Condor Philosophy

Greg Thain

Agenda

The other talks are about the **hows** of HTCondor

This talk is about the **why**

First Principles: Who

- 1) Owner: $$$ (€€€,   £££ ???)

- 2) Job Submitter

- 3) Administrator

The Philosophy on 1 slide

To ***reliably*** run ***as much work*** as possible

on ***as many machines*** as possible

(in order of precedence)

The other side – administrator’s view

To ***maximize*** machine ***utilization***

ABCs:

Always

Be

Computing

“No Cycle Left Behind”

The Unstated Assumption

"Work" can be broken up into smaller jobs

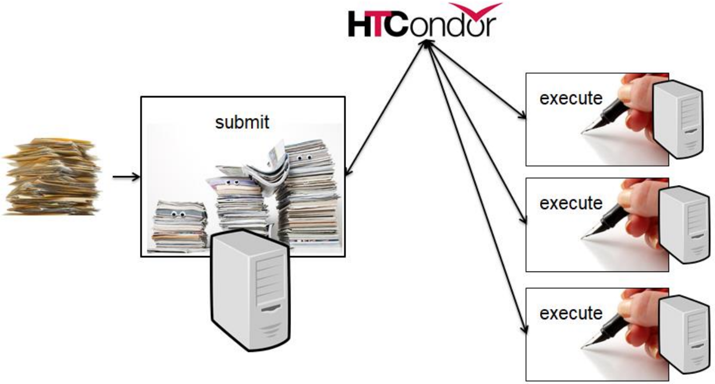
Smaller the better (up to a point)

