Elastic Efficient Execution of Varied Containers

Sharma Podila
Nov 7th 2016, QCon San Francisco

NETFLIX

In other words...

How do we efficiently run heterogeneous workloads on an elastic pool of heterogeneous resources, with capacity guarantees?

Topics

- Containers, Mesos, Fenzo - where are we today?
- Modeling an elastic Mesos cluster
- Capacity guarantees for varied applications
- Network resource and security groups
- Ongoing and future work

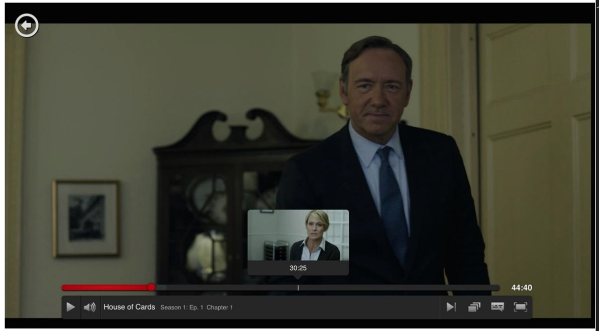
About Me

- Software engineer
  - Resource scheduling, stream processing, distributed systems
  - Netflix Edge Engineering
  - Sun Microsystems + Oracle Corp.

- Author of Fenzo scheduling library
  https://github.com/Netflix/Fenzo

What is Netflix?

Stream TV shows and movies anywhere, any time.



NETFLIX ORIGINAL

Chelsea

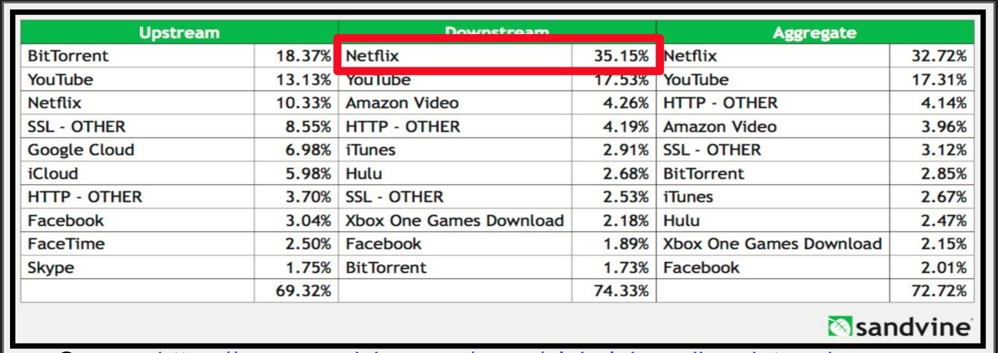
NEW "Please Take My Knickers Off"

Stella and Mary McCartney talk fashion, royalty and sisterhood. Craig Ferguson offers a primer on Scotland. Plus, Colleen Ballinger and Kimbal Musk.

NETFLIX ORIGINALS

NETFLIX ORIGINAL BLACK MIRROR NEW EPISODES
NETFLIX ORIGINAL LUKE CAGE
NETFLIX ORIGINAL easy
NETFLIX ORIGINAL Grace and Frankie
NETFLIX ORIGINAL STRANGER THINGS
NETFLIX ORIGINAL THE FALL NEW EPISODES

81 Million subscribers worldwide and growing!

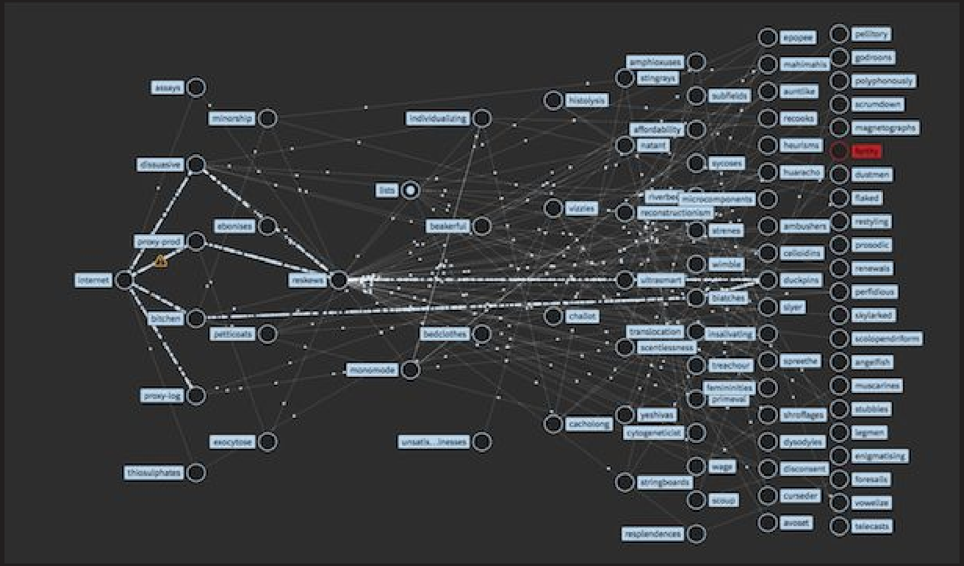


Upstream		Downstream		Aggregate	
BitTorrent	18.37%	Netflix	35.15%	Netflix	32.72%
YouTube	13.13%	YouTube	17.53%	YouTube	17.31%
Netflix	10.33%	Amazon Video	4.26%	HTTP - OTHER	4.14%
SSL - OTHER	8.55%	HTTP - OTHER	4.19%	Amazon Video	3.96%
Google Cloud	6.98%	iTunes	2.91%	SSL - OTHER	3.12%
iCloud	5.98%	Hulu	2.68%	BitTorrent	2.85%
HTTP - OTHER	3.70%	SSL - OTHER	2.53%	iTunes	2.67%
Facebook	3.04%	Xbox One Games Download	2.18%	Hulu	2.47%
FaceTime	2.50%	Facebook	1.89%	Xbox One Games Download	2.15%
Skype	1.75%	BitTorrent	1.73%	Facebook	2.01%
	69.32%		74.33%		72.72%

sandvine

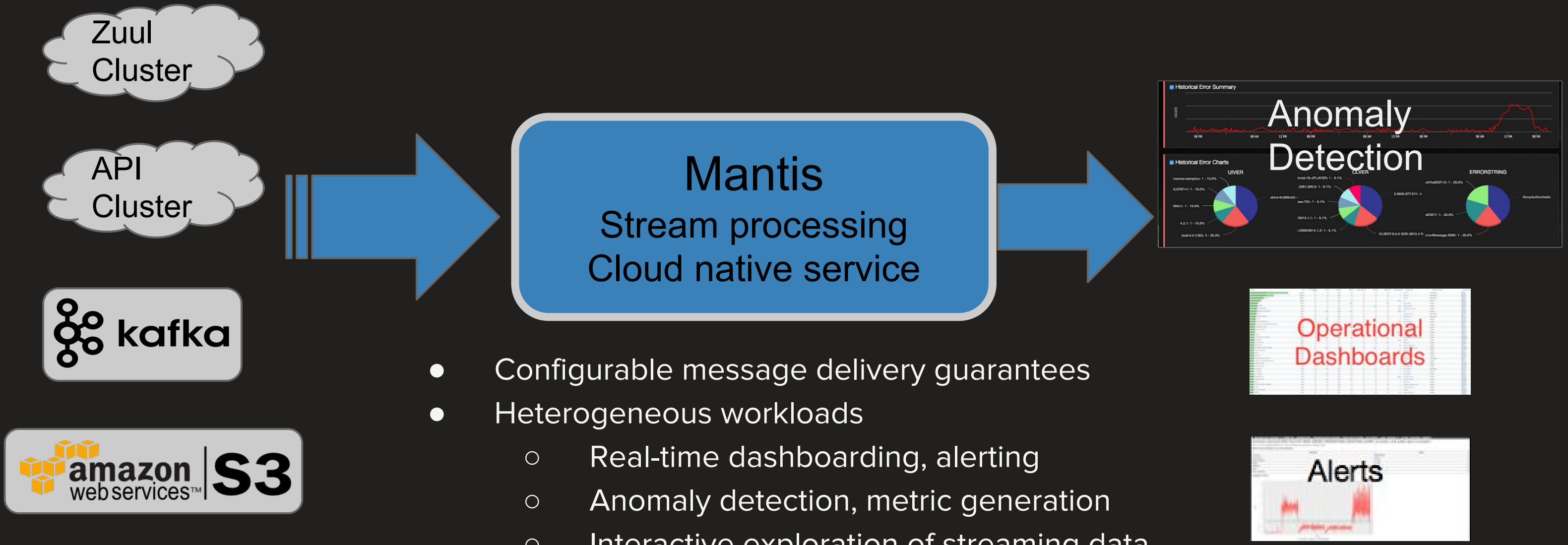
Source: https://www.sandvine.com/news/global_broadband_trends.asp

Microservices architecture on AWS EC2



Containers, Apache Mesos, Fenzo - where are we today?

