Condor in Dynamic Environments

Ian Alderman

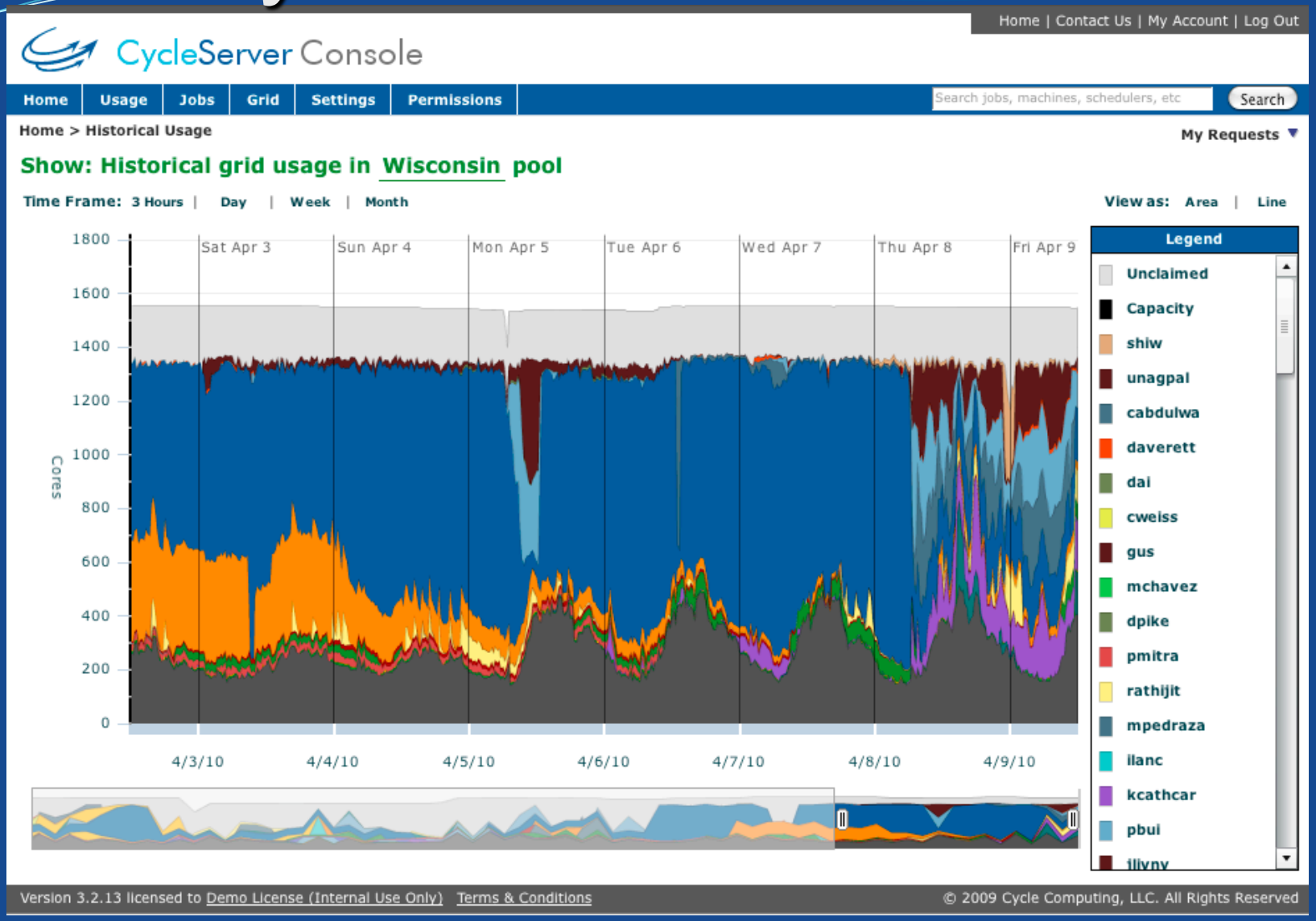
ialderman@cyclecomputing.com

A Little History…

Condor – A Hunter of Idle Workstations

- 1988 paper introducing Condor.
- Described system using cycle-scavenging: batch jobs run on desktop machines when their daytime users are idle.
- Before: 23 VAXstation II workstations utilized at ~30%.
- After: 200 CPU-days used over 5 months by 1000 jobs.
- Key features:
  - Transparent to interactive users, batch users.
  - Fair sharing among batch users.
  - Checkpointing and migration for batch jobs.

A busy week...


Home | Contact Us | My Account | Log Out
CycleServer Console
Home
Usage
Jobs
Grid
Settings
Permissions
Search jobs, machines, schedulers, etc
Search
Home > Historical Usage
My Requests
Show: Historical grid usage in Wisconsin pool
Time Frame: 3 Hours | Day | Week | Month
View as: Area | Line
Cores
1800
1600
1400
1200
1000
800
600
400
200
0
Sat Apr 3
Sun Apr 4
Mon Apr 5
Tue Apr 6
Wed Apr 7
Thu Apr 8
Fri Apr 9
4/3/10
4/4/10
4/5/10
4/6/10
4/7/10
4/8/10
4/9/10
Legend
Unclaimed
Capacity
shiw
unagpal
cabdulwa
daverett
dai
cweiss
gus
mchavez
dpike
pmitra
rathijit
mpedraza
ilanc
kcathcar
pbui
jlivny
Version 3.2.13 licensed to Demo License (Internal Use Only) Terms & Conditions
© 2009 Cycle Computing, LLC. All Rights Reserved

Fast forward to today

- Condor widely used, very successful.
  - Look around you!
- Key reason: assumption of **dynamic environment**:
  - Design goals from 1988 still valid today.

  "Workstations are autonomous computing resources and should be managed by their own users."
- High Throughput via automated policy management.
- There are differences between today's dynamic environments and idle workstations.