files as ipc

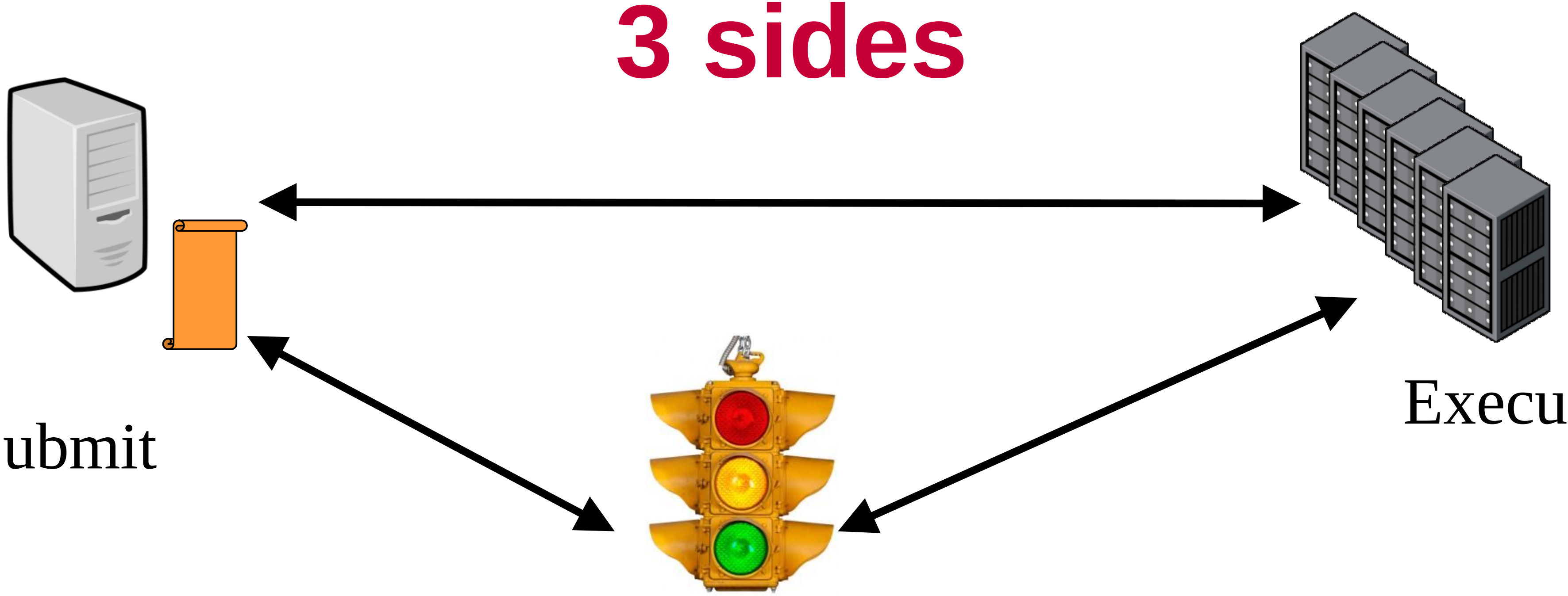
dependencies via dag

Optimize time-to-finish

not time-to-run



Overview of condor: 3 sides



Submit

Execute

Central Manager

To reliably run…

- Reliability 1st priority

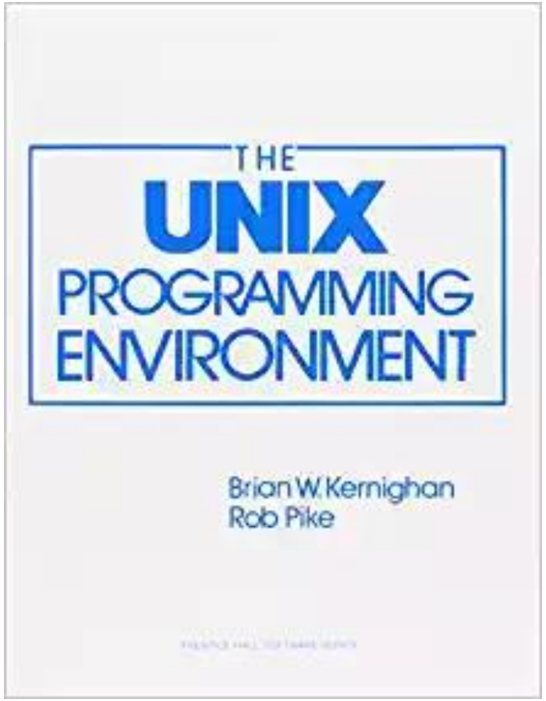
- We can make HTCondor ***fast enough***

  w/o sacrificing any reliability – no screw polishing

To reliably run…

- Unix process per daemon
- Each has failure semantics
- Each cleans up on exit

- Each has responsibility
  - Perhaps many per machine



THE
UNIX
PROGRAMMING
ENVIRONMENT

Brian W. Kernighan
Rob Pike

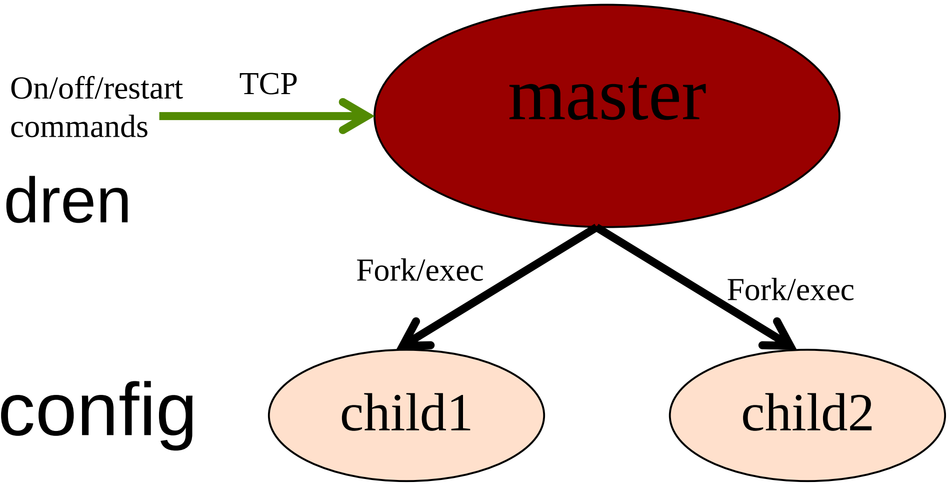
PRENTICE HALL SOFTWARE SERIES

To reliably run… requires parent

Small condor_master runs on all condor machines

Responsibilities:

- Like ~~systemd~~ init,
  - starts, restarts, kills children
- condor_on,
- condor_off, condor_reconfig
- Detects hung kids and kills them(!)
- Exits if disk full
- Runs Linux kernel tuning script



master *manages* process

Manage:

- Remove what you create
  - and what they created…
- Measure what you create
  - And report it
- Limit what you create

