Proteus: agile ML elasticity through tiered reliability in dynamic resource markets

Aaron Harlap, Alexey Tumanov*,
Andrew Chung, Greg Ganger, Phil Gibbons

Carnegie Mellon
Parallel Data Laboratory (borrowed/adapted from Aaron Harlap's Eurosys 2017 slides)

Overview

- Motivation for elasticity in ML training
  - including description of AWS spot market rules
- How to make high-performance ML Elastic
  - AgileML: an elastic Parameter Server system
- How to take advantage of Elasticity
  - BidBrain: resource control for elastic ML
- A taste of the evaluation (cost and perf.)

Dynamic Resource Availability

- Revocable resources are common in clusters
  - Best effort resources that can be preempted
  - Yarn, Borg, Mesos, etc…
- Add the element of cost savings in clouds
  - Preemptible Instances in Google Compute Engine
  - Spot Instances in Amazon EC2

Rules of the Amazon

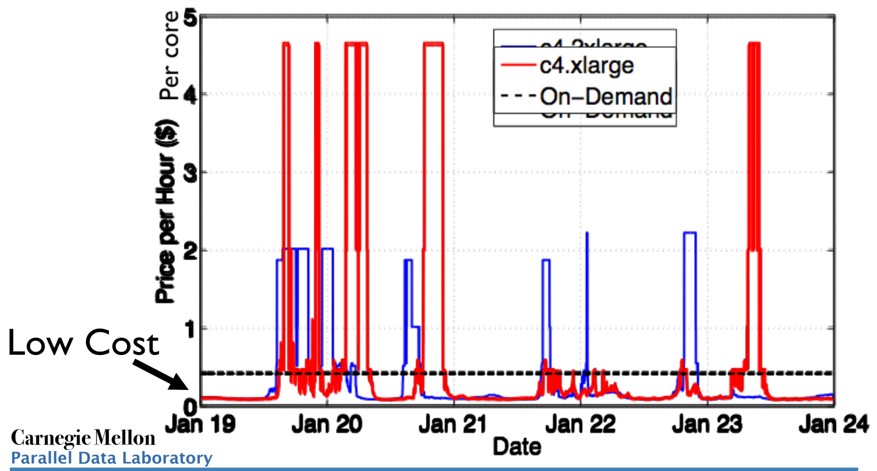
- Amazon EC2 Spot Market Rules:
  - User specifies bid, instance type, and # instances
  - User keeps resources until returned or evicted
    - Evicted if Market Price rises above Bid Price
  - User is billed Market Price at start of each hour
    - note: billed Market Price, not Bid Price! [weird, right?]
  - If evicted, partial hour is free
  - Each instance type has its own market in each zone

Observations about AWS spot

- Not rules, just things that have been observed:
  - Not a “free market” system
    - Amazon sets market prices based on unknown algorithm
    - algorithm may (and does?) change at any time
  - Price sometimes “spikes” above On-Demand price
    - seems intended to evict everyone to prevent comfort
  - Prices less correlated across markets than expected
    - even for like instances, such as c4.xlarge and c4.2xlarge

Big $$$ Saving

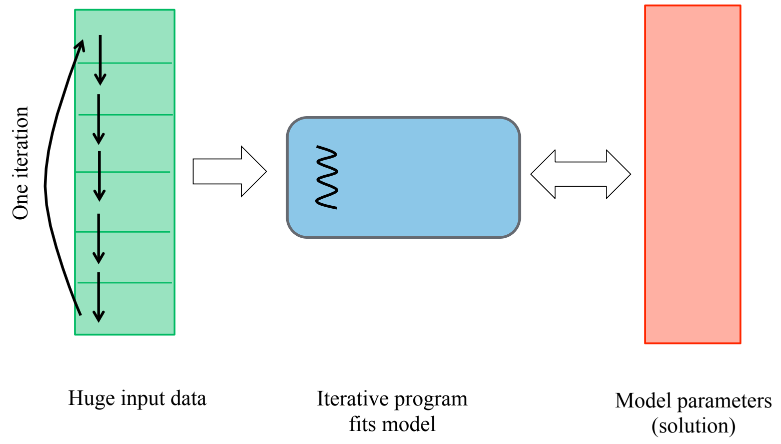
- Often 75-85% cheaper to use Spot Instances