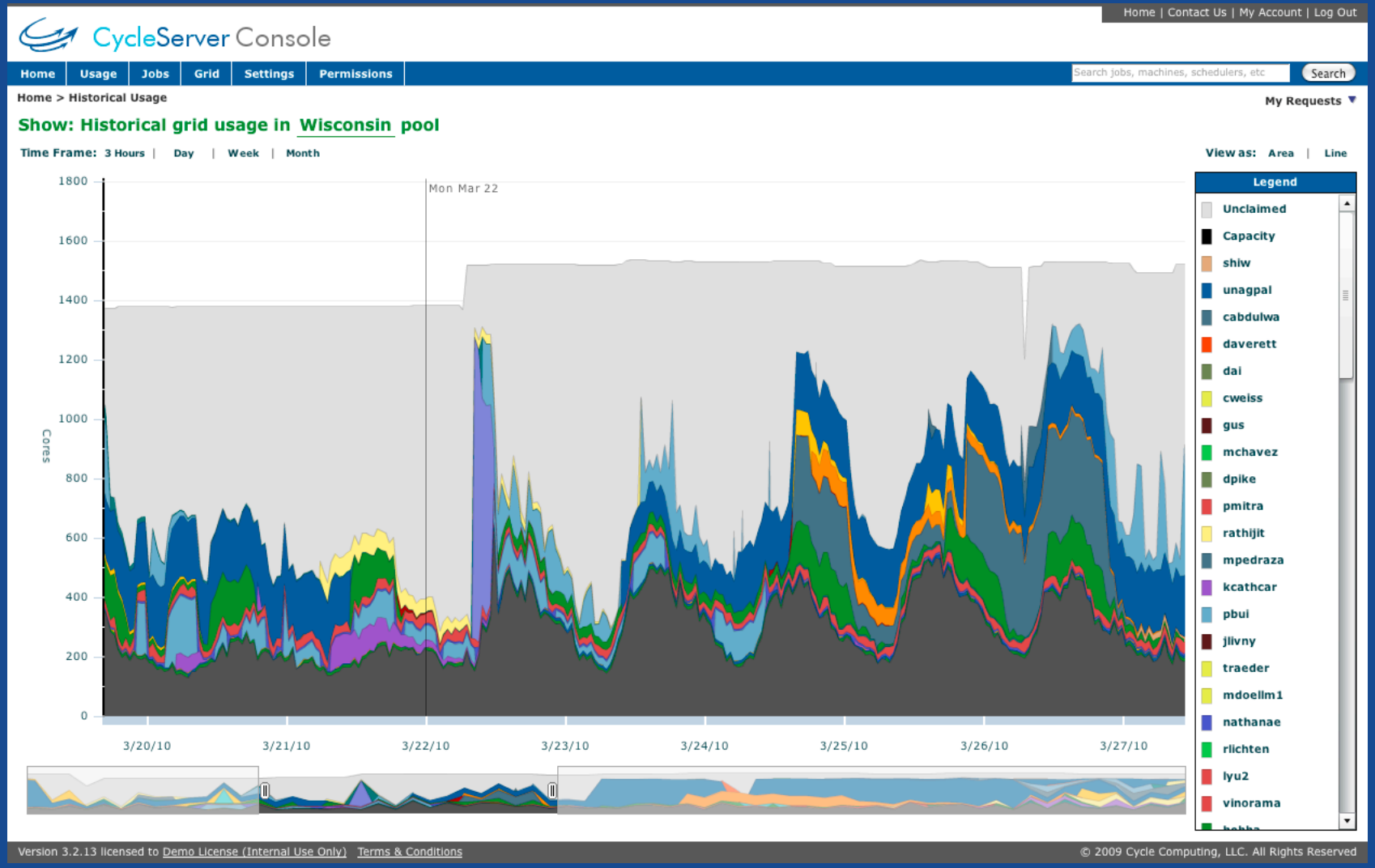
Today's Dynamic Environments?

- Virtualized Environments!
- Slots/machines come and go on demand.
- Essentially the same problem as autonomous user-managed workstations.
- Similar but different policy considerations…
  - e.g. Workstations: Round robin to distribute load.
    VMs: Fill entire host to minimize instances.
- Thesis: Even with these differences, the same features make Condor work well in both use cases…

Today: Notes from the field

- My work on CycleCloud – Managed Condor *pools* on Amazon EC2.
  - Instance Types/Profiles
  - Auto-start/stop
  - Spot instances
- VMWare Environments – tiered use of available resources.
  - Still have peak vs. median usage problem that leads to waste.