… as many jobs…

Requires a scheduler, the condor_schedd

Users submit jobs to schedd

Schedd is a database

reliable, slow

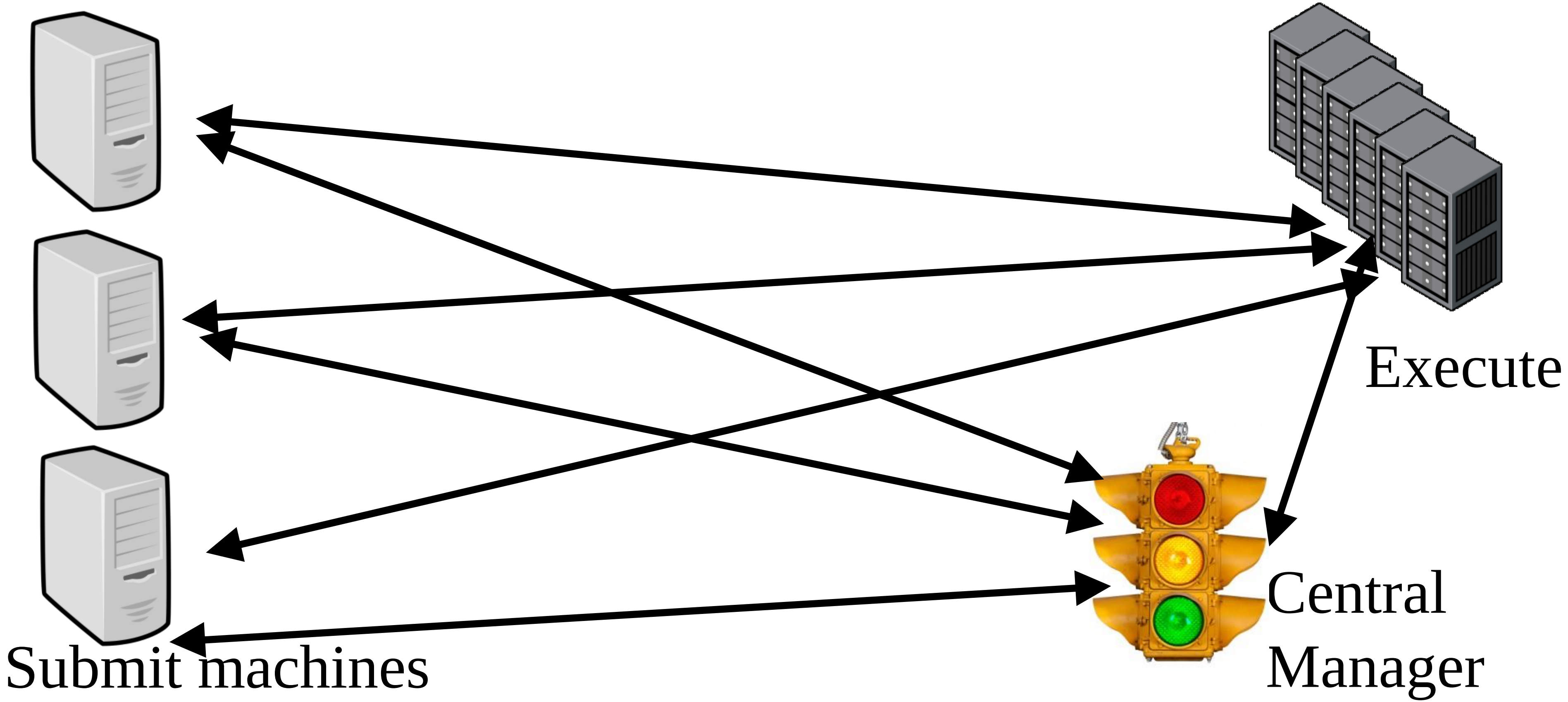
On crash, all restart

To support many jobs,

reliably means…

VAULT DOOR
WEIGHT 22 1/2 Tons
THICKNESS 22 Inches
STEEL 11 Layers of Special Cutting and Drill Resistant
LOCKS 4 Hamilton Watch Movements for Time Locks

Scaling via many submit points



Scaling via many submit points

Adding submit points just helps scaling

Allows submit near the user

“Submit locally, run globally”

But the schedd doesn’t schedule

- It does a little
- Schedd has jobs, can request machines
- But only uses the machines given to it

- Scheduling, not planning

The shadow manage running, remote jobs

- One process per running job on submit
- Responsible for job’s policy remotely
  - Tells the worker node what to do
- Expensive? Yes – worth it

…on as many machines

Implies machines are heterogeneous

Could be foreign pools

Could be same pool with different config

Could be places without shared filesystem

Two-faced nature of HTCondor

Split responsibility:

- Worker side
- Submit side

We ***encourage*** different config on both sides

Always focusing on responsibility of the side

- Always consider where responsibility goes

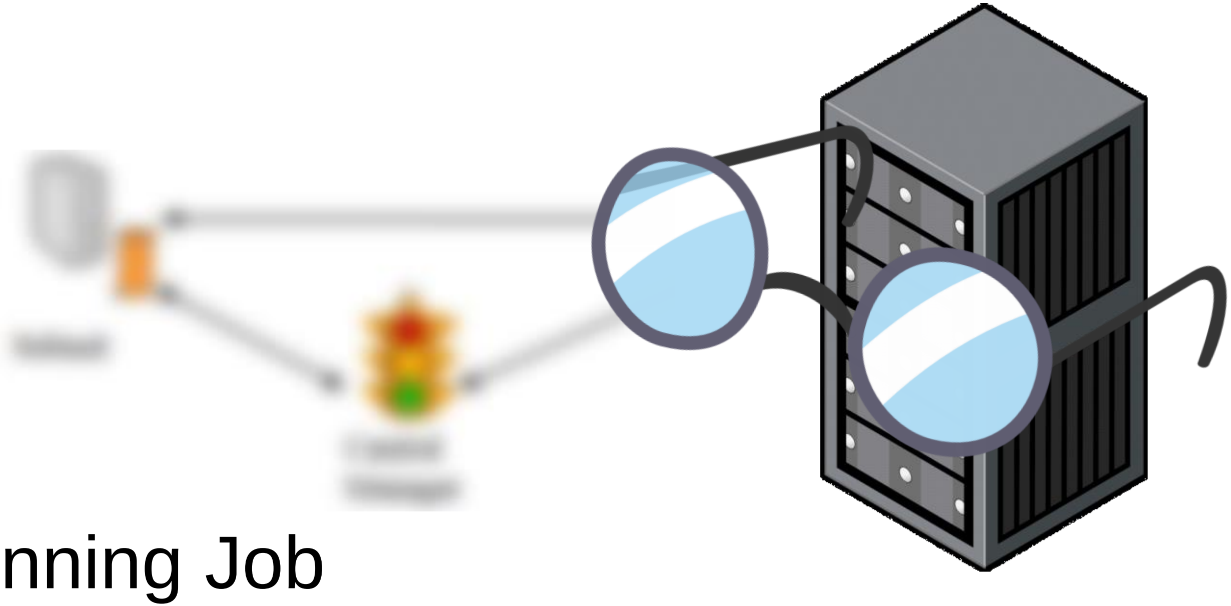
The startd

- Startd represents the policy of the machine
- Creates “slots”, places for jobs to run
- Could conflict with job’s policy?
  - Who wins?