Reactive stream processing: Mantis

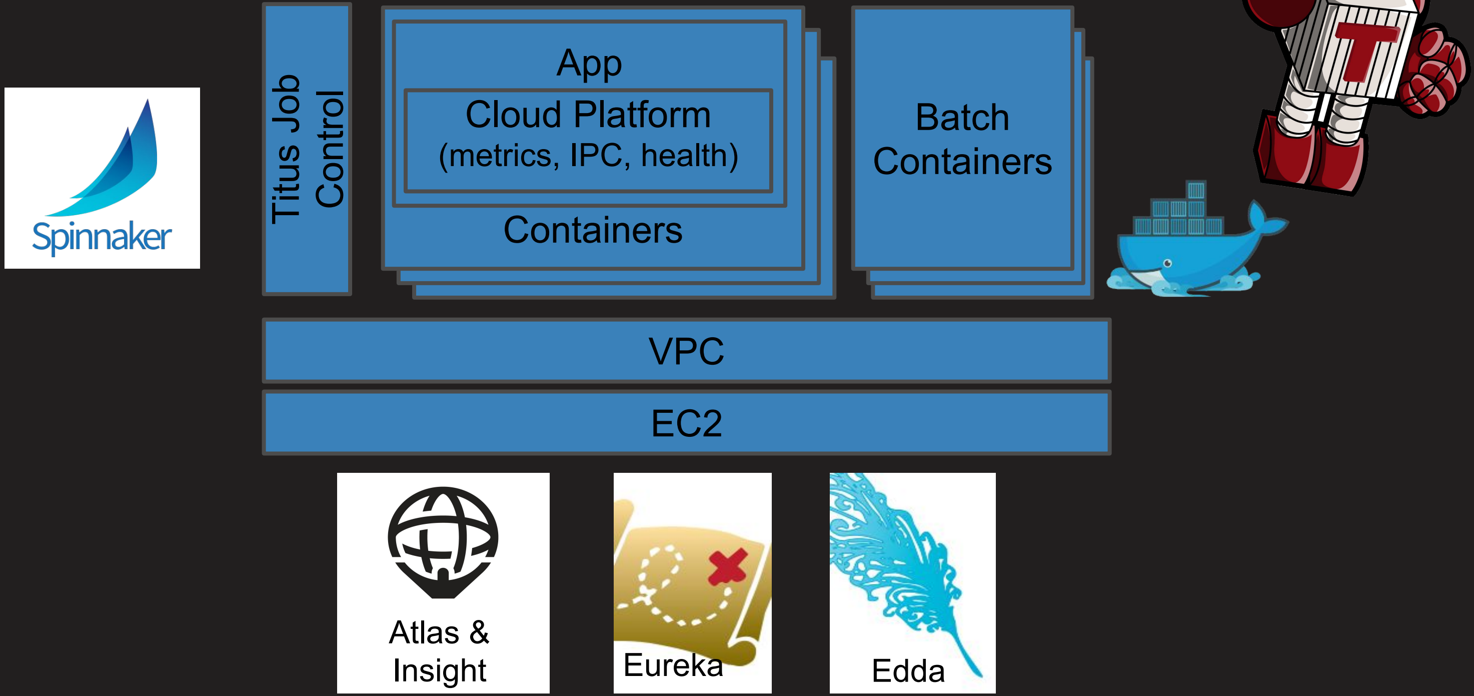


- Configurable message delivery guarantees
- Heterogeneous workloads
  - Real-time dashboarding, alerting
  - Anomaly detection, metric generation
  - Interactive exploration of streaming data

Current Mantis usage

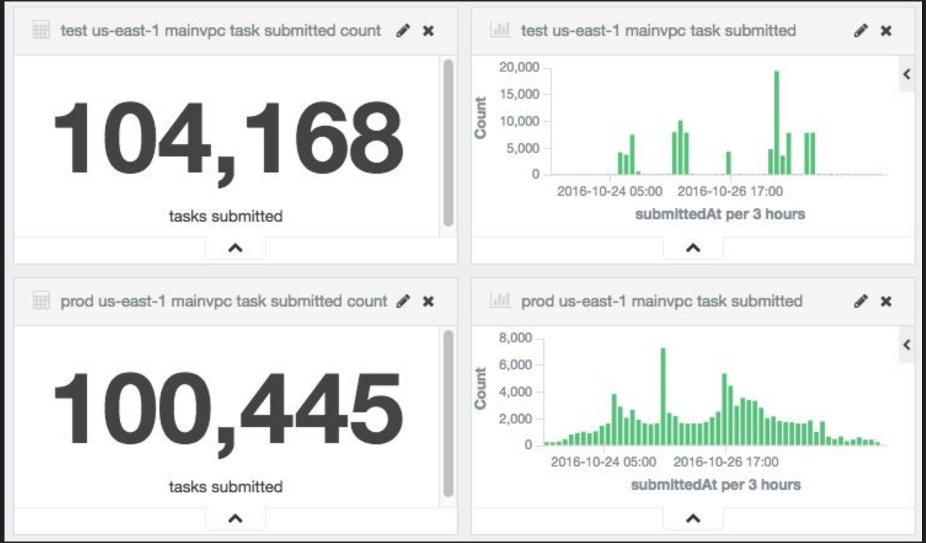
- Peak of 1,800 EC2 instances
    - M3.2xlarge instances
- Peak of 3,700 concurrent containers
    - Trough of 2,700 containers
- Mix of perpetual and interactive exploratory jobs
- Peak of 11 Million events / sec

Container deployment: Titus



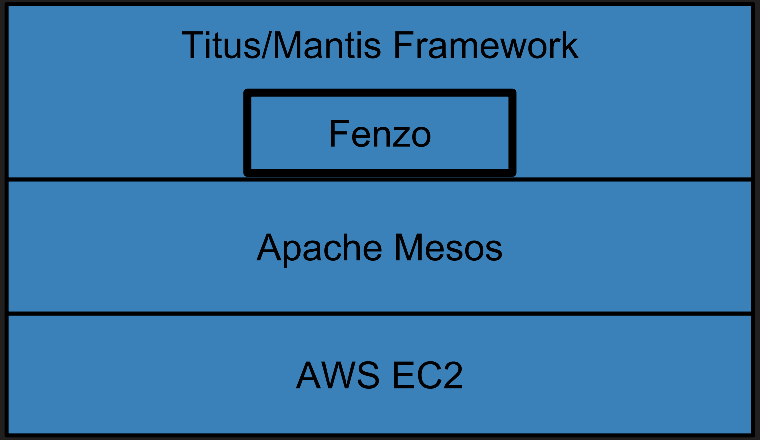
Current Titus usage

- Peak of ~1,800 instances
  - Mix of m4.4xl, r3.8xl, g2.8xl
  - ~800 instances at trough
- Mix of batch, stream processing, and some microservices



#Containers (tasks) for the week of 10/24 in one of the regions

Core architectural components



Fenzo at
https://github.com/Netflix/Fenzo

Apache Mesos at
http://mesos.apache.org/

Jobs, tasks, instances, containers

Jobs can be one of batch, service, or stream processing type of jobs

A jobs has one or more tasks to run

- An instance is equivalent to a task

A task runs one container

A few common themes

Heterogeneous mix of jobs and resources

Resource	Task request	Agent sizes
CPU	1 - 32 CPUs	8 - 32 CPUs
Memory	2 - 200+ GB	32 - 244 GB
Network bandwidth	10 - 2000 Mbps	1024 - 10240

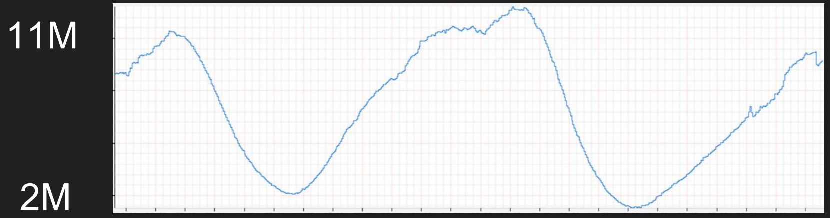
Resource affinity based on task type

Task locality

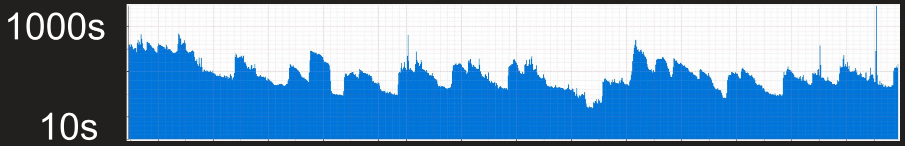
A few common themes

Large variation in peak to trough resource requirements

Mantis events/sec



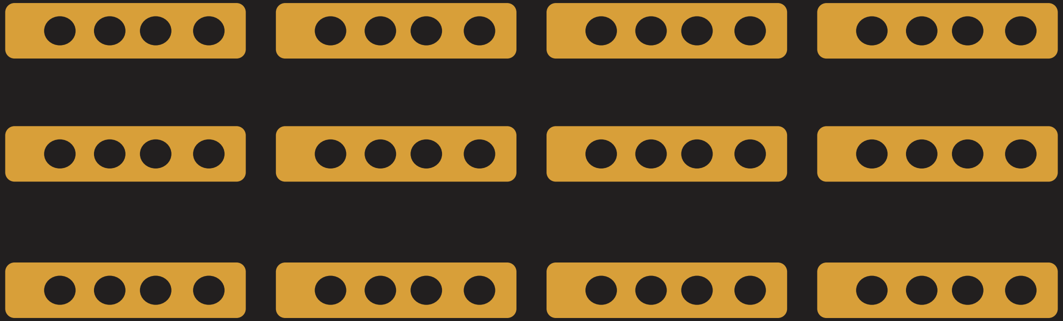
Titus concurrent containers



Modeling an elastic Mesos cluster

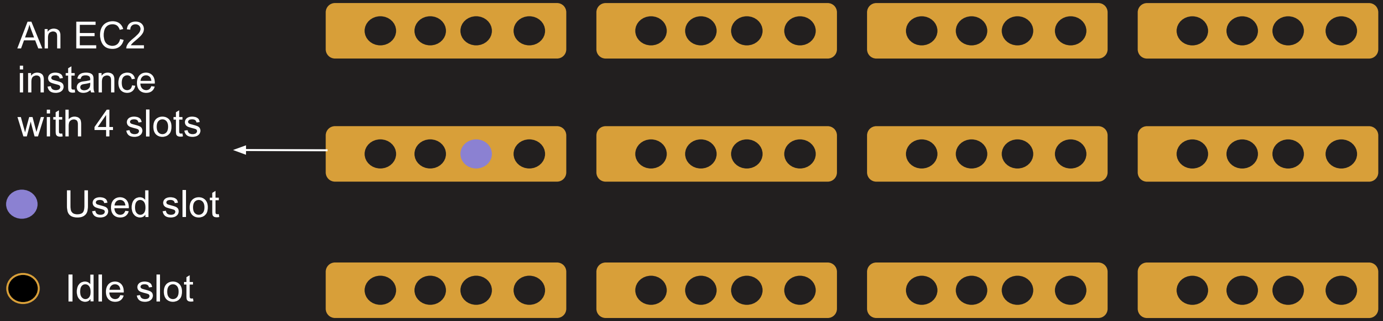
Can we resize agent cluster based on demand?