Wasted during Spring Break…



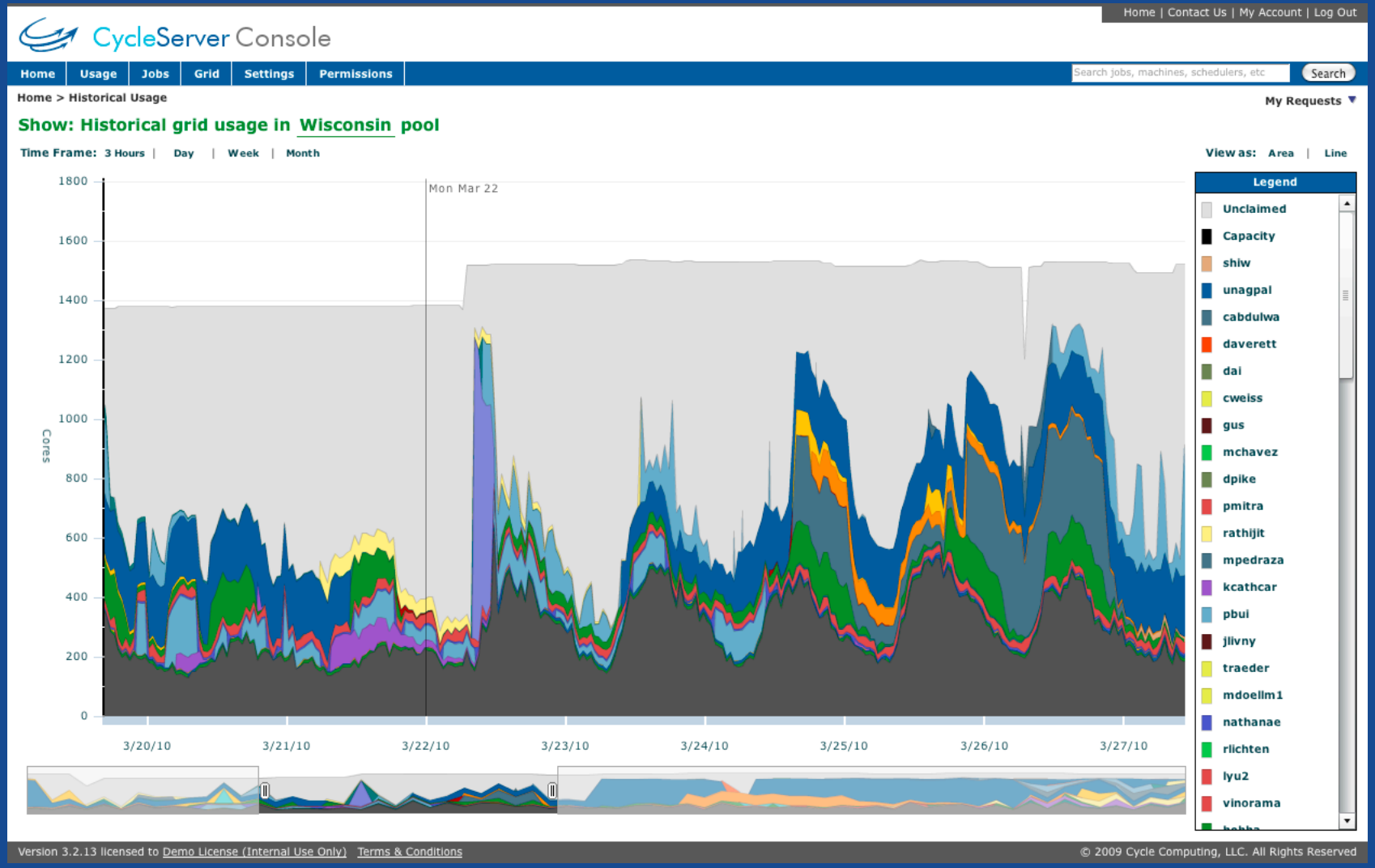
CYCLECOMPUTING

Leaders in Condor Grid and Cloud Solutions
http://www.cyclecomputing.com

EC2 Background: Amazon Elastic Compute Cloud

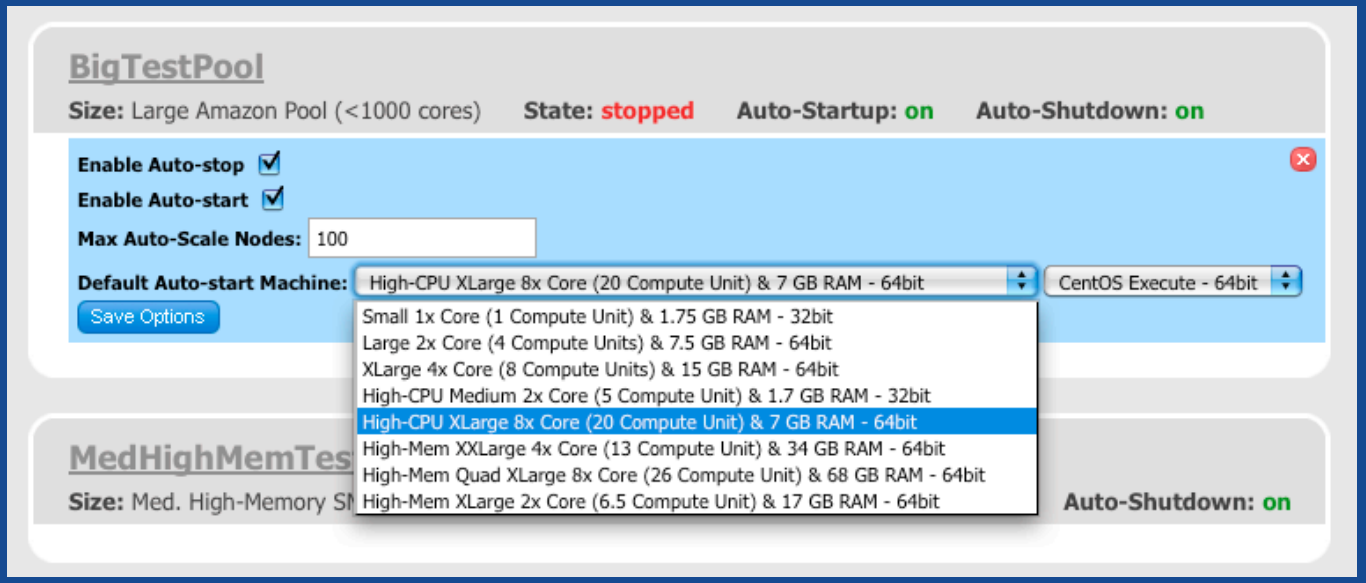
- Pay for use utility computing.
- BYOVM: you have root, they have kernel.
- Request instance; charged for each hour or portion.
- Range of machine configurations.
- Appearance of unlimited capacity.
- Proprietary highly-available storage backend: S3.