- Always the machine – the job is a guest

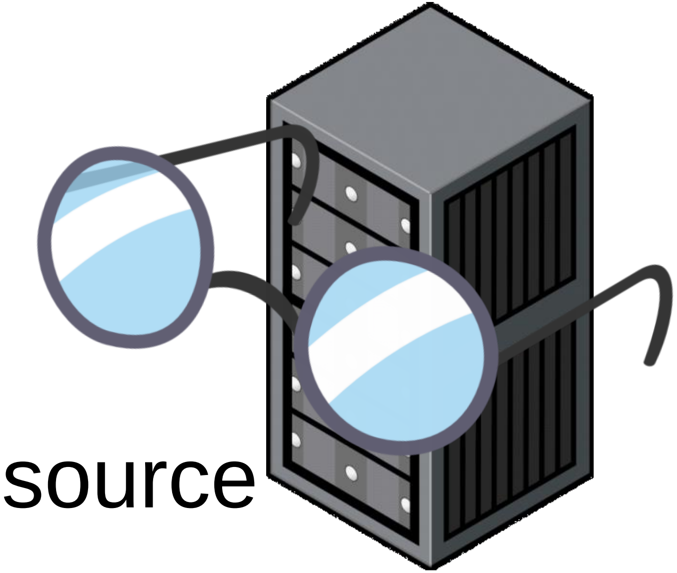
Startd Mission Statement

- Near sighted
- 3 inputs only:
  - Machine
  - Running Jobs
  - Candidate Running Job
- Knows nothing about the rest of the system!



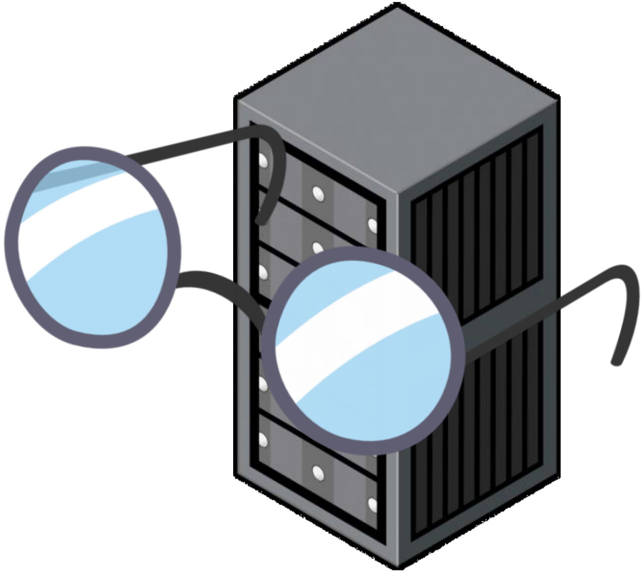
Things the startd can do

- Only run some kinds of jobs
- Preempt one job for another
- Only run 1 job of some type
- Expose and match custom resource



But the startd doesn't run job

- Doesn't run jobs directly,
- Creates (and manages!) child process, the starter



The Starter

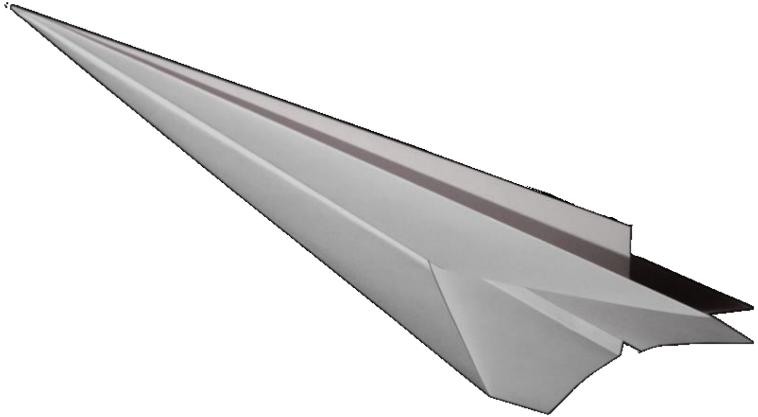
- Startd manages *machine*, starter *job*
- When job starts, startd spawns starter
- One starter per job, thus one per slot

