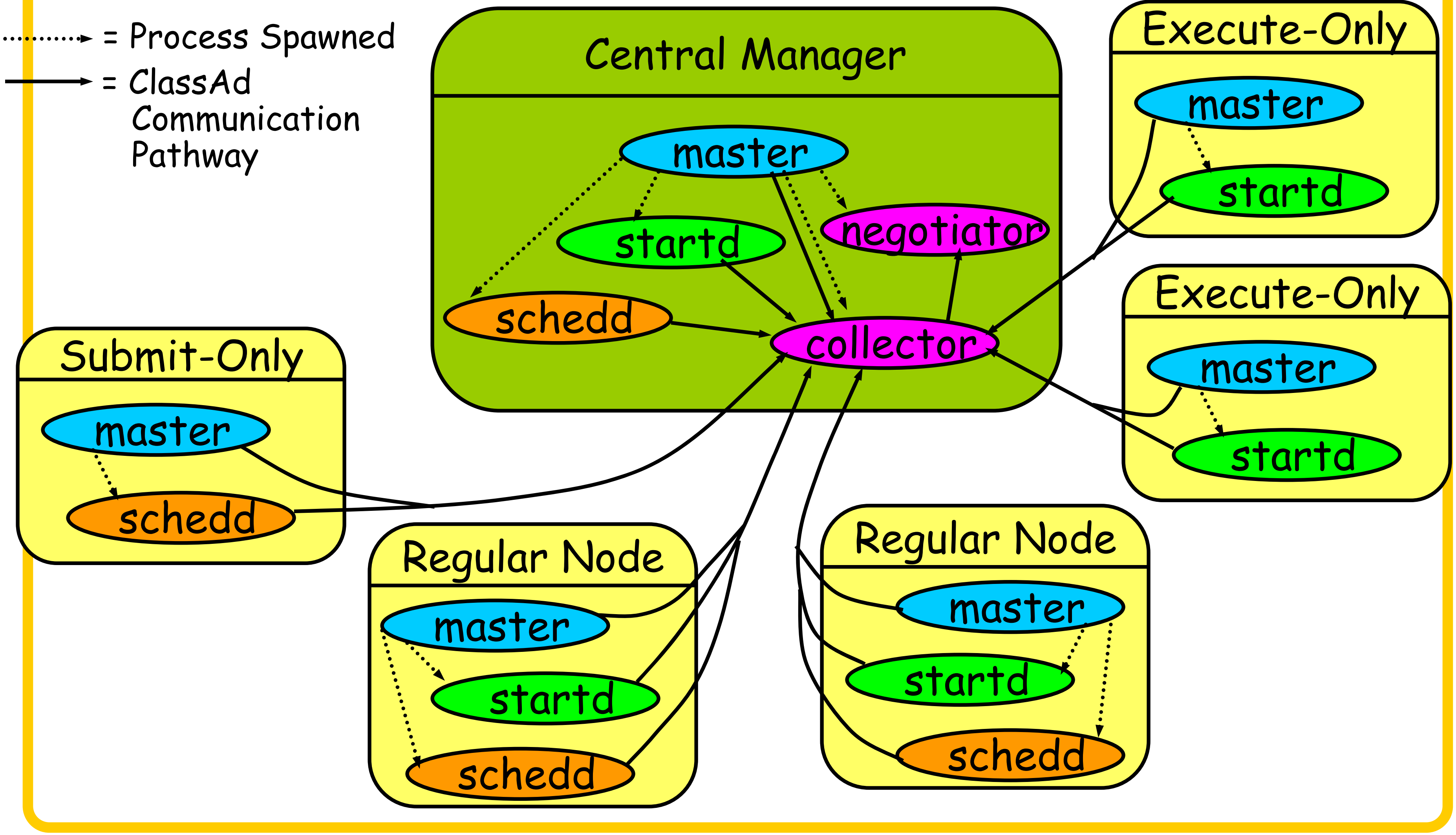
condor_collector

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

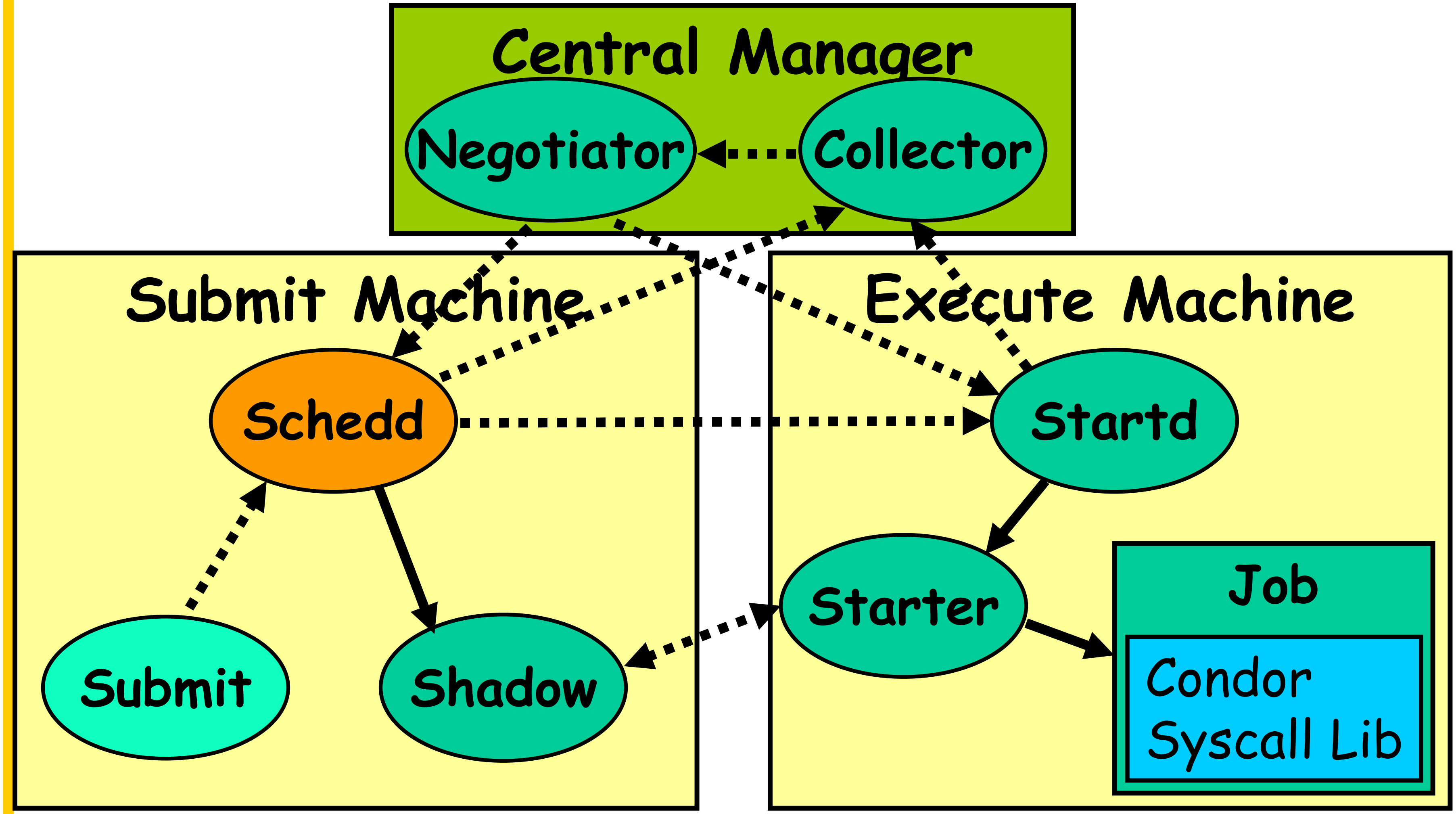
`condor_negotiator`

- 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 ranks and priorities

Typical Condor Pool



Job Startup


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

ClassAds – the basic building block of Condor

- Set of key-value pairs
- Values can be expressions
- Can be matched against each other
  - Requirements and Rank
    - MY.name – Looks for “name” in local ClassAd
    - TARGET.name – Looks for “name” in the other ClassAd
    - Name – Looks for “name” in the local ClassAd, then the other ClassAd

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)

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 =?=

Configuration Files

- Multiple files concatenated
  - Later definitions overwrite previous ones
- Order of files:
  - Global configuration file (only required file)
  - Local and shared configuration files

Global Configuration File

- Found either in file pointed to with the **`CONDOR_CONFIG`** environment variable, **`/etc/condor/condor_config`**, or **`~condor/condor_config`**
- All settings can be in this file
- “Global” on assumption it’s shared between machines. NFS, automated copies, etc.