Back to Spring Break…



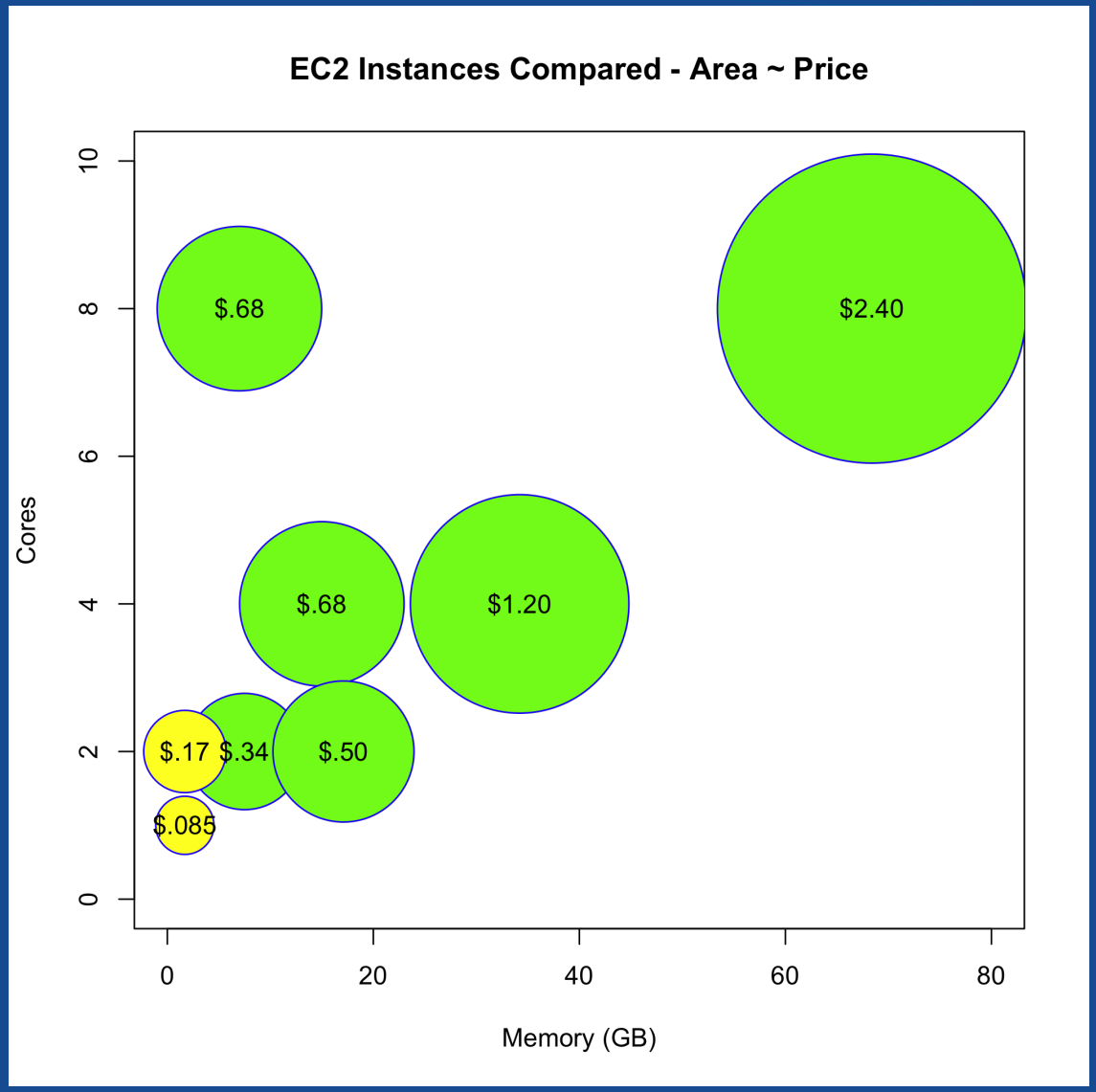
Condor pools on demand

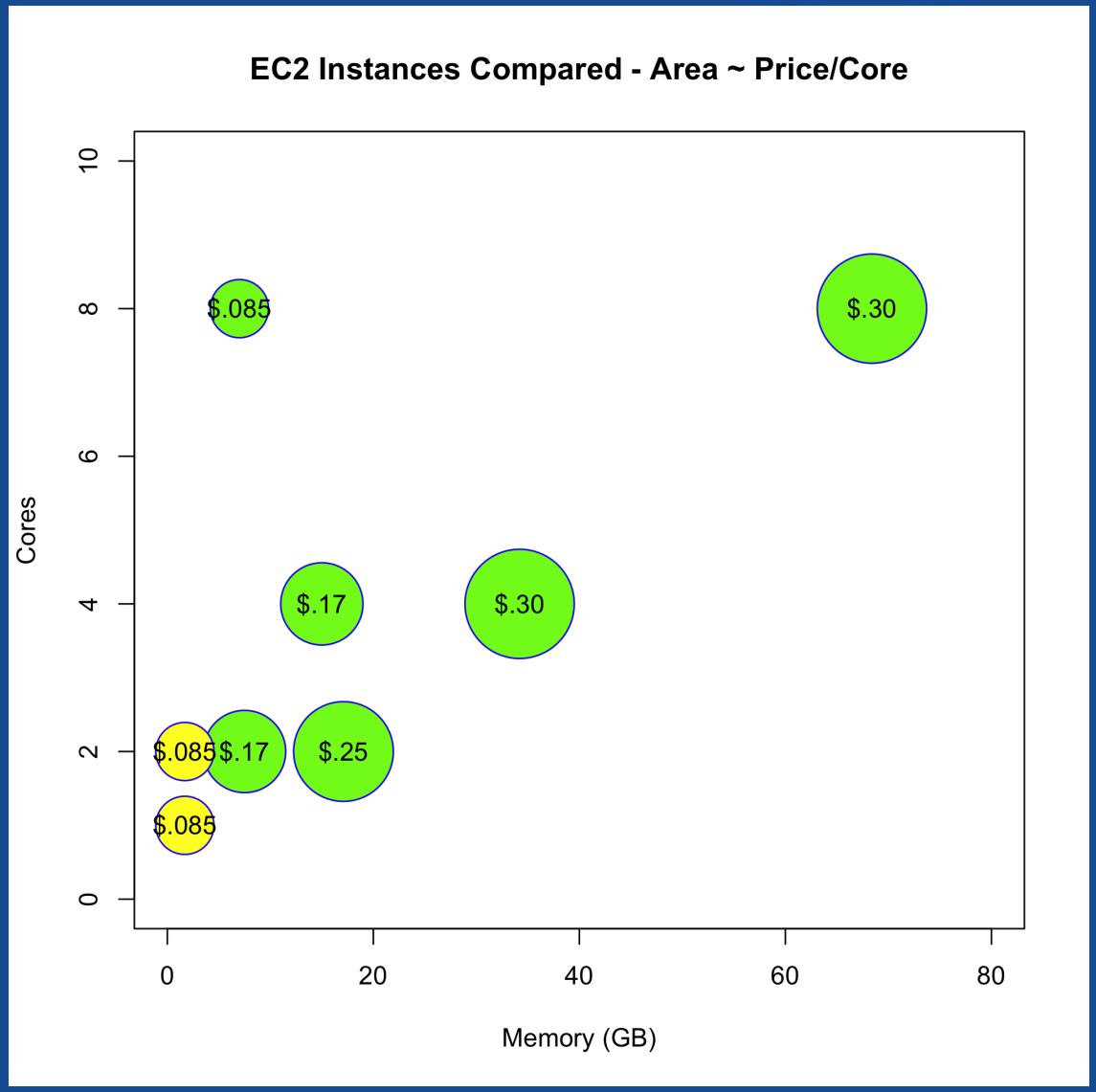
- CycleCloud Service: at-cost or supported pools.
- Relies on dynamic features of Condor: instances stop and start as needed.



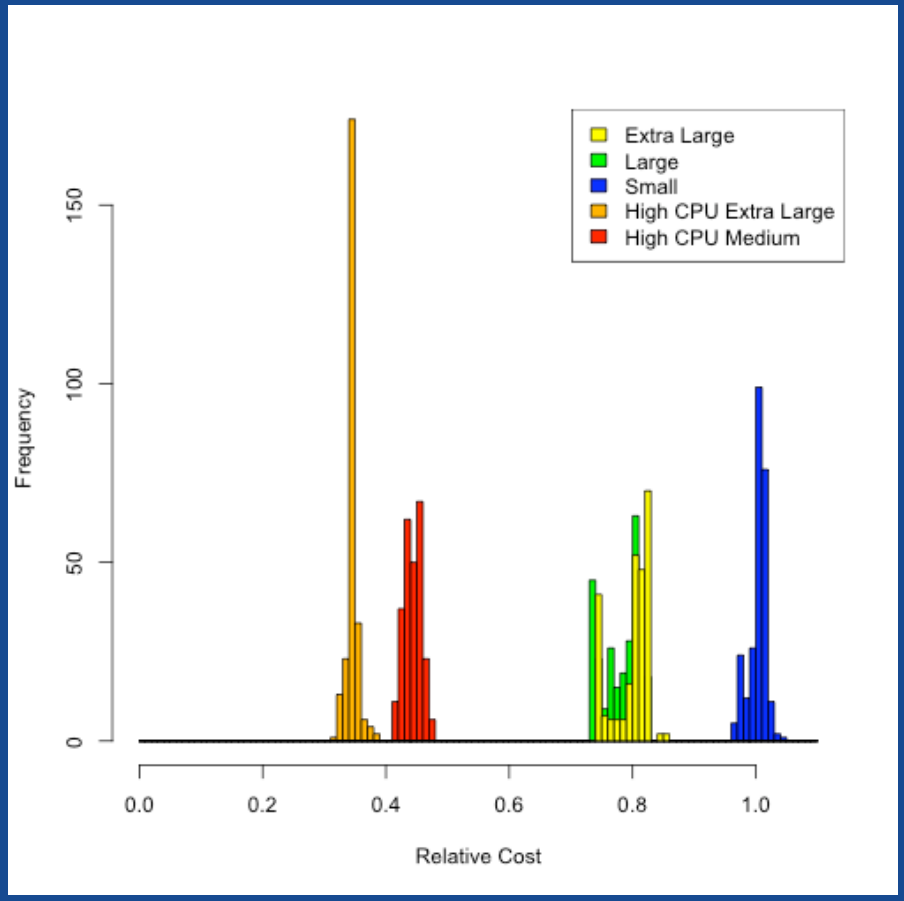
EC2 Instances Menu

AWS Name	Arch	Cores	Memory (GB)	Price (VA AZ)	Price/ Core	Memory (GB)/Core
m1.small	32	1	1.7	$0.085	$0.085	1.7
m1.large	64	2	7.5	$0.340	$0.170	3.75
m1.xlarge	64	4	15	$0.680	$0.170	3.75
c1.medium	32	2	1.7	$0.170	$0.085	0.85
c1.xlarge	64	8	7	$0.680	$0.085	0.875
m2.xlarge	64	2	17.1	$0.500	$0.250	8.55
m2.2xlarge	64	4	34.2	$1.200	$0.300	8.55
m2.4xlarge	64	8	68.4	$2.400	$0.300	8.55