Starter Responsibilities

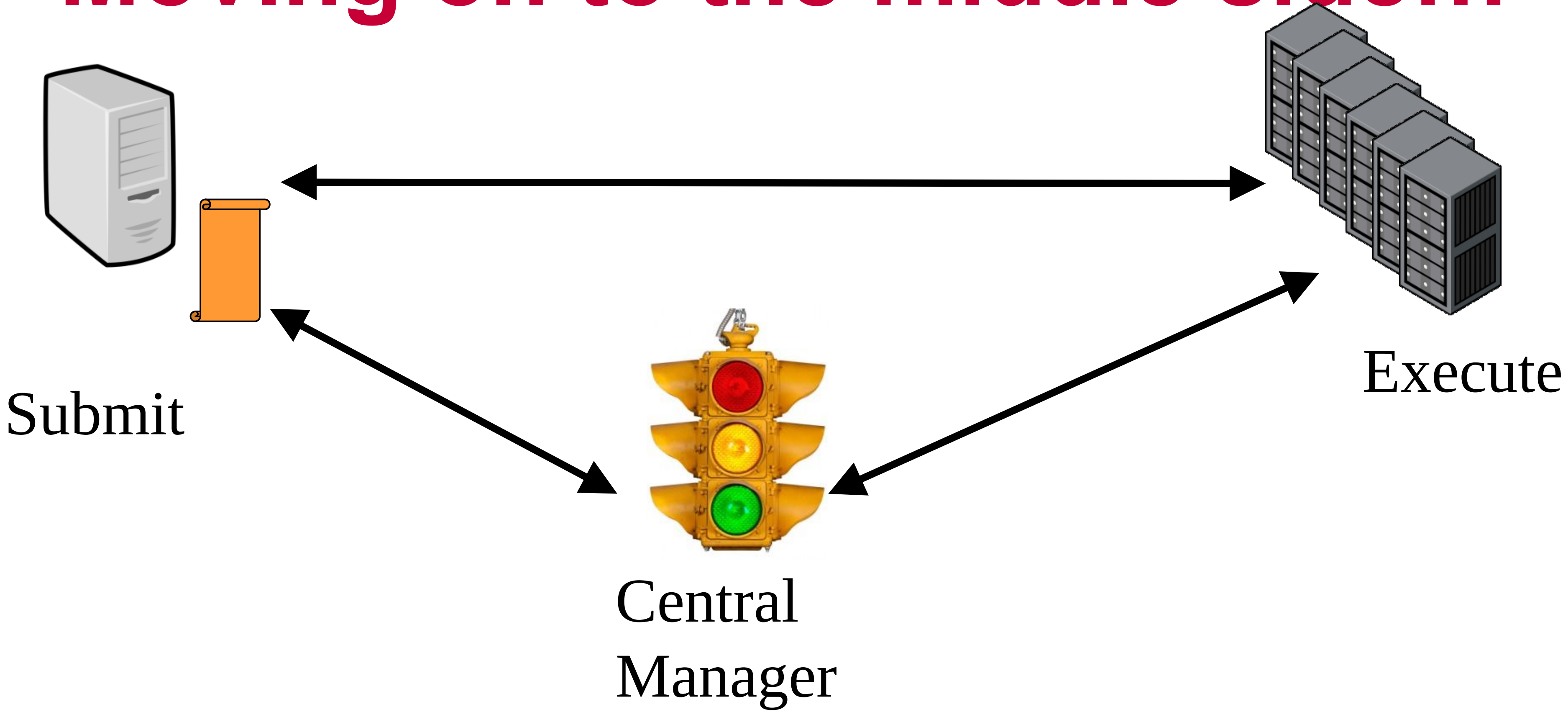
- Starter manages running job on machine:
- Create environment for job
- Monitor, report job resource usage home
- Creates “Universe” metaphor
- Clean up after job
  - Condor Philosophy: renters clean up after use
    - (Startd cleans up after starter…)
- File Transfer

A few words on file transfer…

- We can use shared FS or File Transfer
- Prefer File Transfer:
  - Managed
  - Portable
  - Declarative



Moving on to the middle side…



The Central Manager

- Part 1: The Collector
  - The central database
  - All in memory, lightweight
  - Every thing reports to collector
    - Everything is a classad
  - condor_status queries