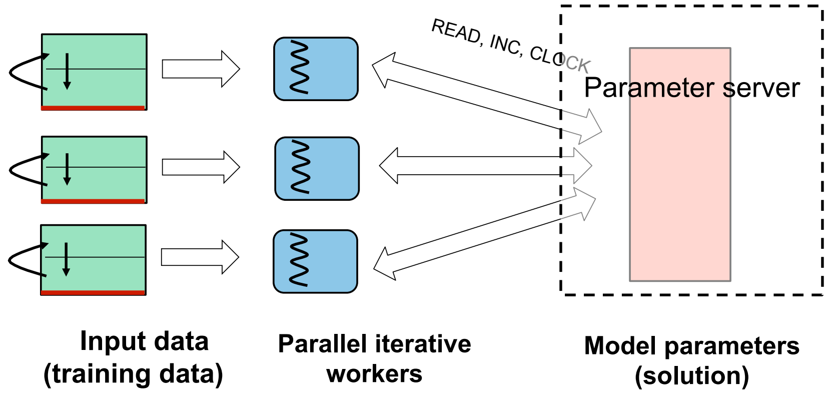
So, how can ML save $$$?

- Support agile elasticity
  - Scale in and out efficiently and quickly
- Handle bulk revocations/evictions efficiently
  - Don't lose progress
- Use spot allocations aggressively

ML model training (high level)

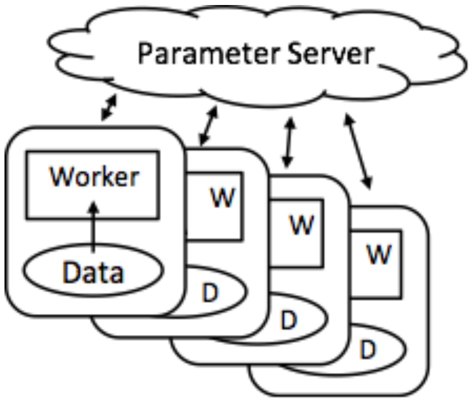


Data parallel with Parameter Server



Parameter Servers are Great for Iterative ML

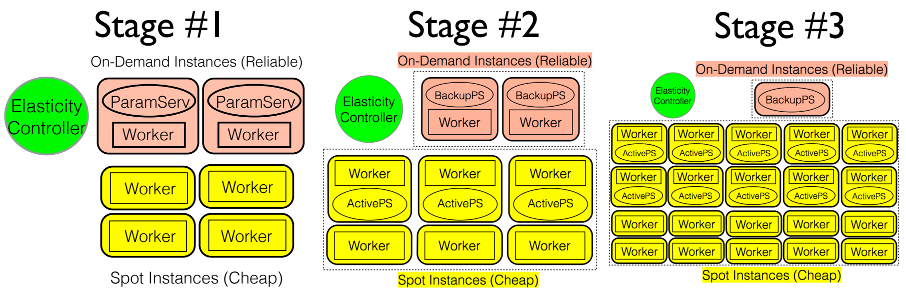
- Parameter Servers shard solution state across machines
- Traditional architecture has servers and workers on all machines
- Used by IterStore, MXNet, Bosen …



AgileML: New Approach to Elasticity

- Use tiers of reliable and un-reliable resources
  - Revocable resources are un-reliable (transient)
- Maintain all state on reliable resources
  - E.g Parameter Servers only on On-demand Instances
  - Spot Instances run workers only (initially)
- 3 architecture stages
  - based on ratio of transient to reliable resources