Task assignments in a cluster



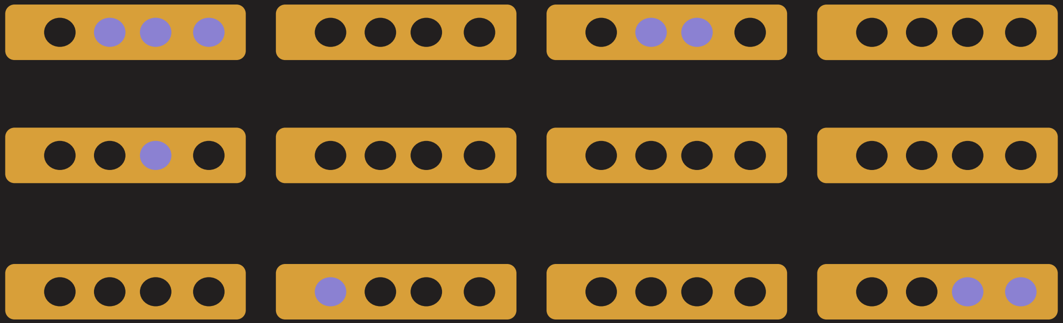
Consider a cluster with 4-slot hosts

“Random” assignments in a cluster



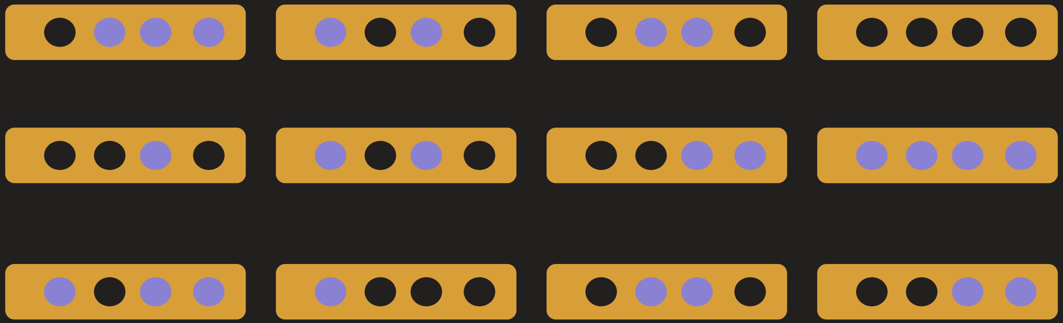
Cluster starts random assignments of resources to tasks

“Random” assignments in a cluster



Cluster starts to fill up...

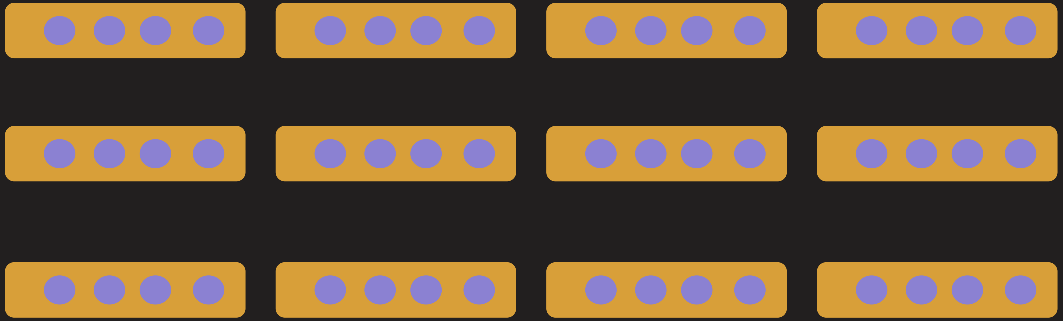
“Random” assignments in a cluster



About 50% utilized

Cluster somewhat full.
But, only 1 agent can be terminated for scale down without losing jobs

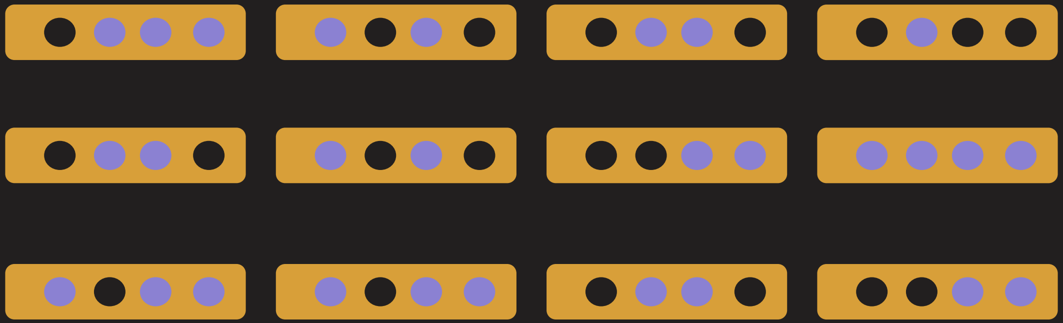
“Random” assignments in a cluster



100%
utilized

Cluster is now full

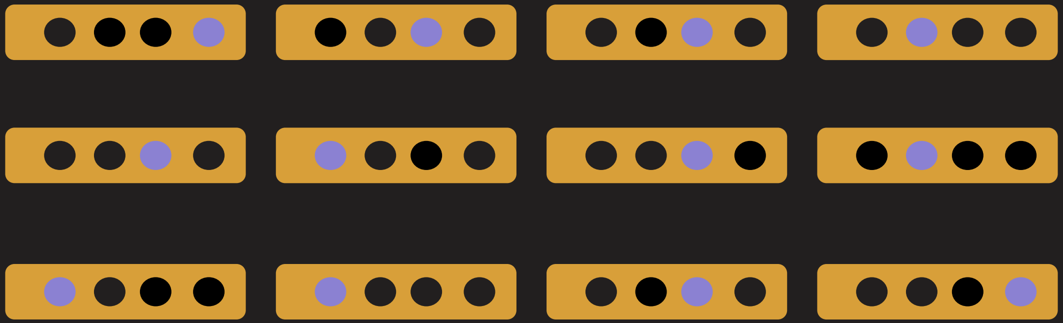
“Random” assignments in a cluster



About 65% utilized

Cluster partially used as jobs finish...

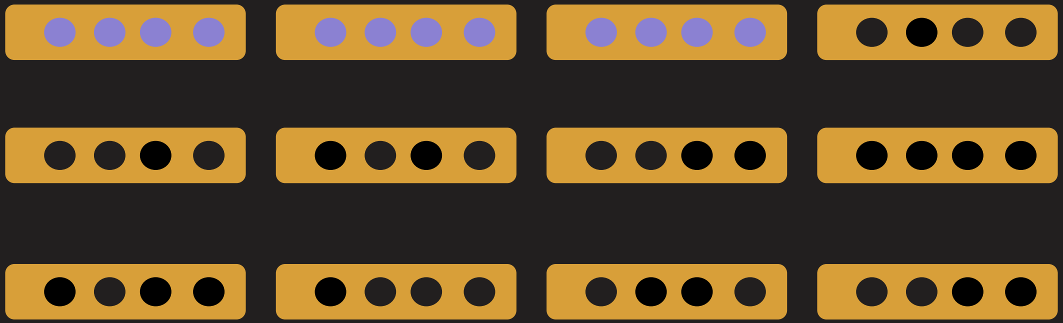
“Random” assignments in a cluster



About 25% utilized

Cluster partially used, but, can’t terminate any instance without losing jobs

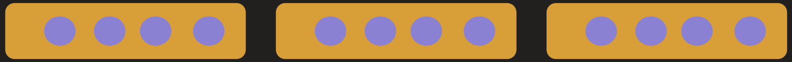
Ideal assignments in a cluster



Similarly,
25% utilized

Cluster utilized to the same level as previous, but, can now terminate 9 of the 12 instances!

Ideal assignments in a cluster



Cluster scaled down easily due to “bin packing”

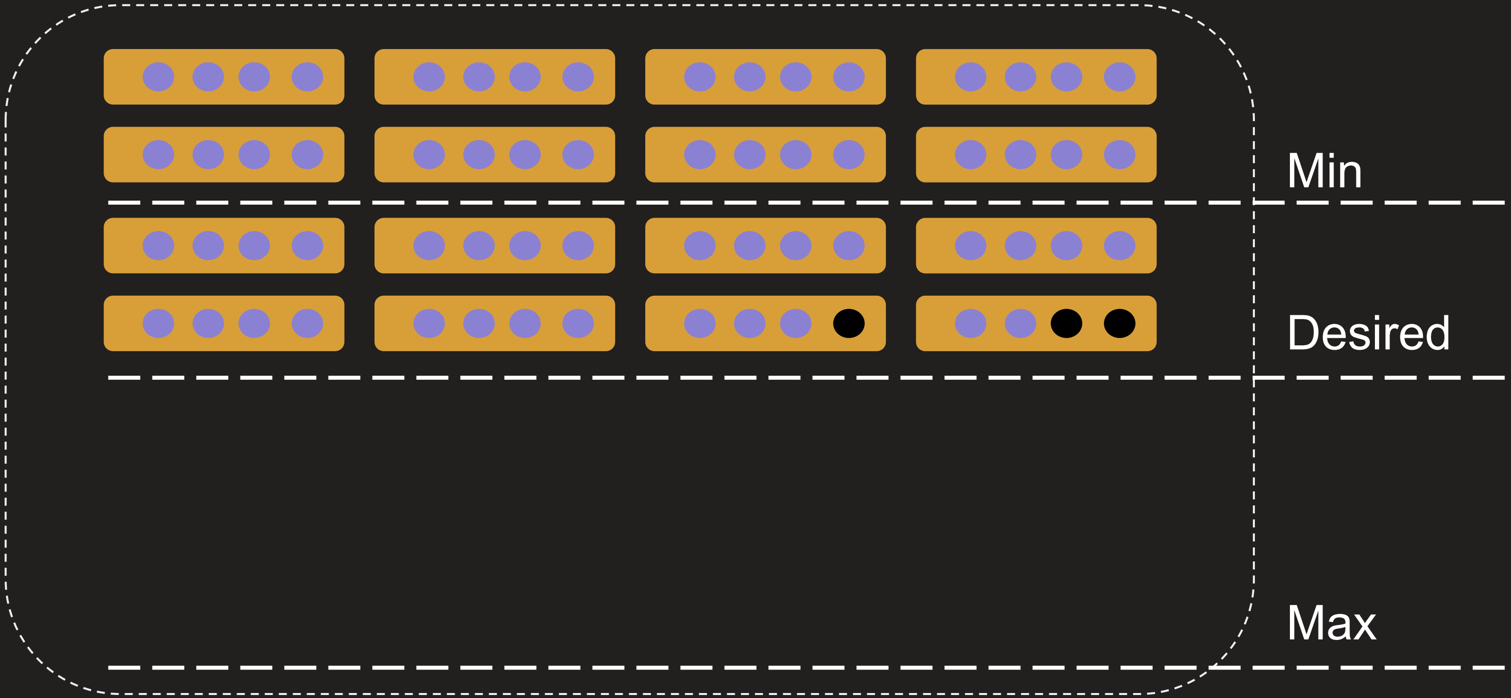
EC2 ASG attributes for setting number of servers in cluster