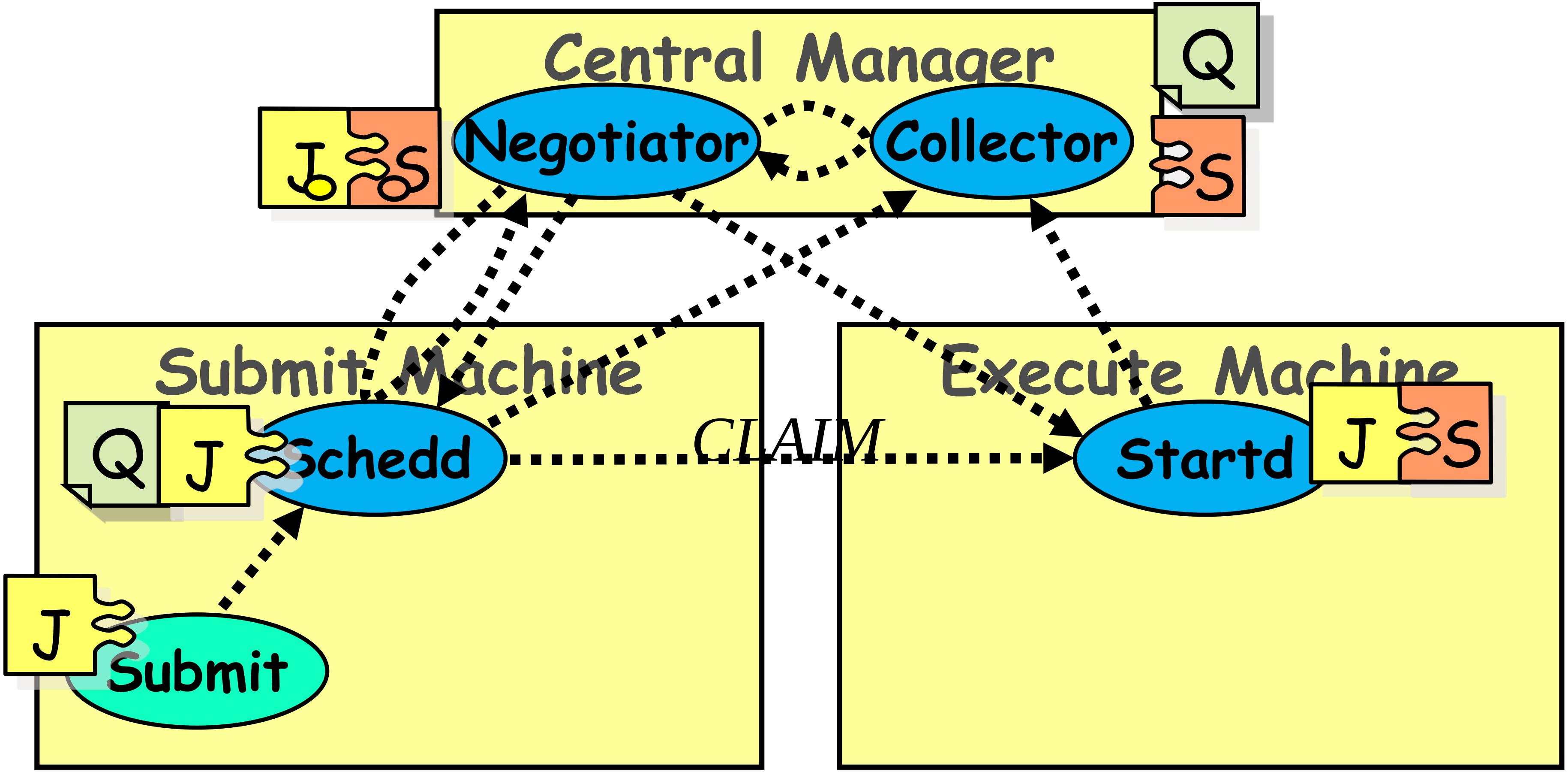
The Collector

- Looses everything when it crashes
- Protocol is always be updating
- Not a central point of failure
- Garbage collects if no updates