Building the Stages of Reliability



- Transition between stages at run-time
  - Little/No overhead for transitions
  - Transitions based on ratios

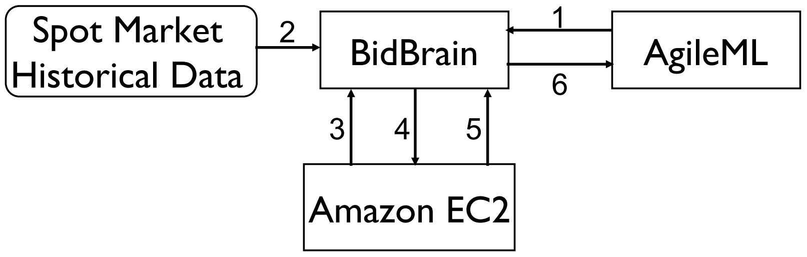
So now we have Agile Elasticity

- **How do we take advantage of it?**

- For services (e.g., Internet servers)
  - scale based on load from clients
  - not about exploiting dynamic resource availability
- For scalable batch computations, like ML training
  - Parallelize over newly available resources and contract when evicted

Carnegie Mellon
Parallel Data Laboratory

Proteus Implementation



1) Application Characteristics

2) Feed Historic Spot Market into BidBrian

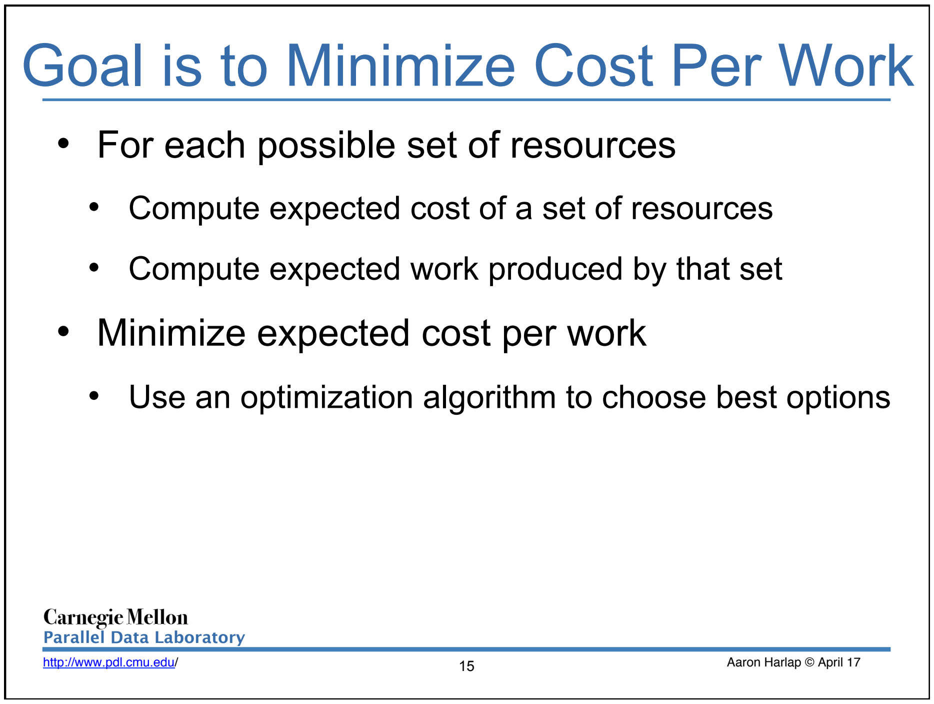
3) Feed Spot Market Price into BidBrian

4) BidBrain makes allocation request

5) AWS provides resources to BidBrain

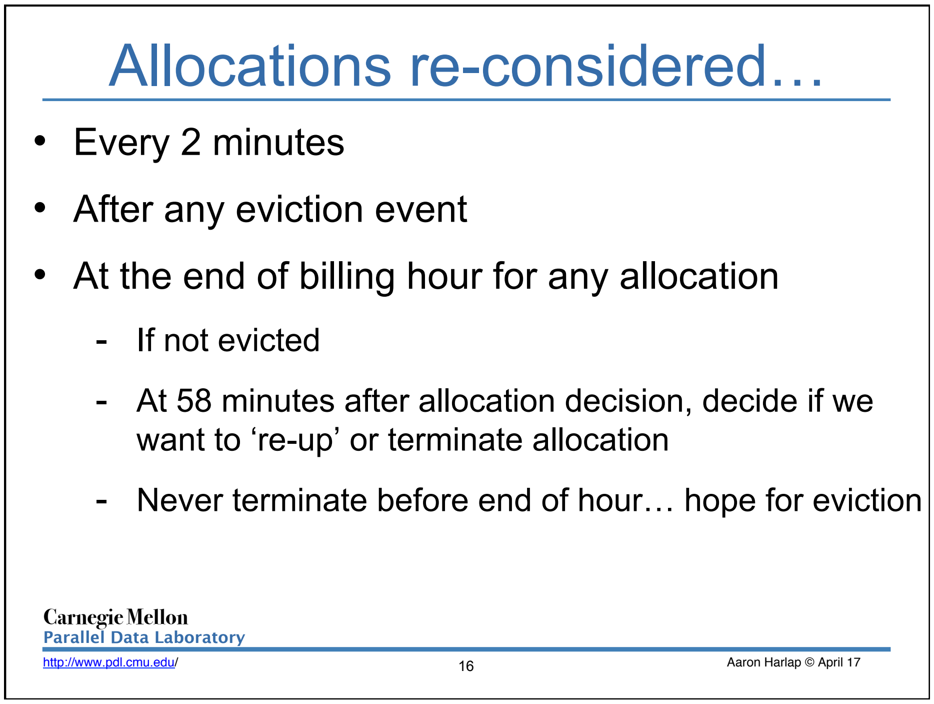
6) BidBrain provides AgileML with resources

Carnegie Mellon
Parallel Data Laboratory



Goal is to Minimize Cost Per Work

- For each possible set of resources
  - Compute expected cost of a set of resources
  - Compute expected work produced by that set
- Minimize expected cost per work
  - Use an optimization algorithm to choose best options



Allocations re-considered…

- Every 2 minutes
- After any eviction event
- At the end of billing hour for any allocation
  - If not evicted
  - At 58 minutes after allocation decision, decide if we want to ‘re-up’ or terminate allocation
  - Never terminate before end of hour… hope for eviction

Computing Expected Cost

- Consider current Market Price
- Use Market history to predict eviction likelihood
  - For a given Bid Delta = Bid Price - Market Price
- e.g., c4.2xlarge instance type in zone us-east-1a:

Bid Delta	Evicted within Hour	Expected Time to Eviction
$0.0005	55%	42 Min
$0.01	5.5%	738 Min