EC2 Instances Compared - Area ~ Price
Cores
0
2
4
6
8
10
0
20
40
60
80
Memory (GB)
$.68
$2.40
$.68
$1.20
$.17
$.34
$.50
$.085

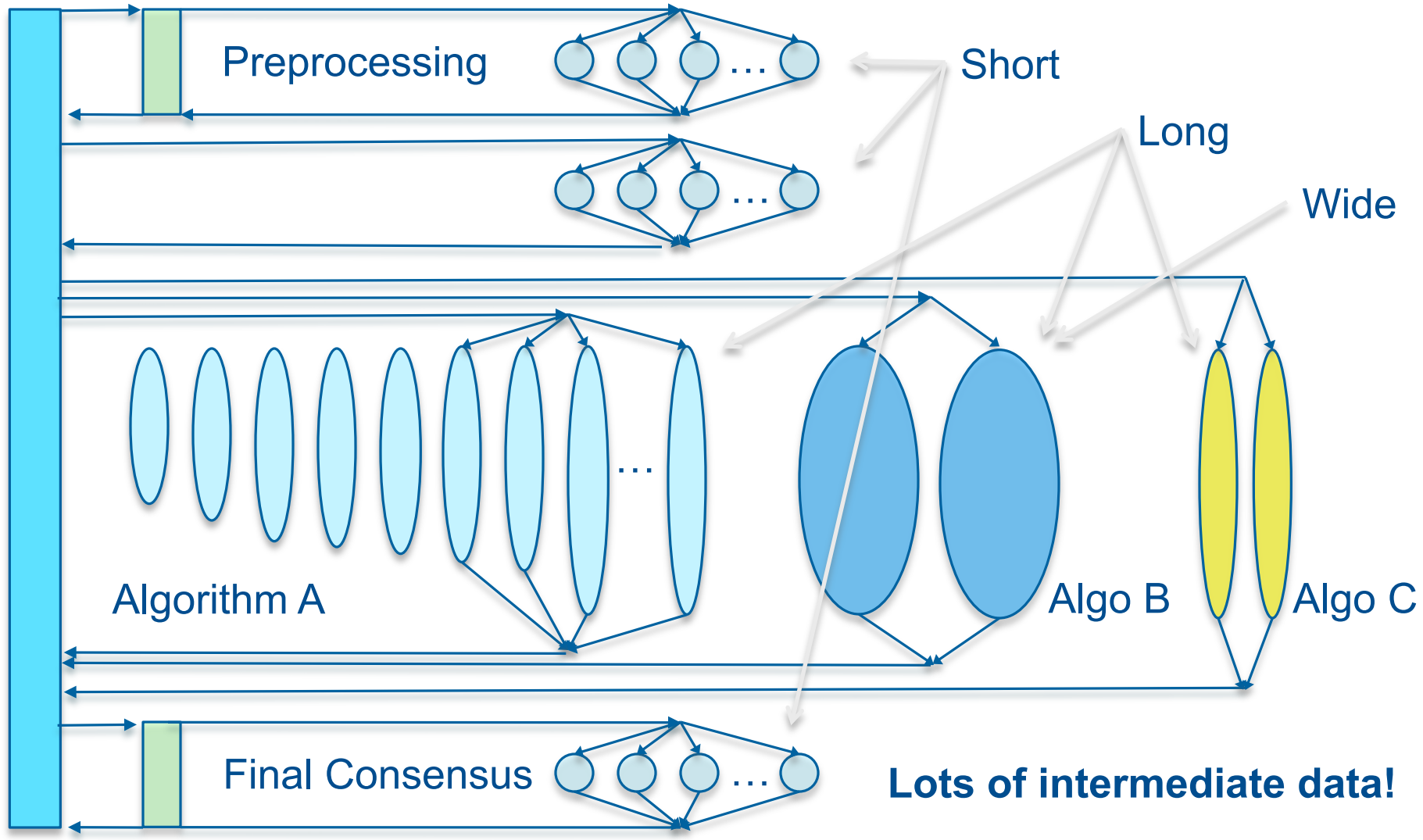

EC2 Instances Compared - Area ~ Price/Core
Cores
0
2
4
6
8
10
0
20
40
60
80
Memory (GB)
$.085
$.30
$.17
$.30
$.085
$.17
$.25
$.085