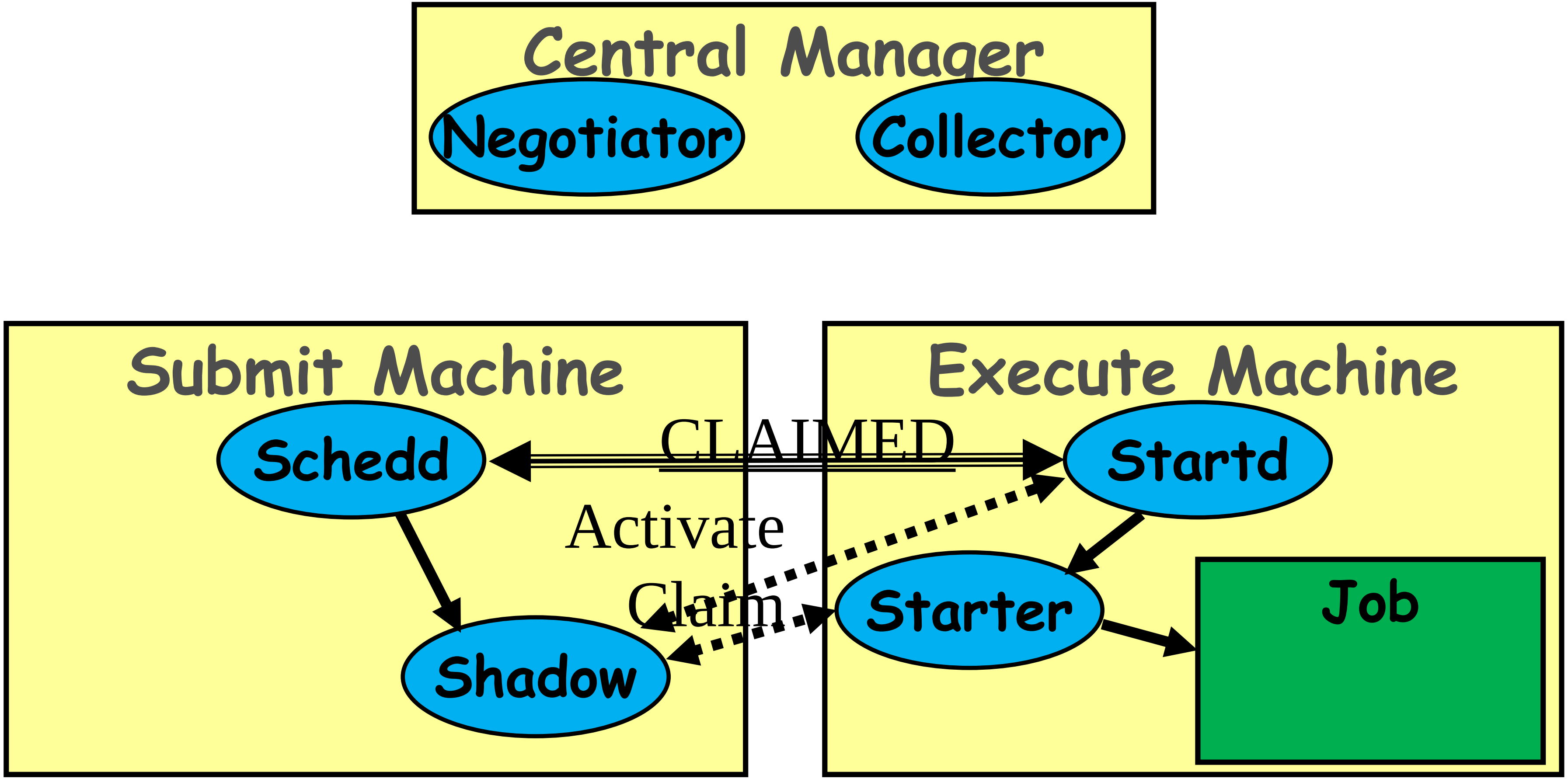
The Negotiator

- Other “half” of scheduling
- Slow, allocates machines to user
  - Two phase scheduling:
    - Slow, negotiator rebalancing
    - Fast, schedd scheduling and reusing of claims
- Not a single point of failure