EC2 AutoScalingGroups have three attributes to set

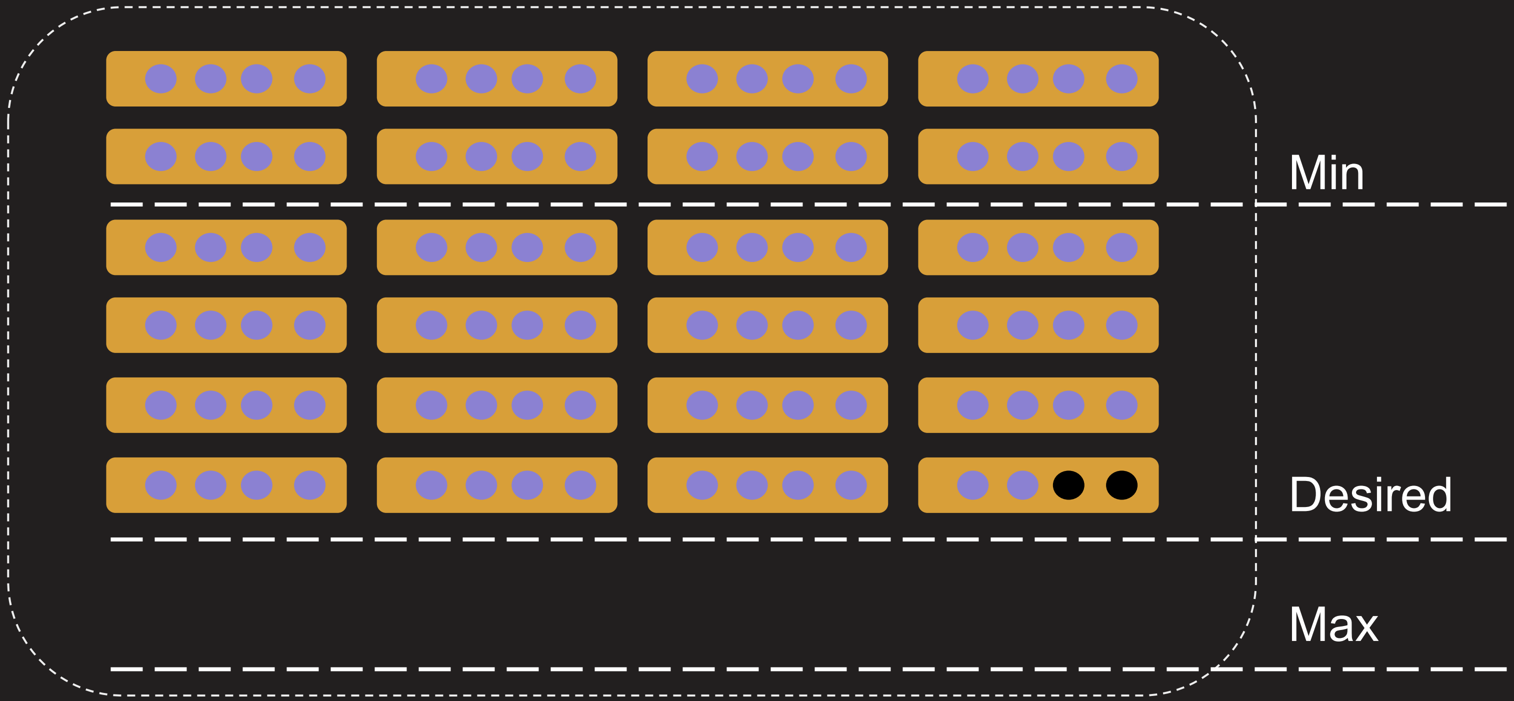
- Min - minimum number of instances to have
- Max - maximum number of instances
- **Desired** - current number of instances to have

Fenzo sets the “Desired” count based on demand

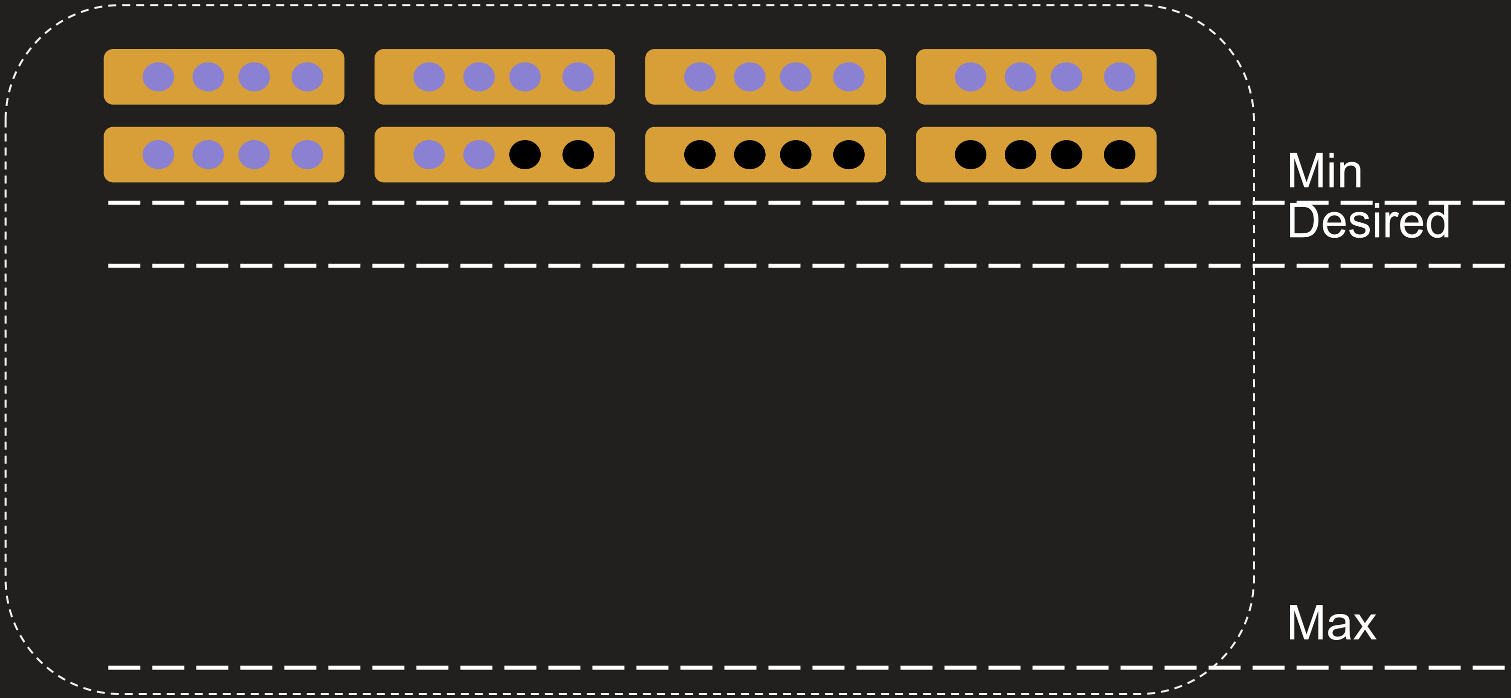
EC2 AutoScalingGroup for Mesos agents



EC2 AutoScalingGroup for Mesos agents



EC2 AutoScalingGroup for Mesos agents



Using multiple instance types

Using multiple instance types

Amazon EC2 provides a variety of servers

- a.k.a “instance types”
- https://aws.amazon.com/ec2/instance-types/

Algorithm model training jobs run well on memory optimized instances of R3 type

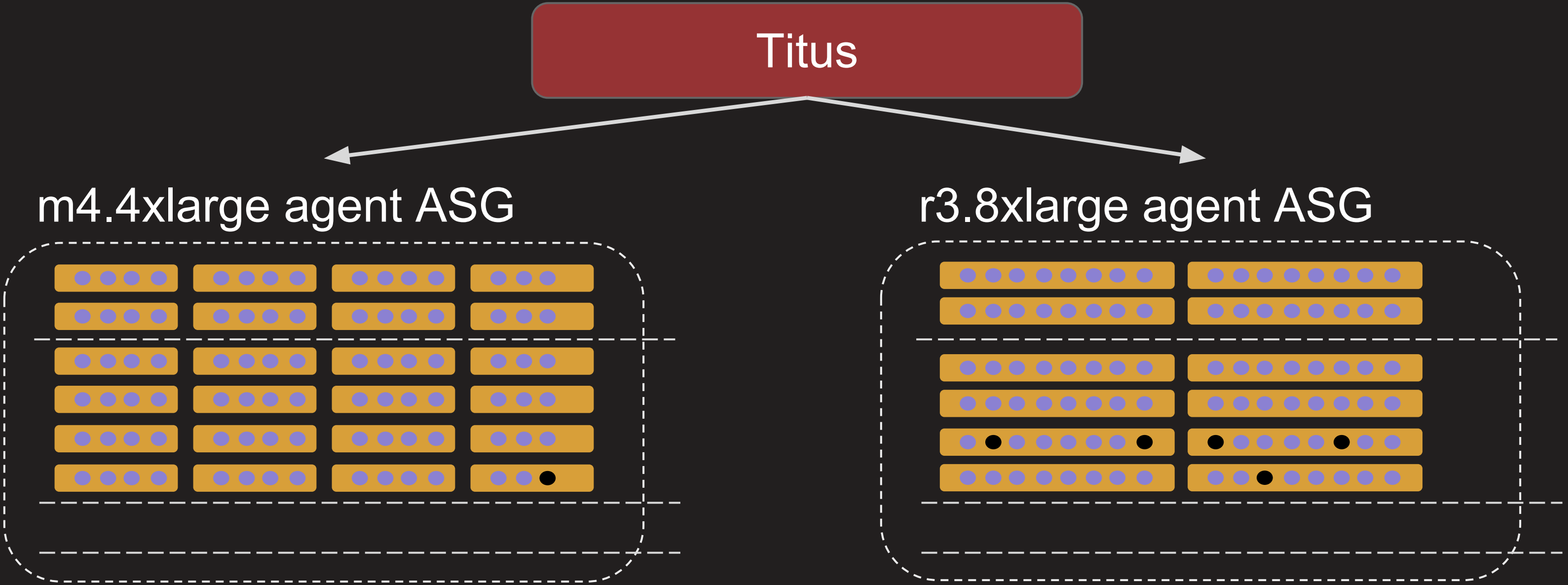
Typical services run well on balanced compute instances of M4 type