High-CPU Amazon EC2 nodes have best price/performance

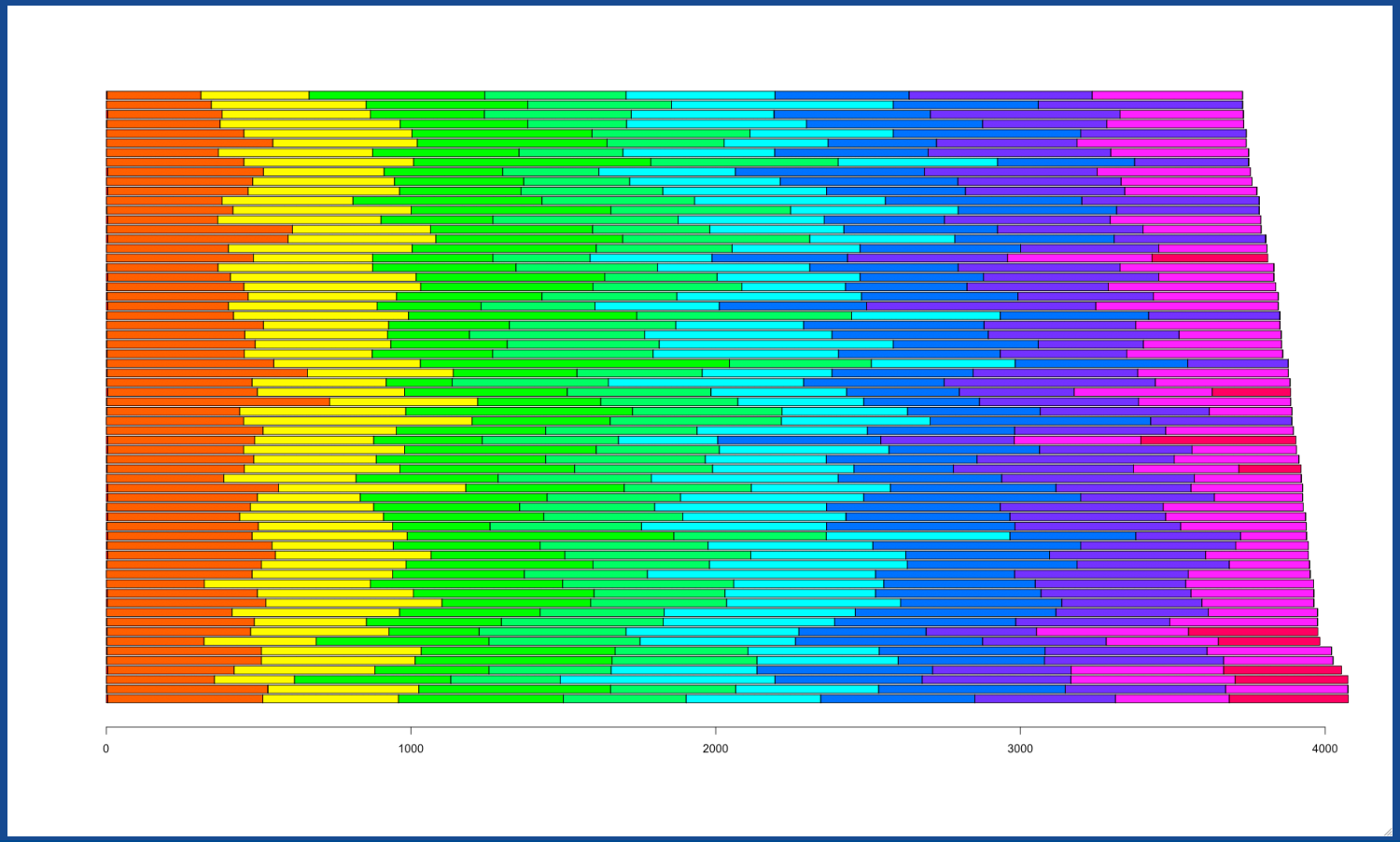


Making Condor work in EC2

- Naïve: Boot instances in EC2, add to pool.
- Better: Create a pool in EC2.
- No Condor configuration changes *required*, but…
  - Disable preemption.
  - Configure authentication, integrity, encryption.
  - Optimize for security, performance, scalability.
- Additional configuration:
  - Patches, information flow, installed applications, shared files, encrypt data at rest and in transit, etc…


Preprocessing
Short
Long
Wide
Algorithm A
Algo B
Algo C
Final Consensus
Lots of intermediate data!

Why High Throughput leads to Efficient Computing



Auto-start, auto-stop mechanisms

Auto-start: Jobs present in queue ➔ machines start.

- Jobs run!
- User sets type of machine, limit # of instances to start.

Auto-stop: Before beginning of next hour:

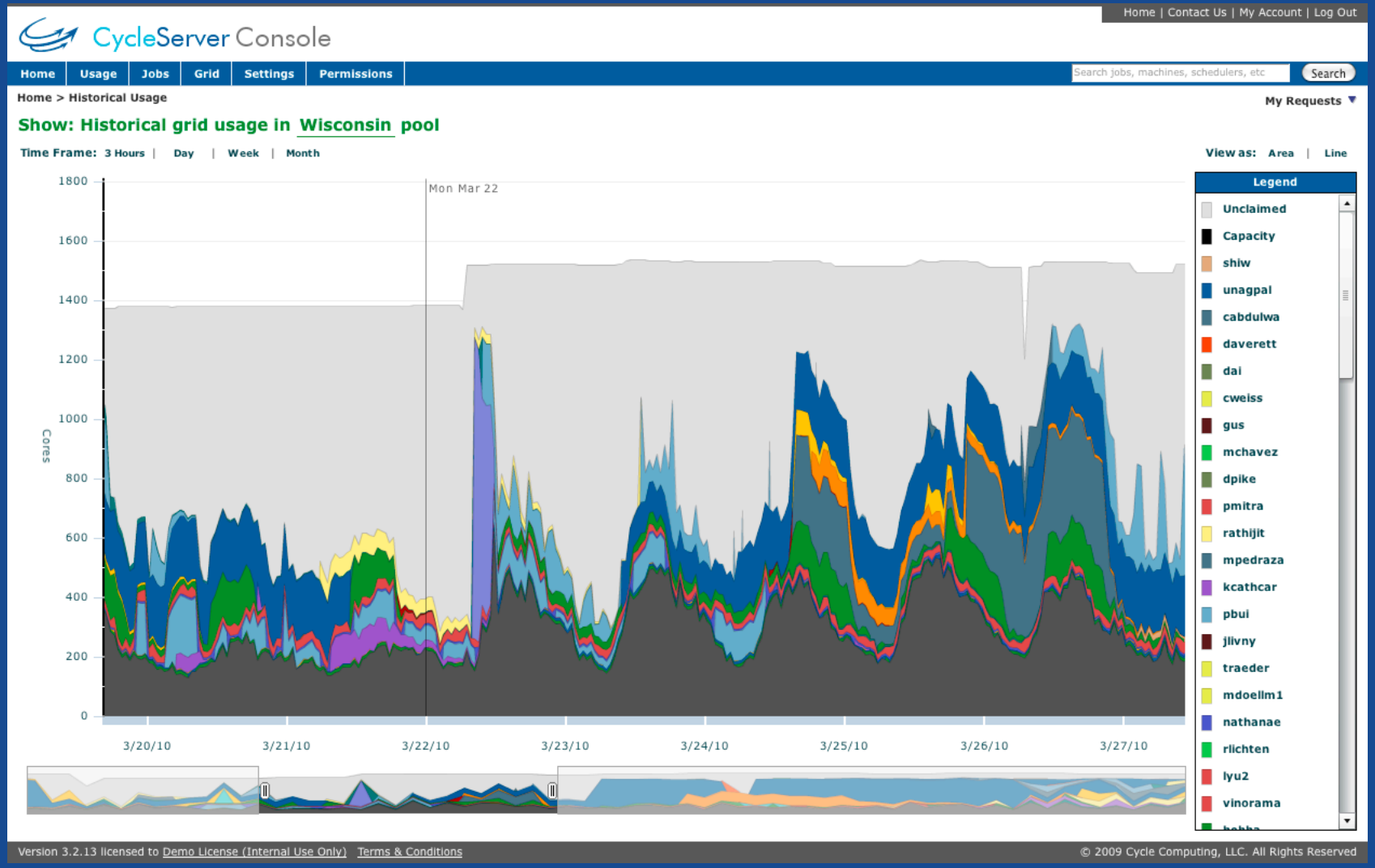
- Check to see if jobs still running – if not, shut down.

- Users manually start machines that will auto-stop.
- Mechanisms for auto-starting different machine types based on user requirements.
- Users can supply *hints* about job run time for auto-start.

Spot Instances

- Same set of instance options.
- Lower cost, weaker service characteristics:
  - Could go away at any time.
  - If it goes away, you don't pay.
- Bid for maximum you're willing to pay, get machines at that price if available (i.e. going rate is <=).
- If going rate goes above your maximum, your instances are terminated.

