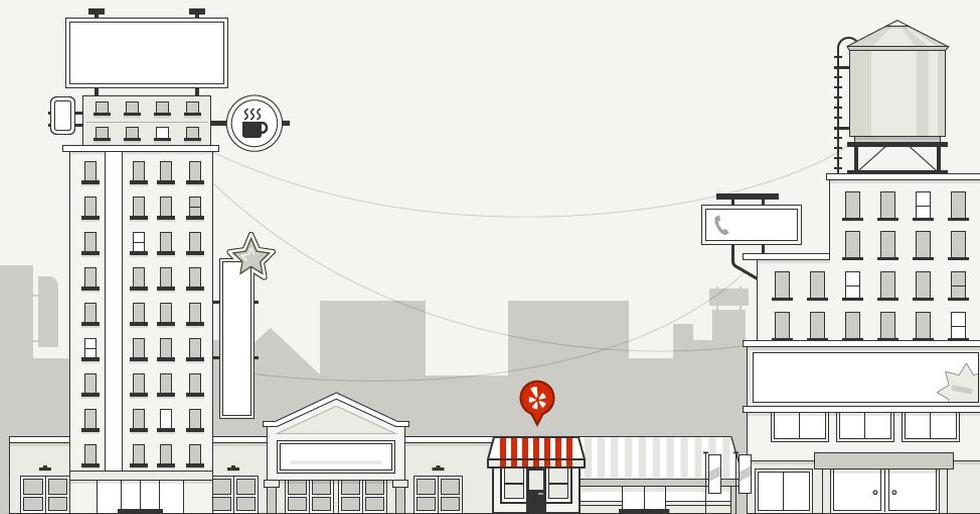


How Yelp.com Runs on Apache Mesos in AWS Spot Fleet for Fun and Profit (75% Off)

Kyle Anderson - Yelp



Yelp's Mission

Connecting people with great local businesses.



Part 0: Spot / Spot Fleet Primer

Part 1: Spot Fleet management + Mesos

Part 2: Spot Fleet “Best Practices” (Fun)

Part 3: Graph all the things (Profit)



0: AWS Spot / Spot Fleet Primer



Definitions

- EC2
- EC2 Instances
- Instance types (m4.4xlarge, c4.8xlarge, r4.16xlarge)
- Availability zones (AZs, us-west-1a, us-east-1b)
- Reserved instances (RI)
- On Demand Price
- Spot Instance
- Spot Fleet Request (SFR)
- Autoscaling Group (ASG)



Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Fulfilled
\$3.00 (C)	Fulfilled
\$3.00 (C)	Waiting
\$2.00 (D)	Waiting
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$3

Servers (Supply)

Customer	Status
1	In use
1	In use
2	In use
2	In use
3	Available
3	Available
4	Available



Simulation

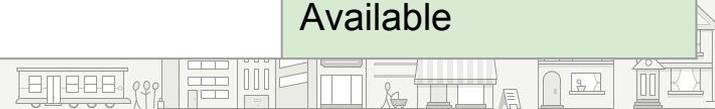
On Demand price:	\$6
Current Spot Price:	\$3

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Fulfilled
\$3.00 (C)	Fulfilled
\$3.00 (C)	Fulfilled!
\$2.00 (D)	Waiting
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

Servers (Supply)

Customer	Status
1	In use
1	In use
2	In use
2	Available (stopped)
3	Available
3	Available
4	Available



Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Fulfilled
\$3.00 (C)	Fulfilled
\$3.00 (C)	Fulfilled
\$2.00 (D)	Waiting
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$3

Servers (Supply)

Customer	Status
1	In use
1	In use
2	In use
2	Available
3	Available
3	Available
4	Available



Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Fulfilled
\$3.00 (C)	Fulfilled
\$3.00 (C)	Fulfilled
\$2.00 (D)	Fulfilled!
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$2

Servers (Supply)

Customer	Status
1	In use
1	In use
2	Available! (stopped)
2	Available
3	Available
3	Available
4	Available



Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Fulfilled
\$3.00 (C)	Fulfilled
\$3.00 (C)	Fulfilled
\$2.00 (D)	Fulfilled
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$2

Servers (Supply)

Customer	Status
1	In use
1	In use
2	Available
2	Available
3	Available
3	Available
4	Available



Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Outbid!
\$3.00 (C)	Outbid!
\$3.00 (C)	Outbid!
\$2.00 (D)	Outbid!
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$5

Servers (Supply)

Customer	Status
1	In use
1	In use
2	In use! (launched!)
2	In use! (launched!)
3	In use! (launched!)
3	In use! (launched!)
4	Available



Spot Simulation

Spot Bids (Demand)

Price (Customer)	Status
\$5.00 (A)	Fulfilled
\$4.00 (B)	Waiting
\$3.00 (C)	Waiting
\$3.00 (C)	Waiting
\$2.00 (D)	Waiting
\$1.00 (E)	Waiting
\$1.00 (E)	Waiting

On Demand price:	\$6
Current Spot Price:	\$5

Servers (Supply)

Customer	Status
1	In use
1	In use
2	In use
2	In use
3	In use
3	In use
4	Available



Spot **Instances** Summary

Spot Instances:

- The spot price == last fulfilled bid price
- Demand fluctuates with spot bidders,
- Supply fluctuates with reserved instance capacity
- Spot customers pay to the hour, rounded up if they terminate, rounded down if AWS terminates