Using multiple instance types

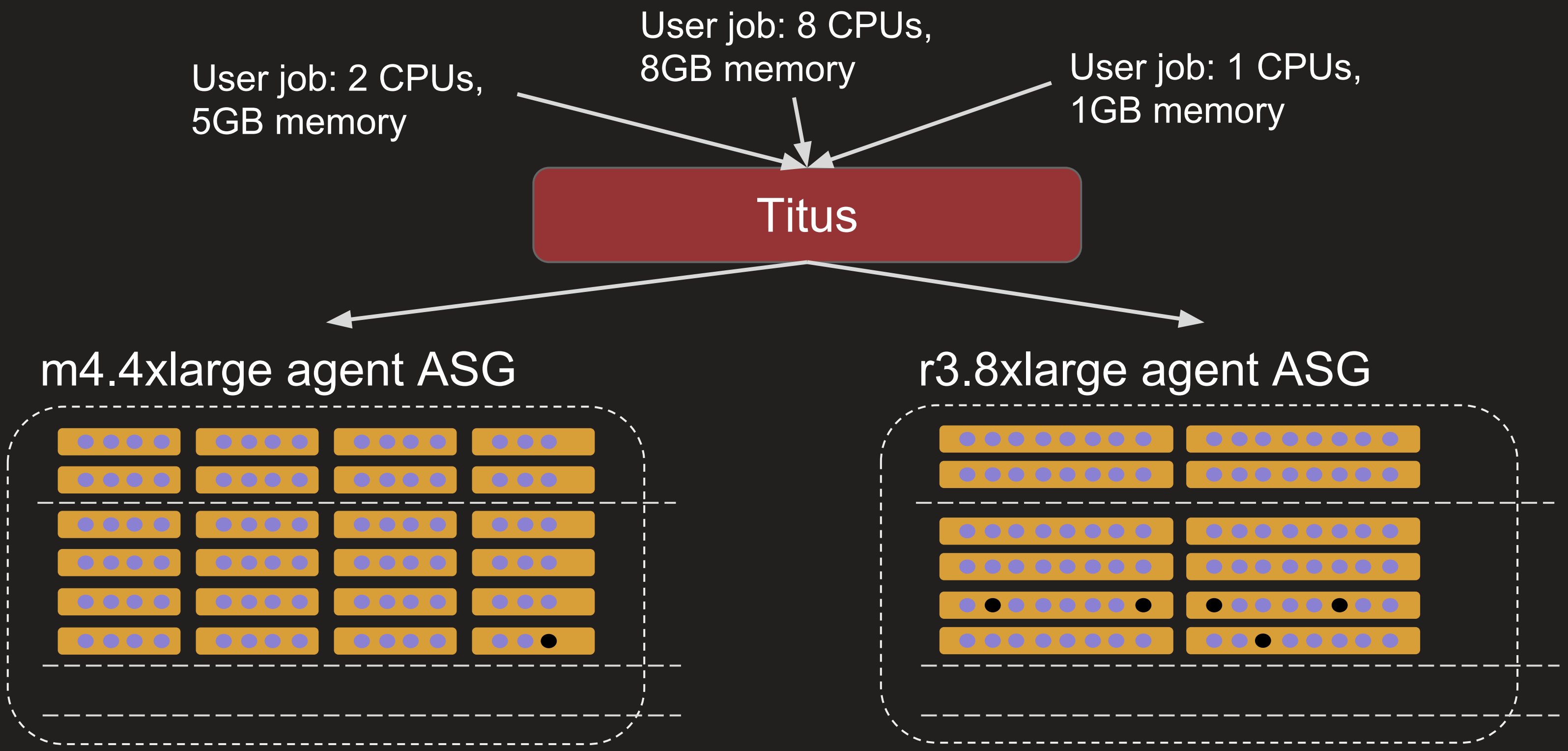
How do we use multiple EC2 instance types in the same Mesos agent cluster?

Using multiple EC2 instance types

Grouping agents by instance type let’s us autoscale them independently



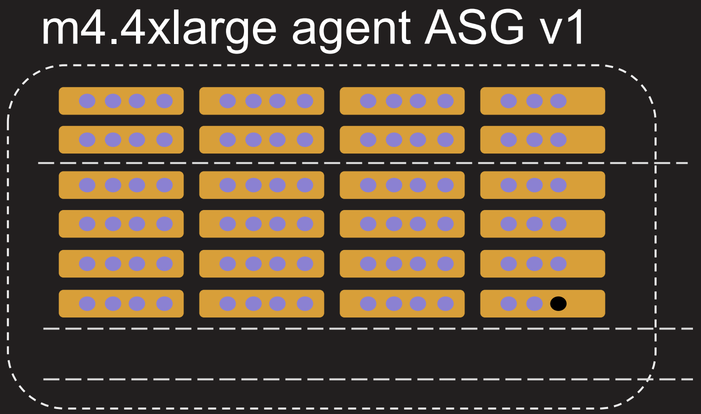
Using multiple EC2 instance types


User job: 2 CPUs, 5GB memory
User job: 8 CPUs, 8GB memory
User job: 1 CPUs, 1GB memory
Titus
m4.4xlarge agent ASG
r3.8xlarge agent ASG