Spring Break volatility…



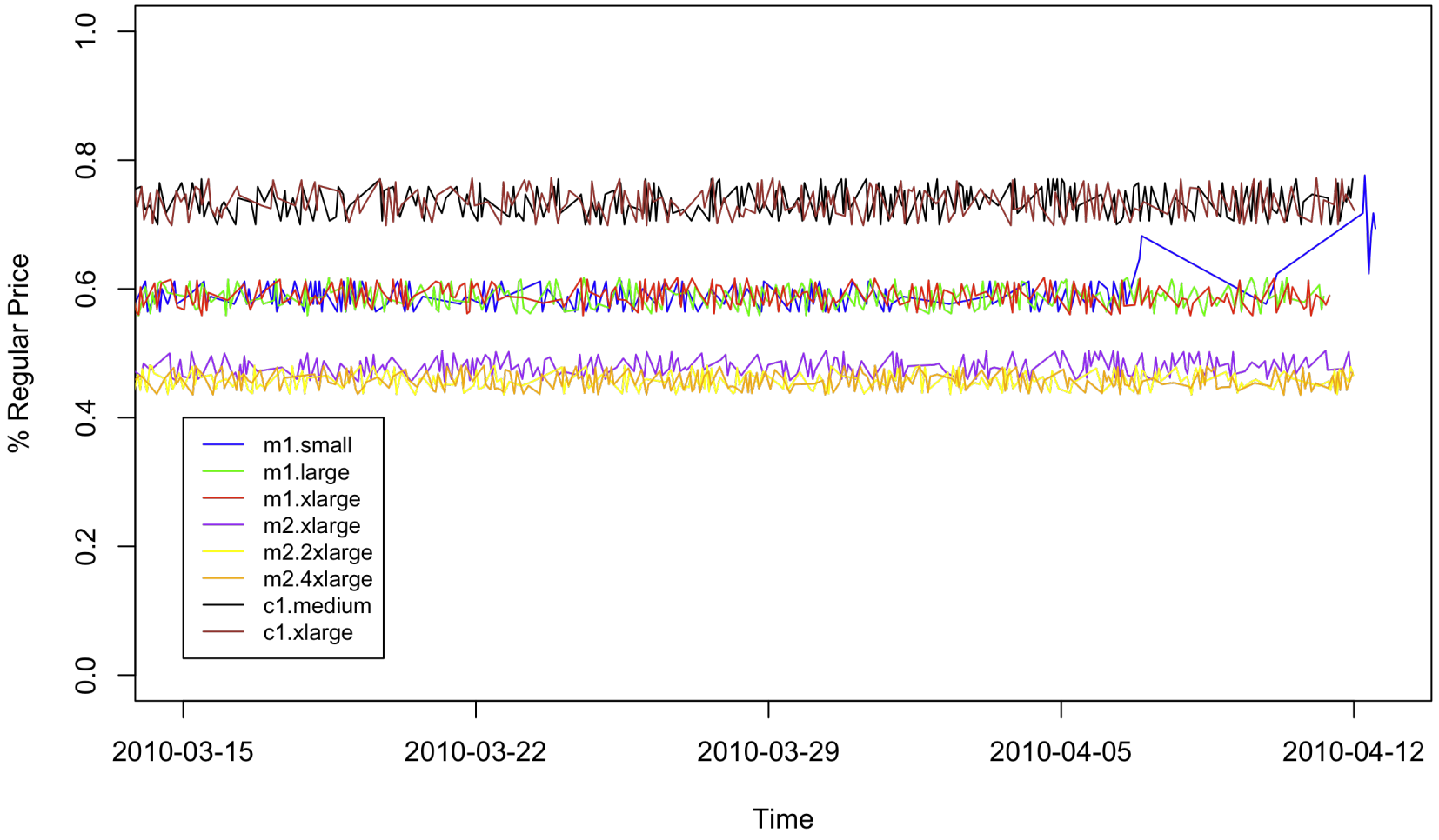
Spot Instance Policy Brainstorm

- Jobs expected to run for more than one hour need dedicated resources to avoid waste (getting billed but not finishing)
  Job:
  REQUIREMENTS = isSpotInstance =?= FALSE

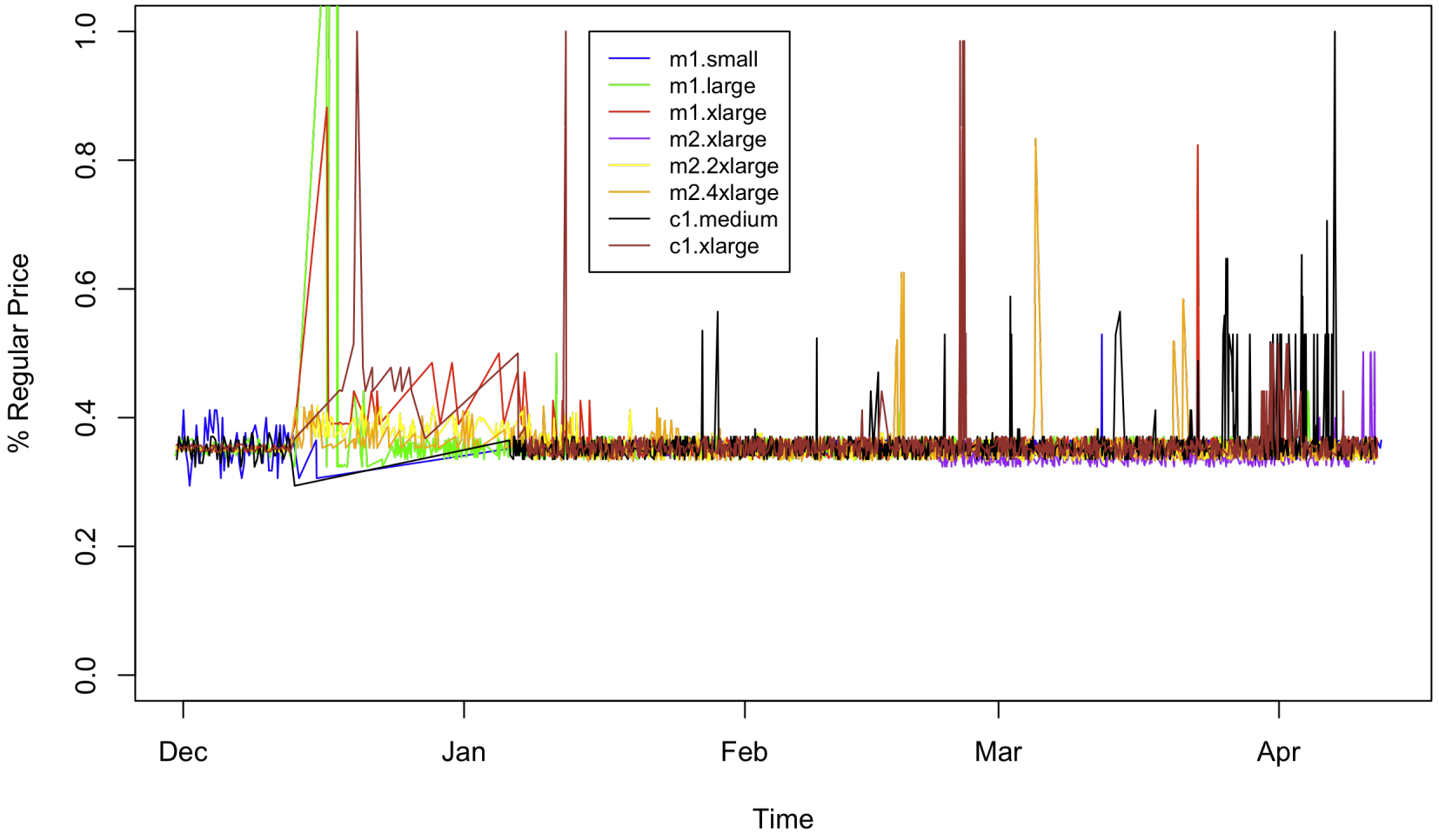
  Machine:
  START = Target.EstimatedRuntime =?= UNDEFINED ||
  Target.EstimatedRuntime >=3600
  isOwner = False