Local Configuration File

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

Local Configuration File

- Can be on local disk of each machine

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

- Can be in a shared directory
  – Use **`$(HOSTNAME)`** which expands to the machine's name

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

Policy

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

Policy Expressions

- Specified in **`condor_config`**
  - Ends up startd/machine ClassAd
- Policy evaluates both a machine ClassAd and a job ClassAd together
  - Policy can reference items in either ClassAd (See manual for list)

Machine (Startd) Policy Expression Summary

- **START** – When is this machine willing to start a job
- **RANK** - Job preferences
- **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

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

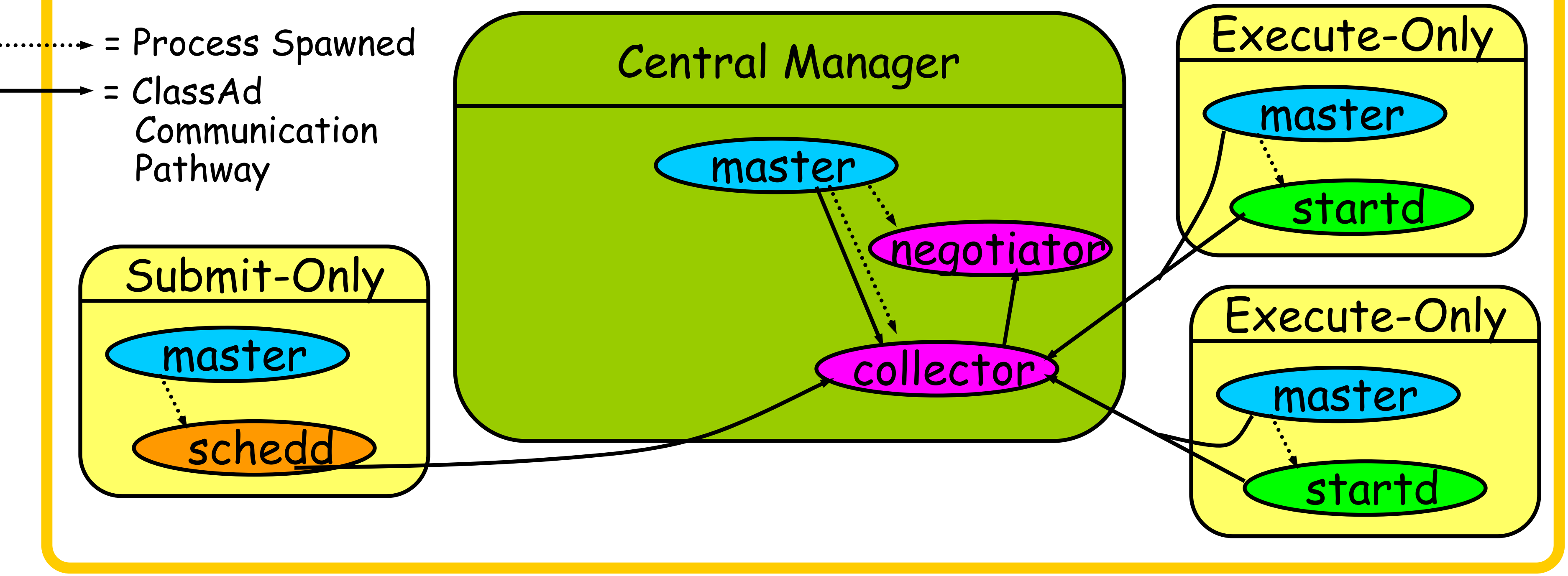
- Often used to give priority to owner of a particular group of machines
- Claimed machines still advertise looking for higher ranked job to preempt the current job

Configuration Case #1

Central/department IT setting up a
central manager

Needed Daemons

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



Security

```
HOST_ALLOW_READ = *.unc.edu, 152.54.0.0/20
HOST_ALLOW_WRITE = *.unc.edu, 152.54.0.0/20
HOSTDENY_WRITE = *.wireless.unc.edu
```

Limit the number of allowed submit nodes (more on the security later)

```
HOSTALLOW_ADVERTISE_SCHEDD =
  tarheelgrid.isis.unc.edu, fire.renci.org, …
```