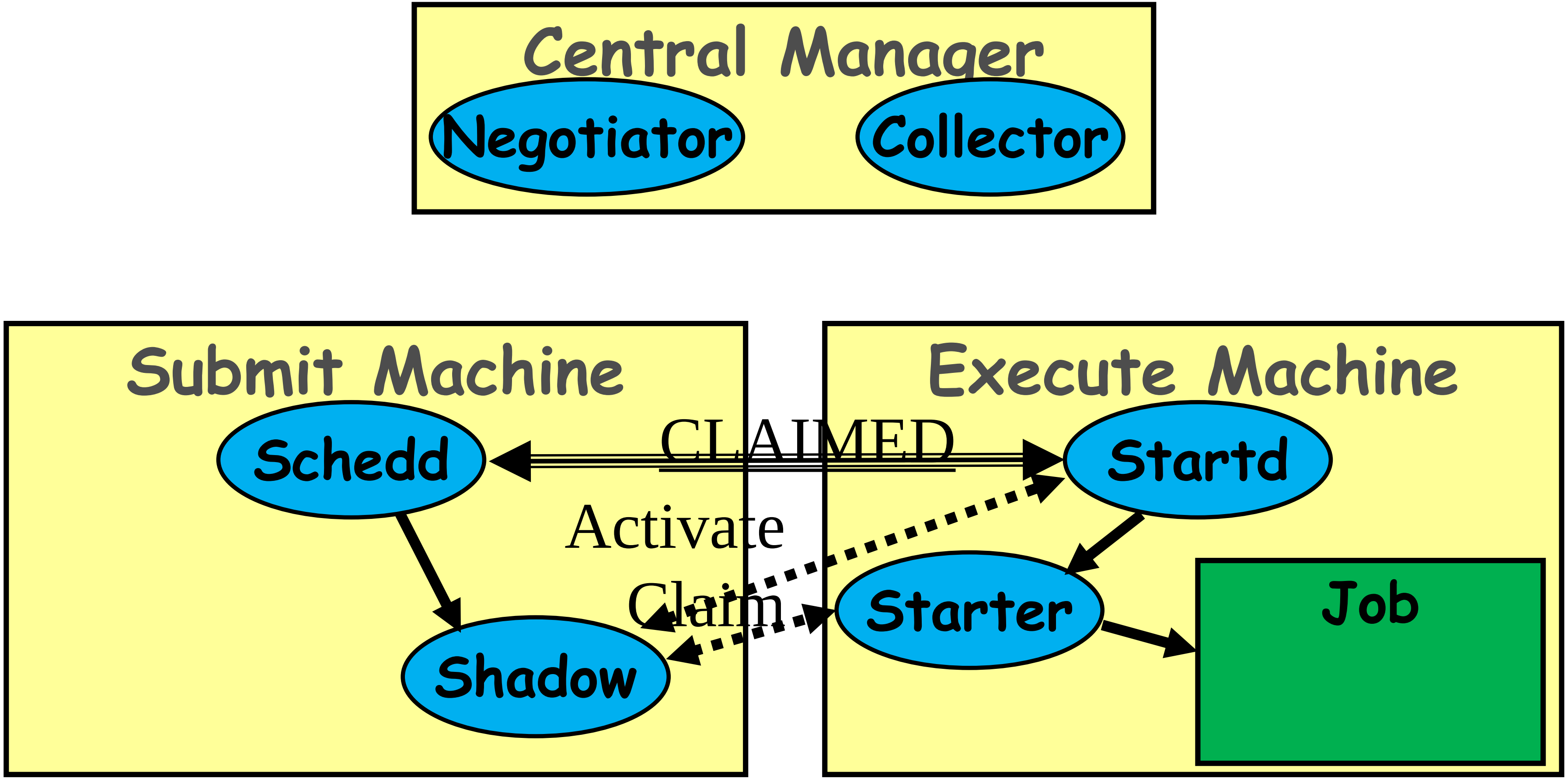
Claiming Protocol



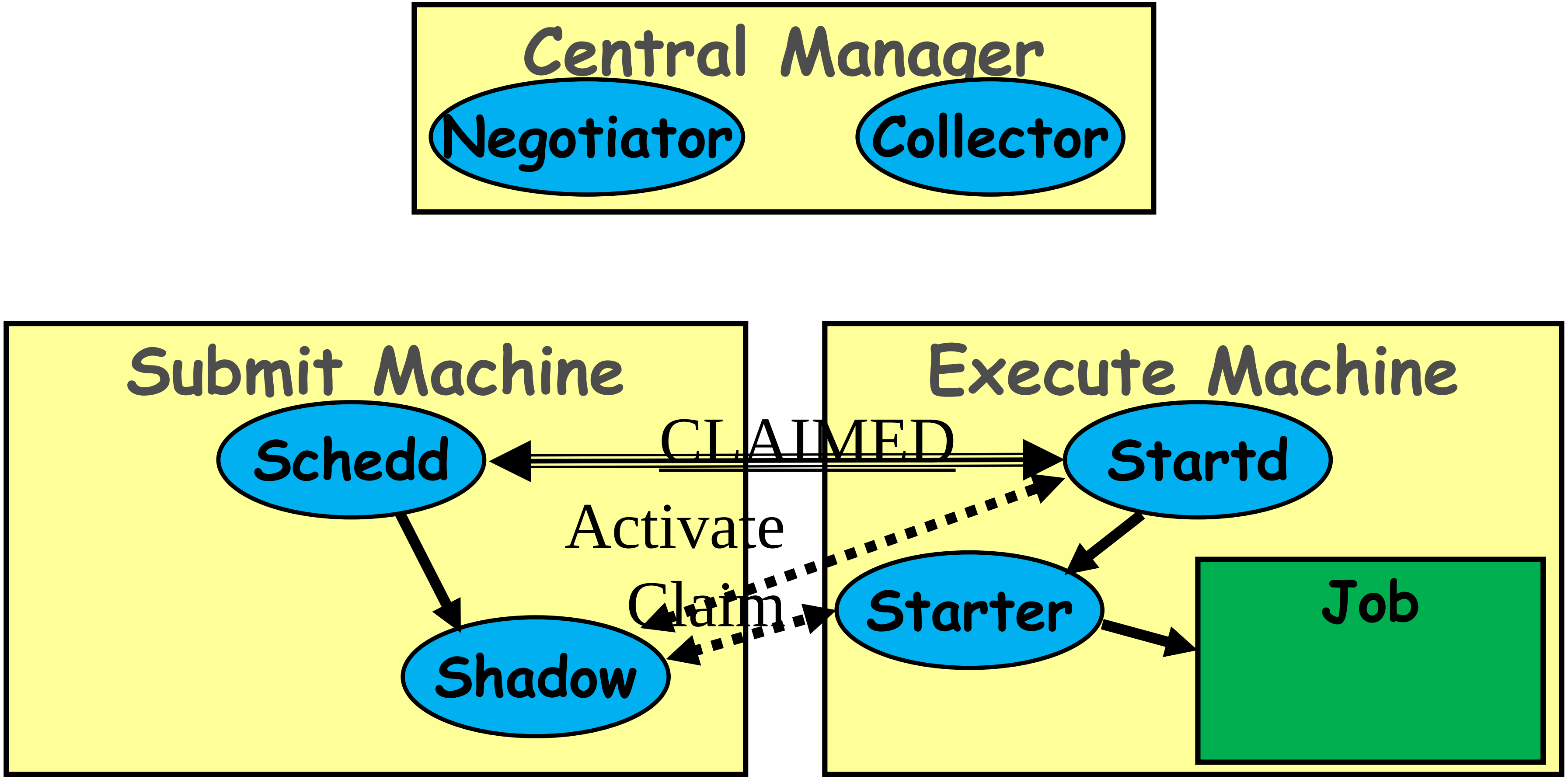
Claim Activation



Repeat until Claim released



Repeat until Claim released



When is claim released?

- When relinquished by one of the following
  - lease on the claim is not renewed
    - Why? Machine powered off, disappeared, etc
  - schedd
    - Why? Out of jobs, shutting down, schedd didn’t “like” the machine, etc
  - startd
    - Why? Policy re CLAIM_WORKLIFE, prefers a different match (via Rank), non-dedicated desktop, etc
  - negotiator
    - Why? User priority inversion policy
  - explicitly via a command-line tool
    - E.g. condor_vacate

Architecture items to note

- Machines (startds) or submitters (schedds) can dynamically appear and disappear
  - Key for expanding a pool into clouds or grids
  - Key for backfilling HPC resources
- Scheduling policy can be flexible and very distributed
- CM makes a match, then gets out of the way
- Distributed policy enables federation across administrative domains
  - Lots of network arrows on previous slides
  - Reflects the P2P nature of HTCondor

Quiz Time

- How to hold job that runs > 24 hours
  - Or rather, where?
- On the submit machine?
- Or Execute Machine?

Discuss!

Quiz Answer

- It depends!

    - Property of *job* or property of *machine*?

Conclusion

- Thank you, and let’s continue discussing…