- Jobs run on the cheapest nodes possible
- Jobs prefer to run on machines up for lower fractions of an hour (to allow auto-stop to work)
  RANK = 100 * SlotHourCost + FractionOfHourUP

Spot instance prices over time: Windows



Spot instance prices over time: Linux



Some folks don't want EC2

- What about internal VM environments?
  - Most places have them, and they're over provisioned
- VMWare environments:
  - Help with server consolidation (Run 40 VMs on beefy servers rather than 40 servers).
  - Still have peak vs. median usage problem.
  - For example, 500 Core VMWare environment running SAP that is 25-40% utilized on average, but still needs all that hardware for peak.

VMWare tiered applications

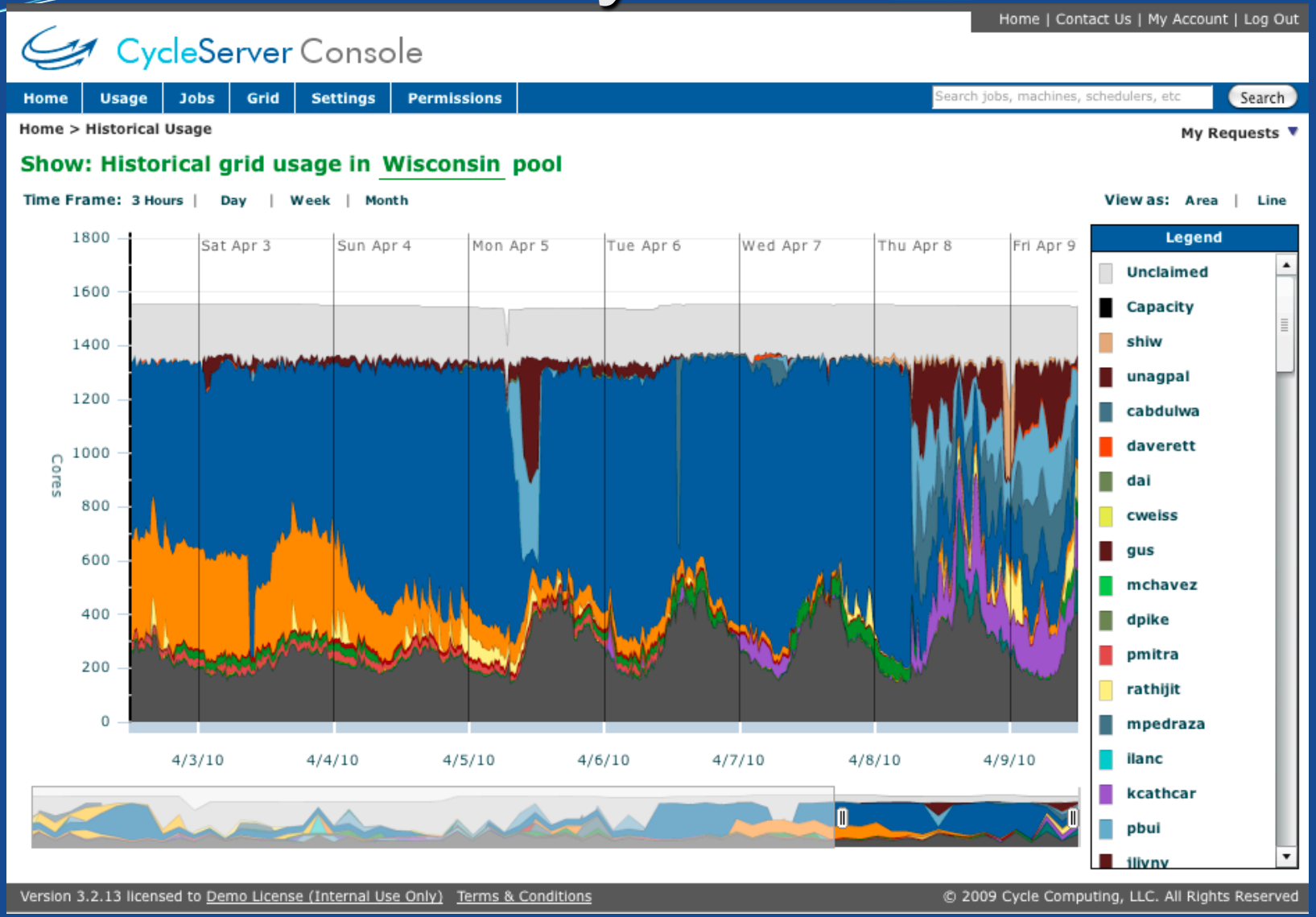
- Thankfully VMWare has tiers:
  - Production (PRD) usurps User Acceptance Testing (UAT) environment, which usurps Dev(DEV) environment.
- Perfect for harvesting (just like Spot Instances).

- Create a GRID tier that PRD/UAT/DEV usurp for resources and have the cores join locally.
- Add VMs to pool when there are jobs in queue, and remove GRID when PROD/UAT need them back.

VMWare Policy Use Cases

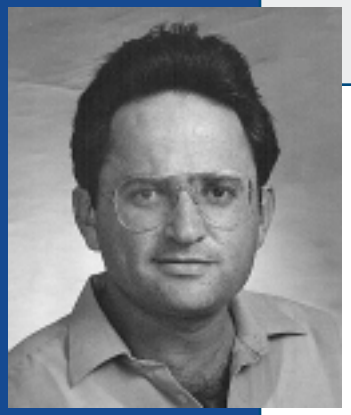
- Same high level policies work:
  - Jobs prefer dedicated nodes.
- Still cycle harvesting, just from a multi-core VM environment rather than just workstations.
- Just like auto-start, we'll add VM requests to VMWare when there are jobs, remove them when they're idle.
- Goal: Turn 40% Utilization to 90-95% utilization using technical computing workloads.

Back to a busy week…



A lot has changed…

1988	2010
The original 200 CPU days	~11 CPU minutes
~3 CPU years	~I core hour



Thanks to Moore's Law

But some things have stayed the same:

1988	2010
“We need more compute power”	“We need more compute power”
Dynamic Environments are plentiful	Dynamic Environments are plentiful