Configuration Case #2

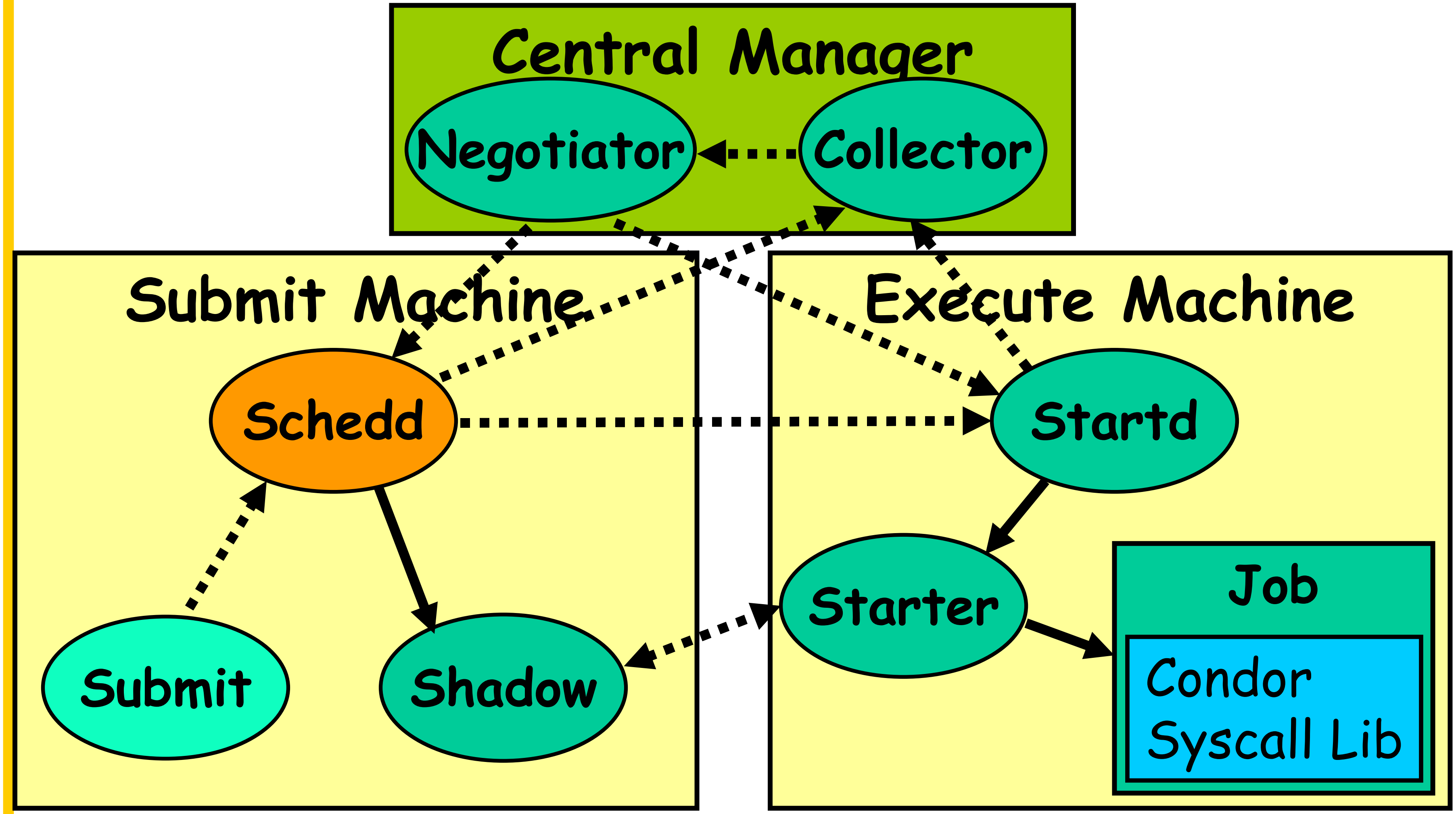
Department or lab wanting to join their existing dedicated processing machine to campus pool

Case #2 - Configuration

```
CONDOR_HOST = cm1.isis.unc.edu

DAEMON_LIST = MASTER, STARTD

HOST_ALLOW_READ = *.unc.edu, 152.54.0.0/20
HOST_ALLOW_WRITE = *.unc.edu, 152.54.0.0/20

HIGHPORT = 12000
LOWPORT = 10000
```

Note that the firewall needs to be open, for the specific port range, against all hosts specified in HOST_ALLOW_READ/HOST_ALLOW_WRITE

Job Startup


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

Case #2 – Policy

Prefer Chemistry jobs

```
START = True
RANK  = Department =!= UNDEFINED &&
  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
```

Use Case #3

Department or lab wanting to join a interactive workstation

Case #3 - Configuration

```
CONDOR_HOST = cm1.isis.unc.edu

DAEMON_LIST = MASTER, STARTD

HOST_ALLOW_READ = *.unc.edu, 152.54.0.0/20
HOST_ALLOW_WRITE = *.unc.edu, 152.54.0.0/20

HIGHPORT = 12000
LOWPORT = 10000
```