Continuous deployment of agents

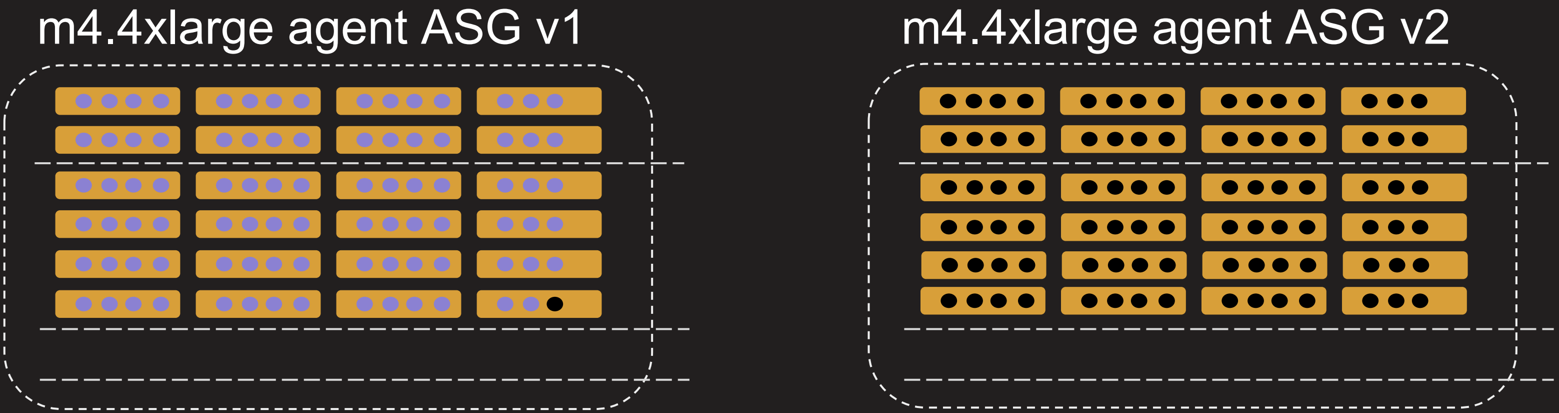
Continuous deployment of agents

A new version of agent introduces a new ASG



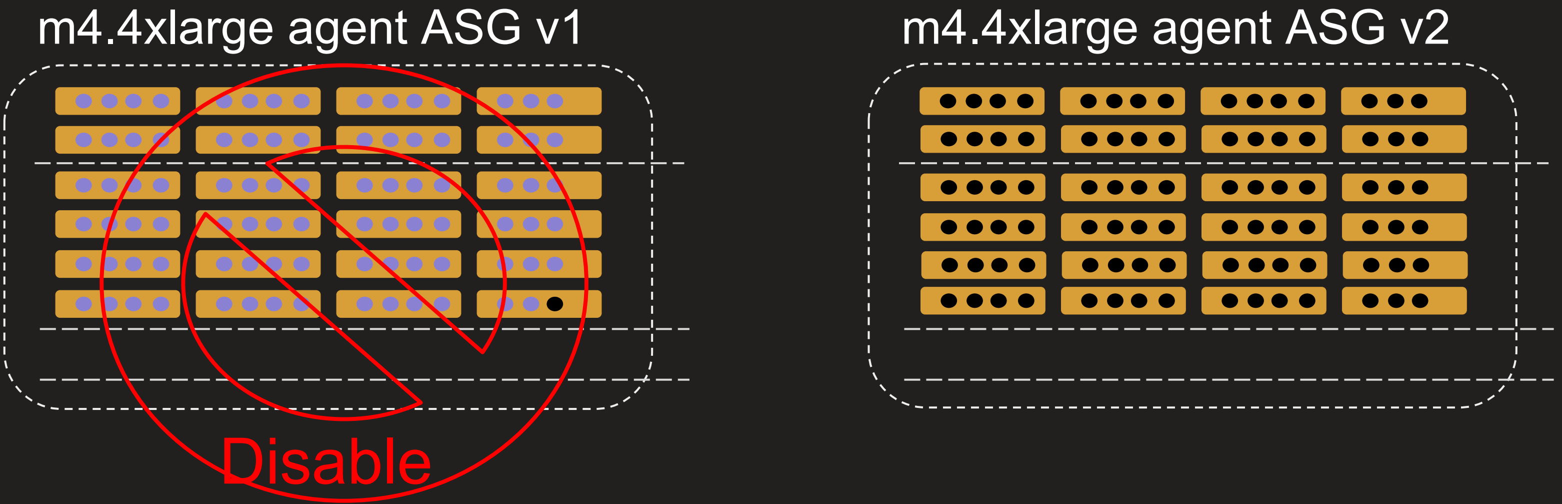
Continuous deployment of agents

A new version of agent introduces a new ASG



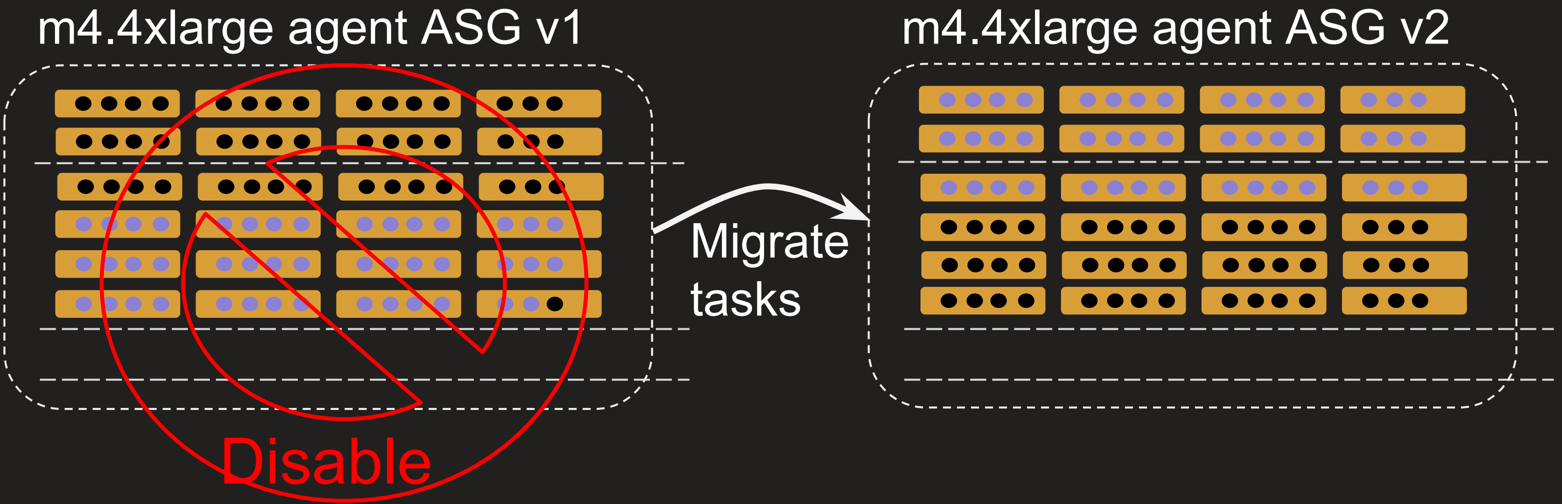
Continuous deployment of agents

A new version of agent introduces a new ASG



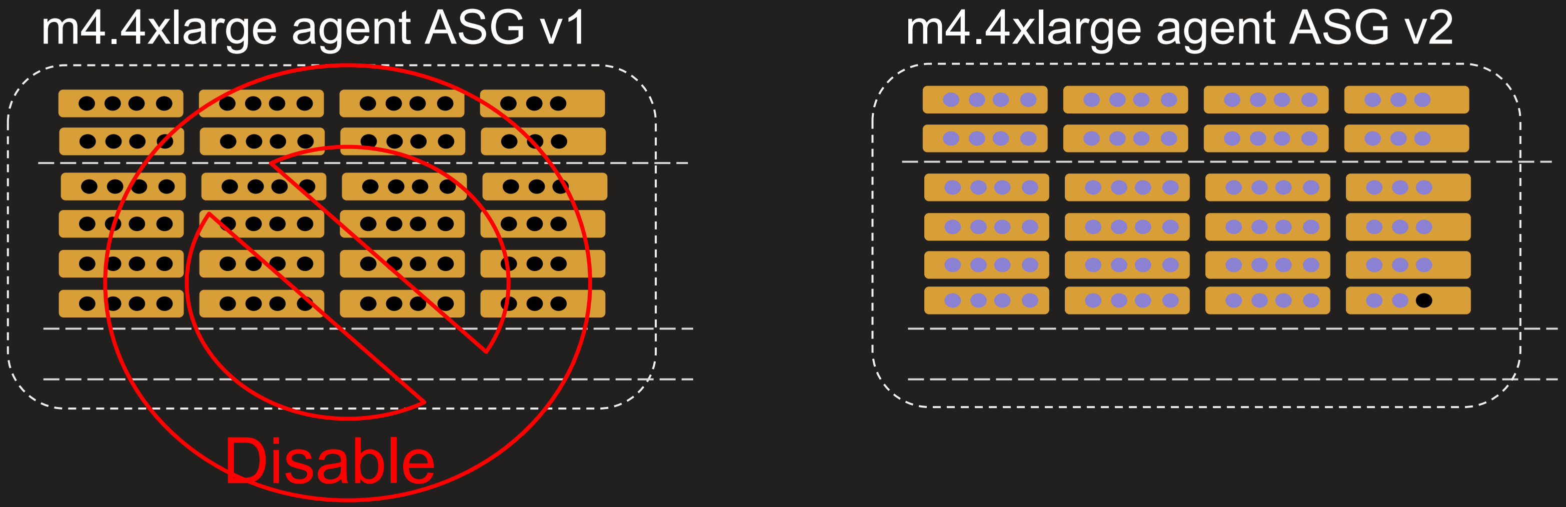
Continuous deployment of agents

A new version of agent introduces a new ASG



Continuous deployment of agents

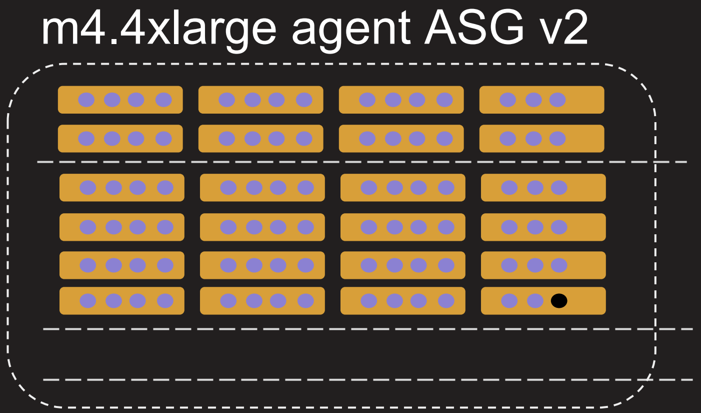
A new version of agent introduces a new ASG



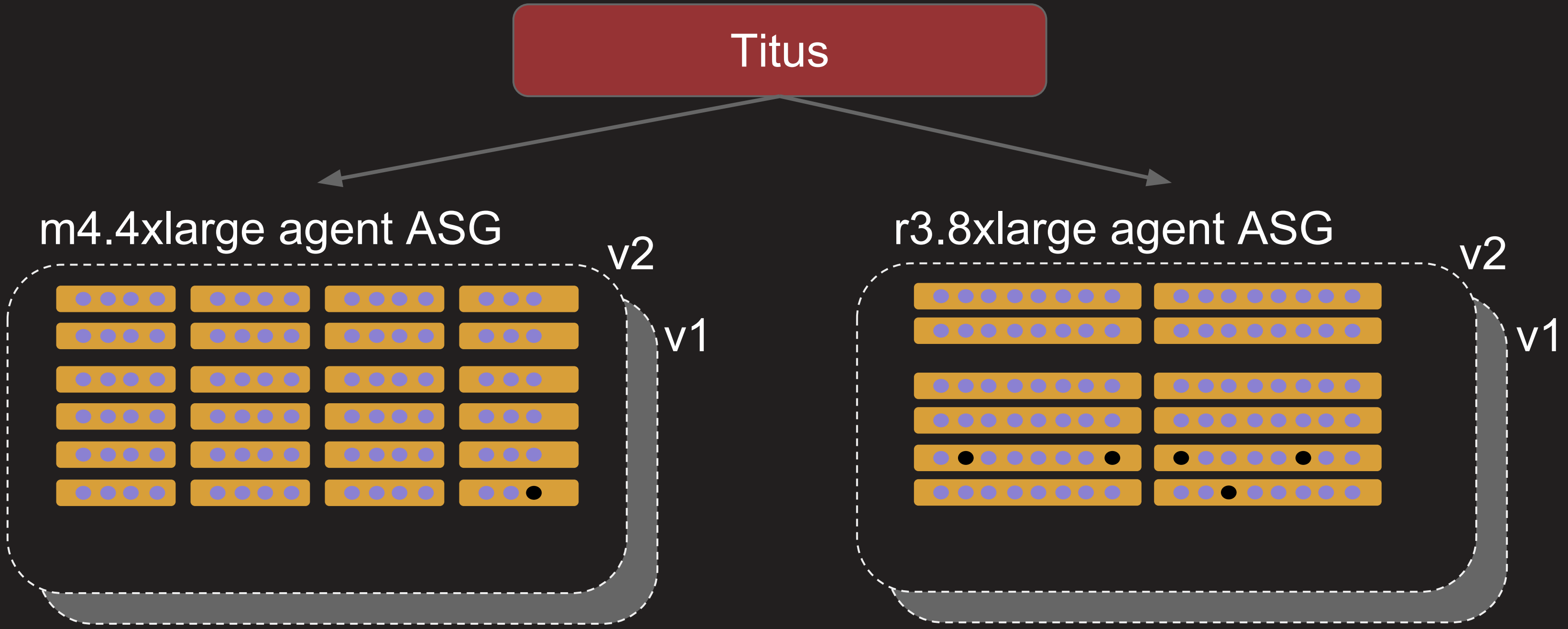
Continuous deployment of agents

A new version of agent introduces a new ASG

Old agent
ASG removed



Bringing it all together...



