Computation Institute

Elastic Colony Manager

Reactive Capacity Growth for Virtual Clusters

David Champion

Computation Institute, Enrico Fermi Institute

US ATLAS, UC3

University of Chicago

dgc@uchicago.edu

UC3

Argonne NATIONAL LABORATORY

Motivations

US ATLAS / UC Tier 3

- Availability/Growth:
  1. the Midwest Tier 2 team jointly manages UCT3 with local admins
  2. MWT2 resources are renewed, while UCT3 resources sunset with no guarantee to replace
  3. yet UCT3 work does not slow

US ATLAS / UC Tier 3

- Transition Point:
  1. UCT3 runs an older PBS variant job scheduler
  2. the PBS has some performance problems under load
  3. MWT2 team runs nearly all Condor, and is most experienced with Condor
  4. users community is interested in moving to Condor
  5. this makes a good transitional point for experimenting with new approaches to computation

Goals

1. Expand computational capacity while controlling cost

   a. Elasticity

      - » ability to expand our environment to meet demand
      - » ability to contract the environment when demand is low
      - » buying static compute (hardware) without 80-90% utilization is lost investment

   b. Pay for use

      - » providing static resources costs capital dollars that nobody has
      - » providing on-demand resources costs operational money that we can come up with
      - » if we can't come up with it, there's no lost investment, just lost opportunity

Goals

2. Employ framework that will extend to other clusters and environments under our purview
   a. MWT2 itself
   b. UC3

3. Return experiences and products back to the computational communities we work with
   a. other tier 2, 3
   b. OSG
   c. UChicago campus
   d. you

Targets

1. flocking, glidein to other clusters
2. virtualization
   a. KVM-based worker units within existing computation frameworks
   b. on-demand capacity in private or commercial cloud stacks
3. Amazon AWS (EC2) fits especially well
   a. pay for use
   b. ultra-cheap operation via "spot pricing": a scavenging opportunity *par excellence*

The Challenge of On-Demand Expansion

Challenges

- virtual worker launch can be slow compared to static server resources
  1. static servers are always ready to accept a new job
  2. virtual servers may need to be constructed and deployed before jobs may be scheduled
- especially so with EC2 and spot pricing

Challenges

- overprovisioning is attractive
  - » excess capacity ready to meet as-yet-unscheduled demand
- but dangerous
  - » if you overprovision by much, you effectively have relatively expensive static resources, not opportunistic dynamic growth
- we can launch in advance, but we need to know that workers are being used
  - » job requirements (classads, etc) may prevent jobs from being eligible to a virtual tier
  - » we must avoid paying for resources that we're not using

Challenges

- how do we prepare virtual workers in advance of demand, without defeating our own goals?

An idea for a solution:

Elastic Colony Manager

Inspired by Swarms

- *Swarm Intelligence*: a behavior model for systems of semi-autonomous agents
  1. Based on behavioral studies of insects;
  2. Conventionally it describes a resource-gathering behavior, but
  3. It can also model hive growth behavior — colonization of new territory
- It can serve as a model for elastic growth of a computation cluster into new infrastructure resources

Swarm Colonization

- When an insect hive looks for new resources or territory, it sends agents out in small numbers to scout

- The discoveries of scout agents informs the hive logic for future exploration

  1. If a scout brings back food, more scouts are sent to the same vicinity
  2. If those scouts return successfully, more scouts or colonists go out and begin to gather stockpiles
  3. After stockpiling, a hive may establish a branch colony

Bearing on Computation

- A computational cluster is similar to an insect hive: many like resources working together, informed by and described by a collective logic
- Unused infrastructure is like unexplored territory, potentially rich in resources for growth
- To learn what resources it may expand into, the cluster may adopt swarm techniques for exploring and confirming territory

High-level Overview

- To always be prepared to execute new jobs, we keep one “scout” worker available in each target zone (kvm, ec2, etc)

- When the scout receives work, it triggers an iterative, *reactive* deployment sequence