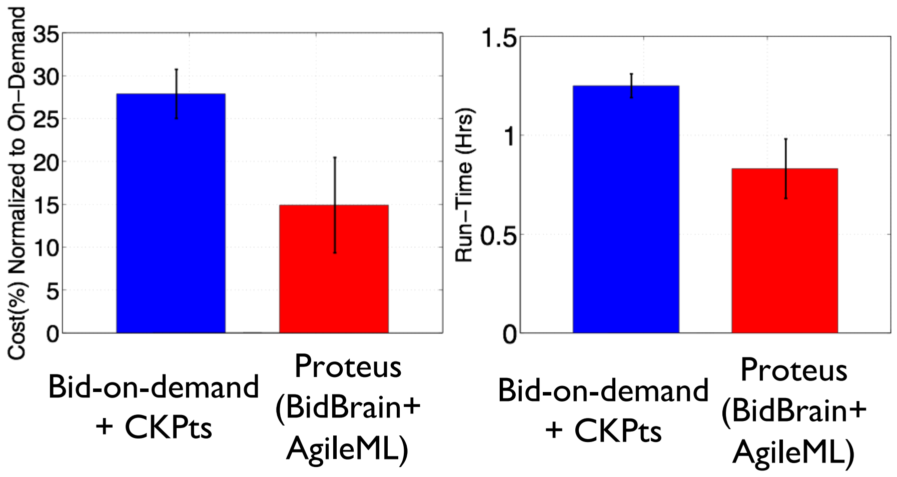
Computing Expected Work

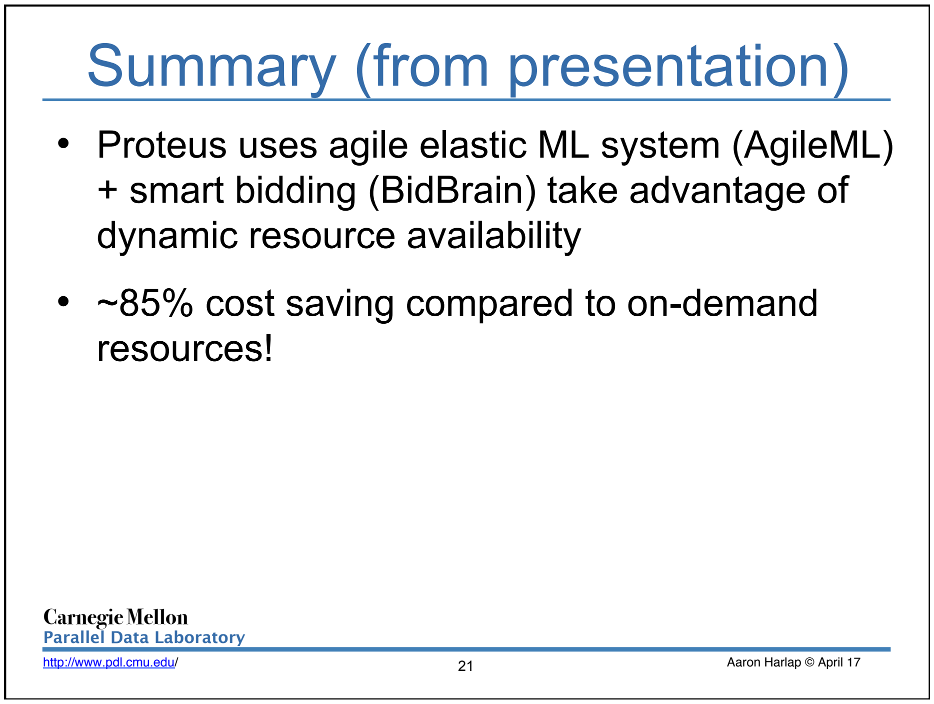
- BidBrain uses info provided by AgileML:
  - how long after startup do resources become productive
  - scalability (how well does work parallelize)
  - scale in/out overhead (work lost when changing)
  - eviction overhead (work lost when change unplanned)

Proteus Evaluation (from paper)

- Proteus vs. State-of-the-Art
  - State-of-the-Art = Checkpointing + Bid On-Demand
- AgileML vs Checkpointing
- BidBrain vs Bid On-Demand Policy
  - Bid On-Demand Policy (standard)
    - Choose cheapest resource
    - Bid On-demand Price (user bid = on-demand price)
    - On eviction, repeat

Proteus Saves Money and Time





Summary (from presentation)

- Proteus uses agile elastic ML system (AgileML) + smart bidding (BidBrain) take advantage of dynamic resource availability
- ~85% cost saving compared to on-demand resources!

Carnegie Mellon
Parallel Data Laboratory

http://www.pdl.cmu.edu/ 21 Aaron Harlap © April 17