Capacity guarantees for varied applications

The capacity guarantee challenge

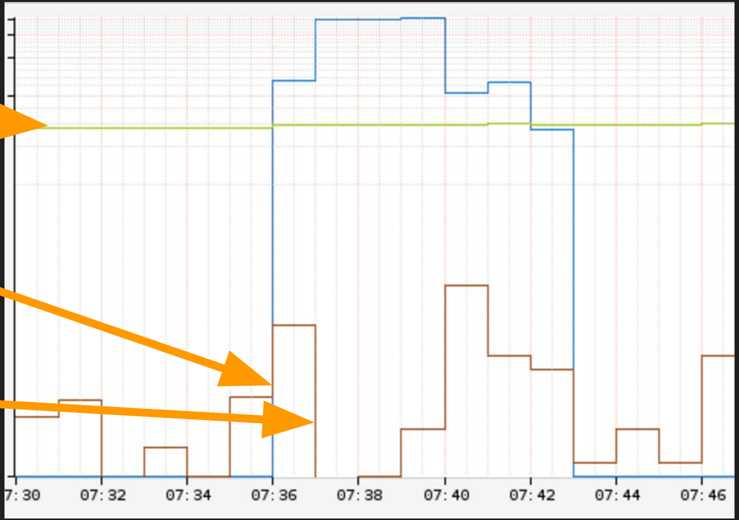
Demand for resources > Supply

An execution sample from a cluster

Running #tasks

New batch of tasks

Tasks launched

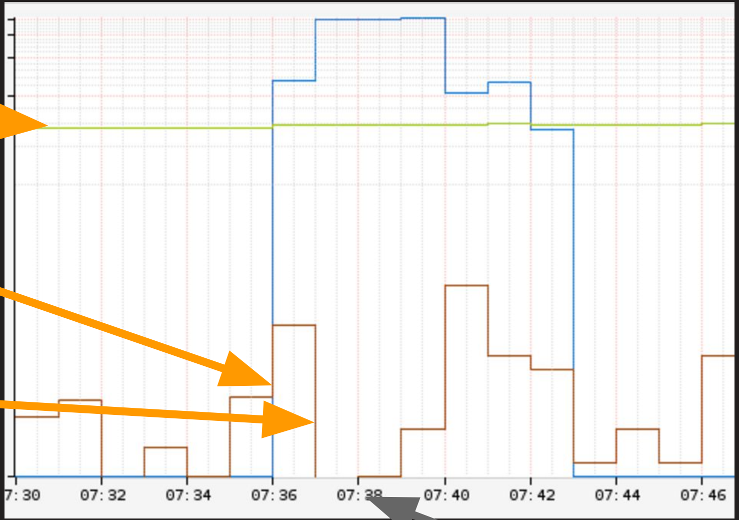


An execution sample from a cluster

Running #tasks

New batch of tasks

Tasks launched



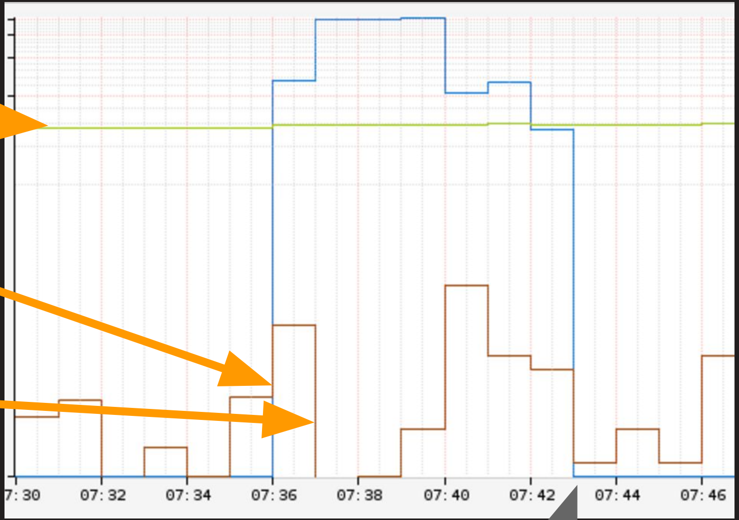
Waiting for agents to free up…
Or, for new agents from scale up

An execution sample from a cluster

Running #tasks

New batch of tasks

Tasks launched



Scale up and freed agents satisfy all new pending tasks

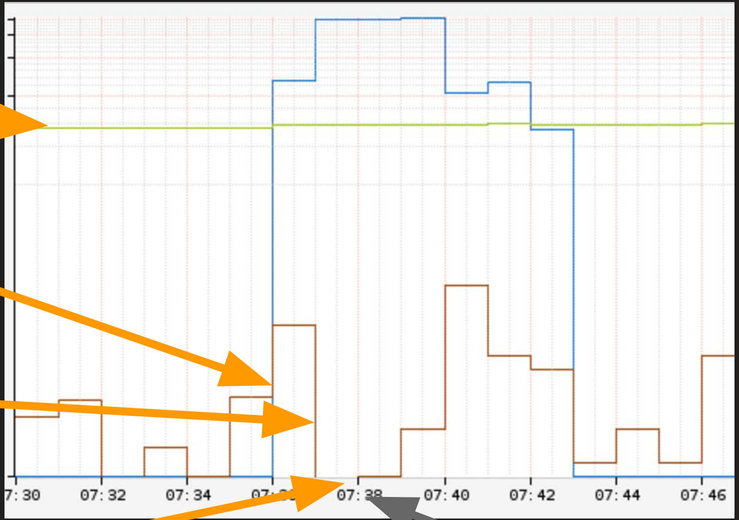
An execution sample from a cluster

Running #tasks

New batch of tasks

Tasks launched

What if a service was launched at this time?



Waiting for agents to free up…
Or, new agents from scale up

Capacity guarantees

Guarantee $\overset{\text{Agreed upon}}{\wedge}$ capacity for timely job starts

- Mesos support for quotas, etc. evolving

Capacity guarantees

Guarantee $\overset{\text{Agreed upon}}{\wedge}$ capacity for timely job starts

Mesos support for quotas, etc. evolving

Generally, optimize throughput for batch jobs and start latency for service jobs

Capacity guarantees

Some service style jobs may be less important

Categorize by expected behavior instead

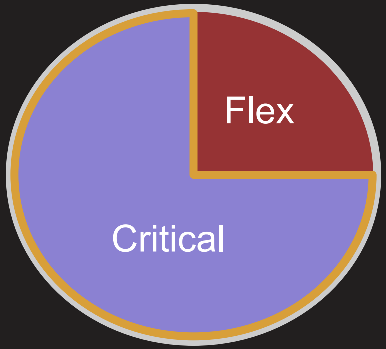
Capacity guarantees

Some service style jobs may be less important

Categorize by expected behavior instead

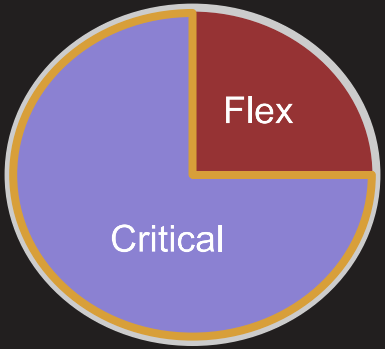
<u>Critical</u> versus <u>Flex</u> (flexible) scheduling requirements

Capacity guarantees



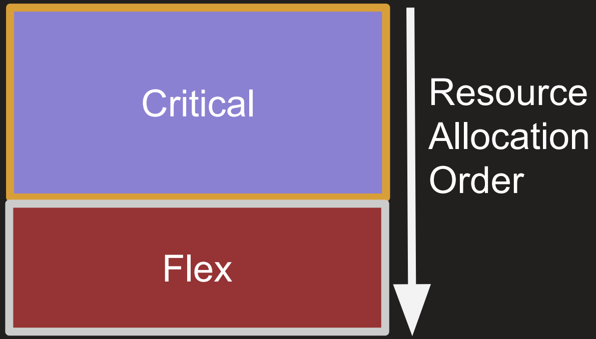
Quotas

Capacity guarantees



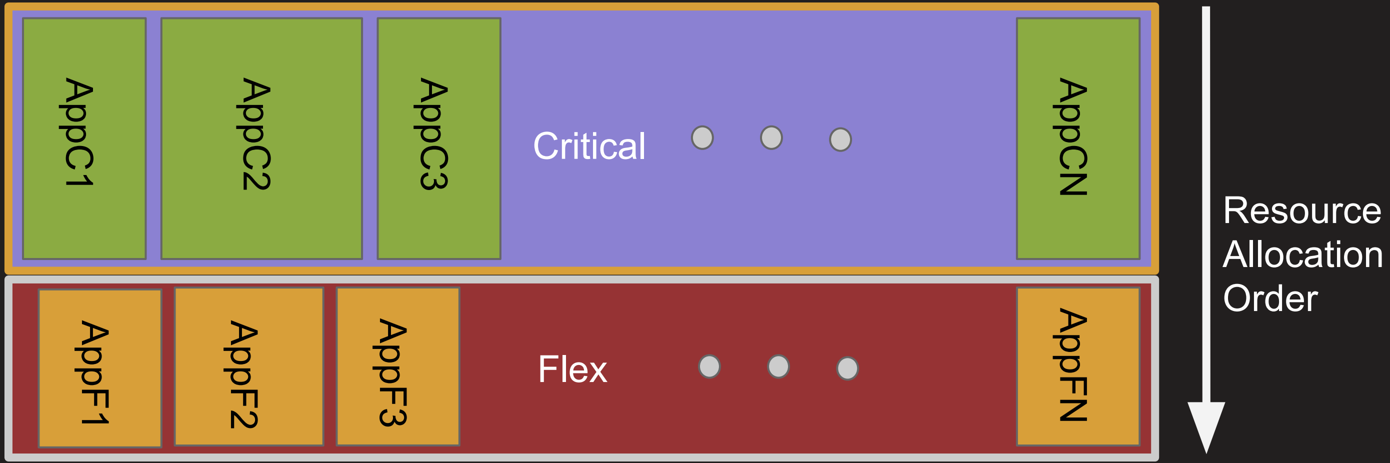
Quotas

vs.