Note that the firewall needs to be open, for the specific port range, against all hosts specified in HOST_ALLOW_READ/HOST_ALLOW_WRITE

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

Useful Attributes

- `LoadAvg`
  - Current load average
- `CondorLoadAvg`
  - Current load average generated by Condor
- `KeyboardIdle`
  - Seconds since last keyboard or mouse activity

Useful Attributes

- `CurrentTime`
  - Current time, in Unix epoch time (seconds since midnight Jan 1, 1970)
- `EnteredCurrentActivity`
  - When did Condor enter the current activity, in Unix epoch time

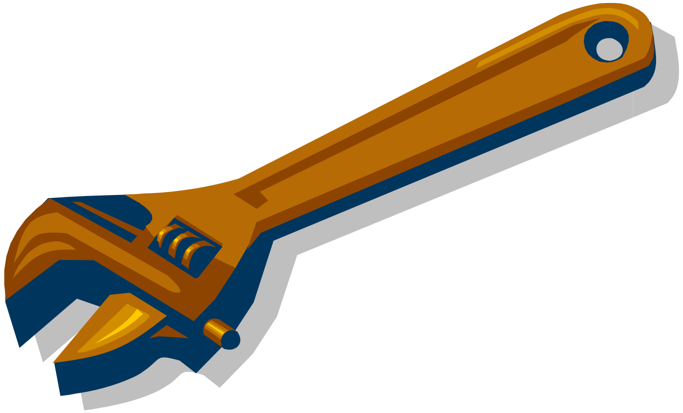
Macros in Configuration Files

```
NonCondorLoadAvg = (LoadAvg - CondorLoadAvg)
BgndLoad = 0.3
CPU_Busy = ($(NonCondorLoadAvg) >= $(BgndLoad))
CPU_Idle = ($(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
```

Command Line Tools



condor_config_val

- Find current configuration values

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

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_config_val -config

- What configuration files are being used?

```
% condor_config_val –config
Config source:
        /var/home/condor/condor_config
Local config sources:
        /unsup/condor/etc/condor_config.hosts
        /unsup/condor/etc/condor_config.global
        /unsup/condor/etc/condor_config.policy
        /unsup/condor-test/etc/hosts/puffin.local
```

Querying daemons `condor_status`

- Queries the collector for information about daemons in your pool
- Defaults to finding **`condor_startd`**s
- **`condor_status –schedd`** summarizes all job queues
- **`condor_status –master`** returns list of all **`condor_master`**s

condor_status

- **-long** displays the full ClassAd
- Optionally specify 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

“slanting” by Stefano Mortellaro (“fazen”) © 2005
Licensed under the Creative Commons Attribution 2.0 license
http://www.flickr.com/photos/fazen/17200735/ http://www.webcitation.org/5XIhNWC7Y

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
```

Examining 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 –better-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
```

Debugging Jobs: condor_q

- Examine the job with **`condor_q`**
  - especially **`-long`** and **`–analyze`**
  - Compare with **`condor_status –long`** for a machine you expected to match

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

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 relevant entries

Security

Security Threats

- Condor System
  - Authentication / Authorization
    - Next couple of slides
- Using Condor as a vehicle for attacks
  - Hard to prevent
  - Local exploits (privilege escalation)
    - Condor jobs are not sandboxed
  - Another example: Distributed DoS

Security

- Authentication
  - Users
    - Done on the submit machine, using OS authentication mechanisms
  - Machines
    - IP based
    - Machines on the wireless subnet are not allowed to join the pool
    - Other subnets to block?

Security - Users

- Based on the local UID
- Example: cdpoon@tarheelgrid.isis.unc.edu
- Note that these are not Campus wide identifiers. The components of the identifier is local username @ hostname of the submit node.
- Another example: engage@belhaven-1.renci.org
- These are getting logged during the negotiation cycle on the central manager
- Authorization can be done at the central manager, or in the policy on the execution machine

Security – Machines

- Anybody on campus can join a machine for job execution
- Only a set of machines will be allowed for job submits – this will ensure we have an audit trail
  - The allowed set is configured on the central manager

Security Links

- Condor documentation on security: http://www.cs.wisc.edu/condor/manual/v7.2/3_6Security.html

- Building a secure Condor pool in an open academic environment http://www.allhands.org.uk/2005/proceedings/papers/435.pdf