HTCondor Architecture
HTCondor Week 2020

Todd Tannenbaum
Center for High Throughput Computing
Start with People
People have Problems
“Some of my jobs need a lot of memory, others a lot of cores”
“If an important group needs all the computers for three days to make a paper deadline, I’m ok with that”
HTCondor Manages These constraints
Not even that easy
In the real world, many users,
Many resource providers
This is a distributed problem.

Distributed because of *people*
Not because of machines.
Our goal is to satisfy all these constraints.
The Philosophy on 1 slide

To reliably run as much work as possible

on as many machines as possible

Subject to all constraints
The other side: administrator’s

To *maximize* machine *utilization*

*subject to constraints*

100 100 100 100 100 100 100 100

*High Throughput is also High Utilization Computing!*
computing
A always  |
B be  |
C computing
The Unstated Assumption

“Work” can be broken up into smaller jobs
Smaller the better (up to a point)
files as ipc
any interdependencies via DAGs
Optimize time-to-finish
not time-to-run
Overview of condor:
3 sides

Submit

Central Manager

Execute
We are going to fill in the boxes!

Central Manager

Submit Machine

Execute Machine
ClassAds: The *lingua franca* of HTCondor
What are ClassAds?

ClassAds is a language for objects (jobs and machines) to

- Express attributes about themselves
- Express what they require/desire in a “match” (similar to personal classified ads)

**Structure**: Set of attribute name/value pairs, where the value can be a literal or an expression. Semi-structured, no fixed schema.
ClassAd Values

› Literals
  • Strings ( “RedHat6” ), integers, floats, boolean (true/false), …

› Expressions
  • Similar look to C/C++ or Java : operators, references, functions
  • References: to other attributes in the same ad, or attributes in an ad that is a candidate for a match
  • Operators: +, -, *, /, <, <=,>, >=, ==, !=, &&, and || all work as expected
  • Built-in Functions: if/then/else, string manipulation, regular expression pattern matching, list operations, dates, randomization, math (ceil, floor, quantize,…), time functions, eval, …
Simple Example

**Job Ad**
- **Type** = "Job"
- **Requirements** =
  - HasMatlabLicense == True &&
  - Memory >= 1024
- **Rank** = kflops + 1000000 * Memory
- **Cmd** = "/bin/sleep"
- **Args** = "3600"
- **Owner** = "gthain"
- **NumJobStarts** = 8
- **KindOfJob** = "simulation"
- **Department** = "Math"

**Machine Ad**
- **Type** = "Machine"
- **Cpus** = 40
- **Memory** = 2048
- **Requirements** =
  - (Owner == "gthain") ||
  - (KindOfJob == "simulation")
- **Rank** = Department == "Math"
- **HasMatlabLicense** = true
- **MaxTries** = 4
- **kflops** = 41403
The Magic of Matchmaking

- Two ClassAds can be matched via special attributes: Requirements and Rank.
- Two ads match if both their Requirements expressions evaluate to True.
- Rank evaluates to a float where higher is preferred; specifies the which match is desired if several ads meet the Requirements.
- Scoping of attribute references when matching:
  - MY.name – Value for attribute “name” in local ClassAd
  - TARGET.name – Value for attribute “name” in match candidate ClassAd
  - Name – Looks for “name” in the local ClassAd, then the candidate ClassAd.
HTCondor has many types of ClassAds

- A "Job Ad" represents a job to Condor
- A "Machine Ad" represents a computing resource
- Others types of ads represent other instances of other services (daemons), users, accounting records.
Architecture & Job Startup
Quick Review of Daemons

condor_master: runs on all machine, always plus a condor_procd, condor_shared_port

condor_schedd: runs on submit machine

condor_startd: runs on execute machine

condor_negotiator, condor_collector: runs on central manager
Submit Machine Process View

condor_master
(pid: 1740)

fork/exec

condor_schedd

condor_procd
condor_shared_port

fork/exec

condor_shadow
condor_shadow
condor_shadow

Tools: condor_submit, condor_q, condor_rm, condor_hold, …
Execute Machine Process View

condor_master
(pid: 1740)

fork/exec

condor_startd

condor_procd
condor_shared_port

condor_starter
condor_starter
condor_starter

Job
Job
Job
Central Manager Process View

condor_master (pid: 1740)

- condor_collector (fork/exec)
- condor_negotiator

condor_procd
condor_shared_port
Claim Activation

Central Manager

Negotiator
Collector

Submit Machine

Schedd
Shadow

Execute Machine

Startd
Starter
Job

CLAIMED
Activate Claim
Repeat until Claim released

Central Manager
- Negotiator
- Collector

Submit Machine
- Schedd
- Shadow

Execute Machine
- Startd
- Starter

CLAIMED
Activate Claim
Job
Repeat until Claim released

Central Manager

Negotiator

Collector

Submit Machine

Schedd

Shadow

Execute Machine

Startd

Starter

Job

CLAIMED

Activate Claim
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 very flexible (custom attributes) and very distributed

› Central manager just makes a match, then gets out of the way

› Distributed policy enables federation of resources across different organizations (administrative domains)
  • Lots of network arrows on previous slides
  • Reflects the P2P nature of HTCondor
Layout of a General Condor Pool

- Process Spawned
- Communication Pathway

Central Manager:
- master
- negotiator
- collector

Submit-Only:
- master
- schedd

Execute-Only:
- master
- startd

Submit-Only:
- master
- schedd

Both:
- master
- startd
- schedd
Thank You!