Priorities

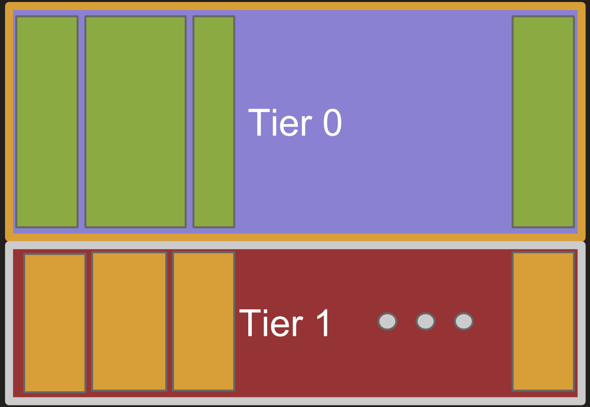
Capacity guarantees: hybrid view



Capacity guarantees via Fenzo

Fenzo supports multi-tiered task queues

Multiple “buckets” per tier with “fair sharing” by dominant resource usage



Translating application capacity to EC2 instances

- Define per application capacity guarantees
- Define per tier capacity guarantees
- Translate to number of EC2 instances

Defining application capacity

App1-cap = num_app_instances *
app_instance_dimensions

app_instance_dimensions:
{ #cpus, memory, disk, network}

Agnostic to EC2 instance types

Defining application capacity

Applications specify resource needs, not EC2 instance types

- Can manage capacity guarantees using a variety of instance types
- Eases migration to new instance types, thereby helps capacity procurement teams

Defining Tier capacity

Tier Capacity =

SUM (App1-cap + App2-cap + … + AppN-cap)

+ BUFFER

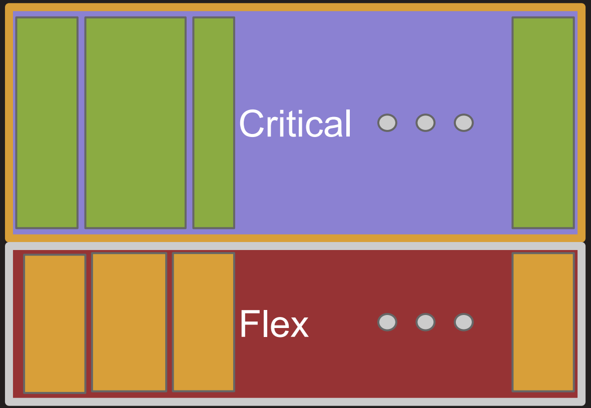
BUFFER:

- Accommodate some new or ad hoc jobs with no guarantees
- Red-black pushes of services temporarily double capacity

Translate to number of instances

#EC2_instances = Tier_capacity / EC2_instance_dimensions

A tier may use multiple instance types

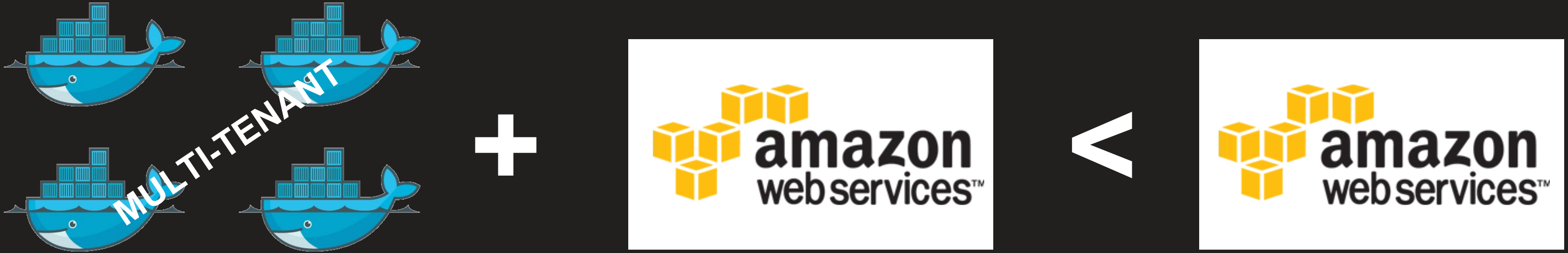


= { m4.4xlarge, m3.2xlarge }

= { r3.8xlarge, g2.8xlarge }

Network resource and security groups

Container executor

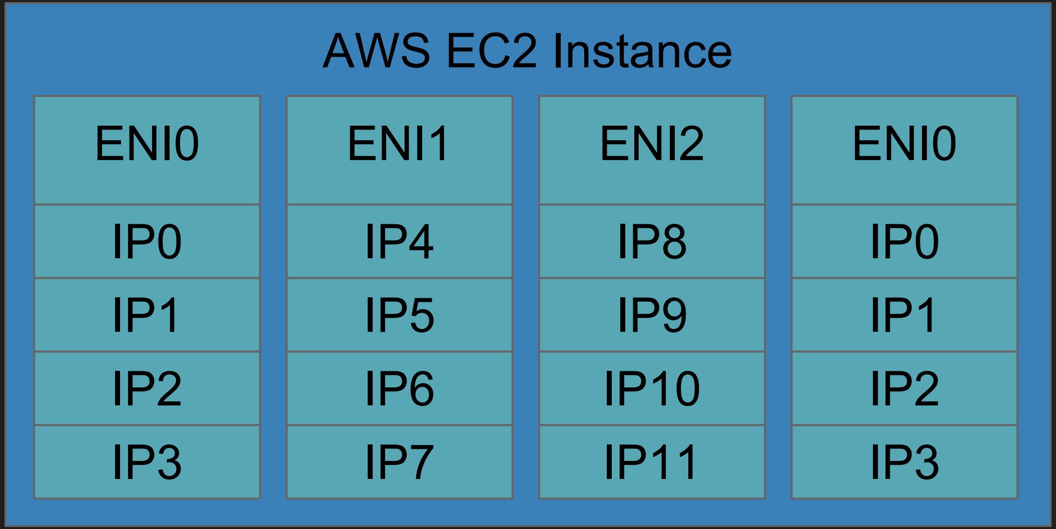


Augment missing pieces:

- IP per container
- Security - Security Groups, IAM roles
- Isolation for networking b/w, disk I/O

Elastic Network Interfaces (ENI)

- Each EC2 instance in VPC has 2 or more ENIs
- Each ENI can have 2 or more IPs
- Security Groups are set on the ENI



ENI+IP resource allocation model

A two level resource modeled in Fenzo
Each agent reports #ENIs and #IPs per ENI via custom attribute
Fenzo does allocation and usage tracking

ENI 1

Assigned Security Group: SG1	Used IPs Count: 2 of 7

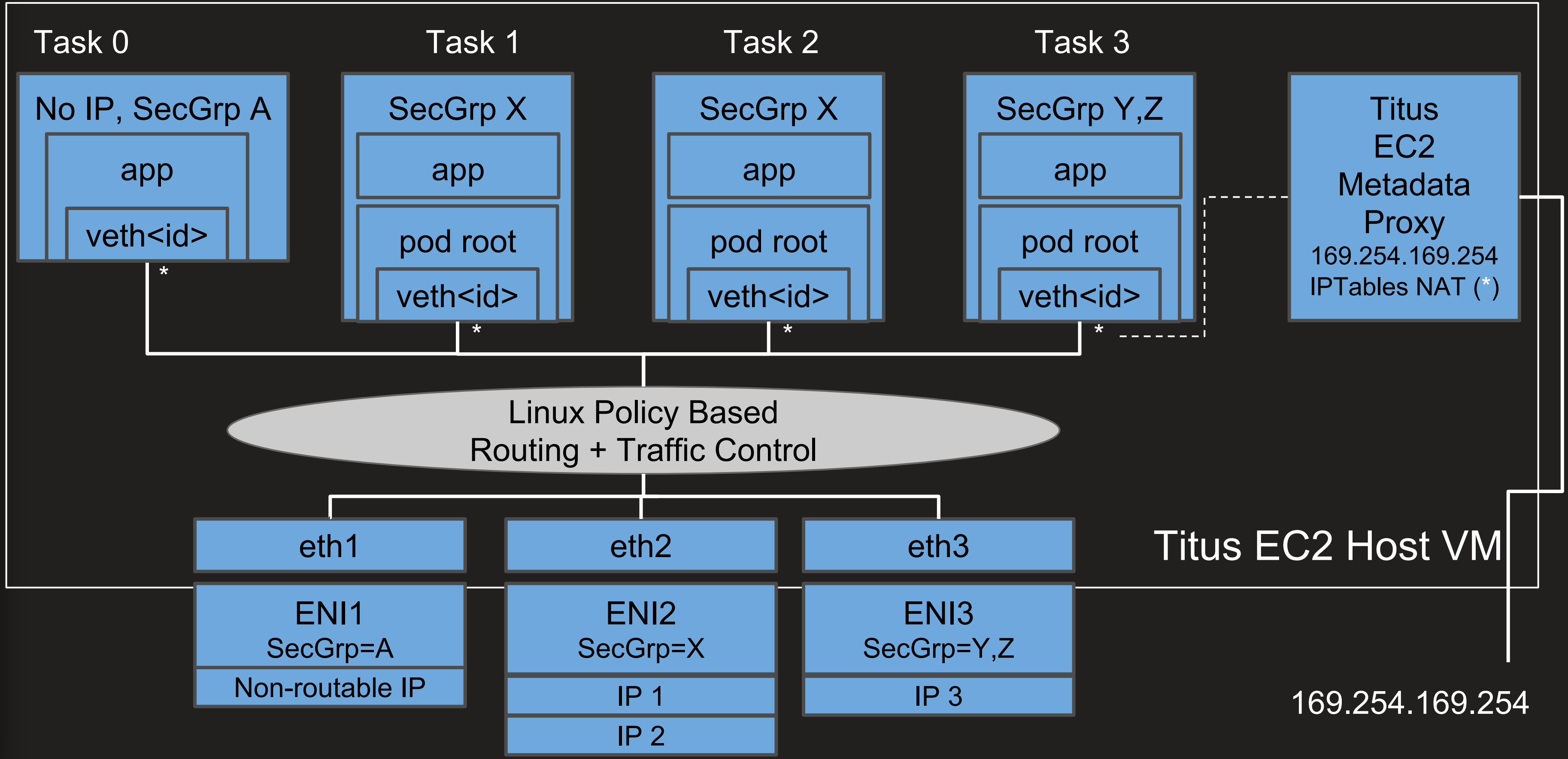
ENI 2

Assigned Security Group: SG1,SG2	Used IPs Count: 1 of 7

ENI 3

Assigned Security Group: SG3	Used IPs Count: 7 of 7

Plumbing VPC Networking into Docker



Network bandwidth isolation

Each container gets an IP on one of the ENIs

Linux tc policies used on virtual Ethernet

- For both incoming and outgoing traffic

Bandwidth limited to the requested value

- No borrowing of unused bandwidth
- Easy to reason about

Ongoing and future work

Current and future work

- Fine grain capacity guarantees
  - Hierarchical sharing policies
  - Preemptions to satisfy priority tiers and sharing policies
- Execution environment security hardening
- Onboarding new applications
- Looking forward to working with the community

In Summary...

In summary...

Mesos and Fenzo help us run lots of containers

- In an elastic fashion
- With guaranteed capacity for varied applications
- Custom AWS integration gives us network resource isolation and security groups

Questions?

Elastic Efficient Execution of Varied Containers

Sharma Podila

spodila @ netflix . com

@podila

linkedin . com / in / spodila

NETFLIX