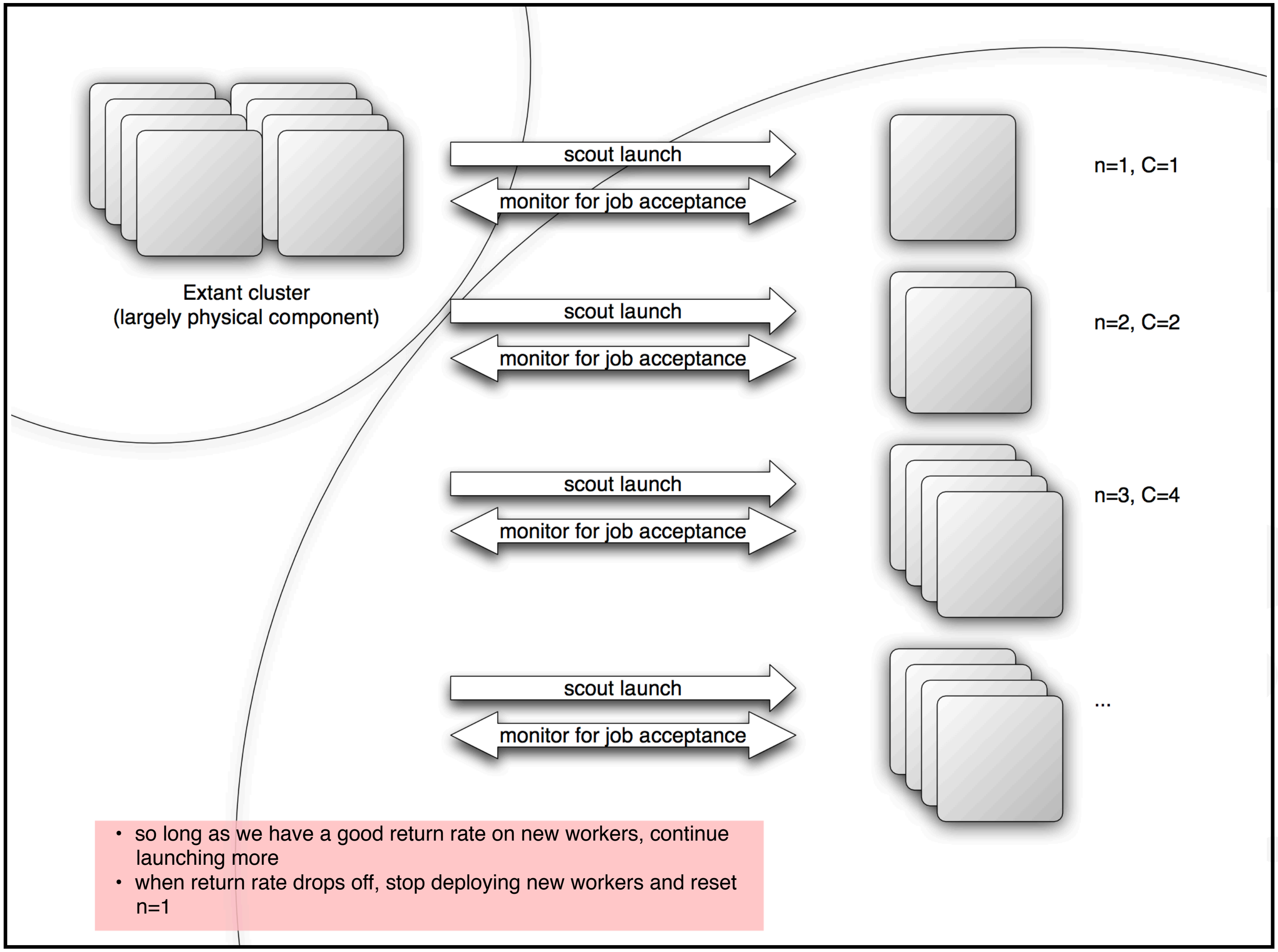
  1. Scout is iteration 1
  2. Iteration 2: launch two more workers
  3. Iteration *n*: proceed with some defined formula
     - » Avoid rapid growth, but be responsive to trends

High-level Overview

- Workers will expire (shut themselves down) after a period (T, 2h) with no work
  - » avoids surplus operational cost for no gain
- Growth rate (r) grows with successful job deployment
- Growth rate is capped at a limit (L) to avoid a sudden, expensive dropoff
  - »$r_n > r_{n-1}$ until $r_n > L$
  - »after this r_n is constant

An algorithm

- Iterative scouting


Extant cluster
(largely physical component)
scout launch
monitor for job acceptance
n=1, C=1
scout launch
monitor for job acceptance
n=2, C=2
scout launch
monitor for job acceptance
n=3, C=4
scout launch
monitor for job acceptance
...

- so long as we have a good return rate on new workers, continue launching more
- when return rate drops off, stop deploying new workers and reset n=1

Key effects

- At least one slot is always prepared to receive an initial qualifying job
- Filling that scout slot is feedback to the launcher to prepare more images
- We continue to prepare images so long as there is work
- When work stops, we shut down and stop paying
- Reactive approach ensures that we're actually using what we pay for

Questions (probably later)