Spot Fleet Simulation

SFR: 15 cpus

Zone A	Zone B	Zone C
1xl	1xl	1xl
4xl	4xl	4xl



Spot Fleet Simulation

SFR: 15 cpus

Zone A	Zone B	Zone C
1xl	1xl	1xl
4xl	4xl OUTBID	4xl
2xl	2xl	



Spot Fleet Simulation

SFR: 15 cpus

Zone A	Zone B	Zone C
4x OUTBIT	4x -OUTBID	4x -OUTBID
4xl	4x OUTBID	4xl
2xl	2xl	2xl
		2xl



Spot Fleet Summary

- Control system for launching spot instances en-mass and maintaining capacity
- Users dictate the acceptable composition and bid price for each type of server with weighting (Spot Fleet Request, SFR)
- Spot fleet responds to outbid events and launches replacement spot instances



How To Manage Spot Fleets



Select request type

- Request type**
-  **Request**
Submit a one-time Spot instance request
 -  **Request and Maintain**
Request a fleet of Spot instances to maintain your target capacity
 -  **Reserve for duration**
Request a Spot instance with no interruption for 1 to 6 hours (a Spot block)

Target capacity ⓘ

instances ▾

AMI ⓘ

Search for AMI

Instance type(s) ⓘ

c3.large (2 vCPU, 3.75 GiB, 2 x 16) ✕

Select

Select multiple instance types to find the lowest priced instances available

Allocation strategy ⓘ

Lowest price

Automatically select the cheapest Availability Zone and instance type

Diversified

Balance Spot instances across selected Availability Zones and instance types

Network ⓘ



Create new VPC

Availability Zone ⓘ



Create new subnet

us-west-2a

us-west-2b

us-west-2c

Maximum price ⓘ

Use automated bidding (recommended)

Provision Spot instances at the current Spot price capped at the On-Demand price

Set your max price (per instance/hour)



How (NOT) to Manage Spot Fleets

```
{
  "AllocationStrategy": "lowestPrice|"diversified",
  "ClientToken": "string",
  "ExcessCapacityTerminationPolicy": "noTermination|"default",
  "FulfilledCapacity": double,
  "IamFleetRole": "string",
  "LaunchSpecifications": [
    {
      "SecurityGroups": [
        {
          "GroupName": "string",
          "GroupId": "string"
        }
      ],
      ...
      "AddressingType": "string",
      "BlockDeviceMappings": [
        {
          "DeviceName": "string",
          "VirtualName": "string",
          "Ebs": {
            "Encrypted": true|false,
            "DeleteOnTermination": true|false,
            "Iops": integer,
            "SnapshotId": "string",
            "VolumeSize": integer,
            "VolumeType": "standard|"io1|"gp2|"sc1|"st1"
          },
          "NoDevice": "string"
        }
      ],
      ...
      "EbsOptimized": true|false,
      "IamInstanceProfile": {
        "Arn": "string",
        "Name": "string"
      },
      "ImageId": "string",
      "InstanceType":
      "t1.micro|"t2.nano|"t2.micro|"t2.small|"t2.medium|"t2.large|"t2.xlarge|"t
      2.2xlarge|"m1.small|"m1.medium|"m1.large|"m1.xlarge|"m3.medium|"m3.large|"
      "m3.xlarge|"m3.2xlarge|"m4.large|"m4.xlarge|"m4.2xlarge|"m4.4xlarge|"m4.10
      xlarge|"m4.16xlarge|"m2.xlarge|"m2.2xlarge|"m2.4xlarge|"cr1.8xlarge|"r3.la
      rge|"r3.xlarge|"r3.2xlarge|"r3.4xlarge|"r3.8xlarge|"r4.large|"r4.xlarge|"
      r4.2xlarge|"r4.4xlarge|"r4.8xlarge|"r4.16xlarge|"x1.16xlarge|"x1.32xlarge|"
      "i2.xlarge|"i2.2xlarge|"i2.4xlarge|"i2.8xlarge|"i3.large|"i3.xlarge|"i3.2x
      large|"i3.4xlarge|"i3.8xlarge|"i3.16xlarge|"hi1.4xlarge|"hs1.8xlarge|"c1.m
      edium|"c1.xlarge|"c3.large|"c3.xlarge|"c3.2xlarge|"c3.4xlarge|"c3.8xlarge"
     |"c4.large|"c4.xlarge|"c4.2xlarge|"c4.4xlarge|"c4.8xlarge|"cc1.4xlarge|"cc
      2.8xlarge|"g2.2xlarge|"g2.8xlarge|"g3.4xlarge|"g3.8xlarge|"g3.16xlarge|"cg
      1.4xlarge|"p2.xlarge|"p2.8xlarge|"p2.16xlarge|"d2.xlarge|"d2.2xlarge|"d2.4
```

```
"NetworkInterfaces": [
  {
    "AssociatePublicIpAddress": true|false,
    "DeleteOnTermination": true|false,
    "Description": "string",
    "DeviceIndex": integer,
    "Groups": ["string", ...],
    "Ipv6AddressCount": integer,
    "Ipv6Addresses": [
      {
        "Ipv6Address": "string"
      }
    ],
    ...
    "NetworkInterfaceId": "string",
    "PrivateIpAddress": "string",
    "PrivateIpAddresses": [
      {
        "Primary": true|false,
        "PrivateIpAddress": "string"
      }
    ],
    ...
    "SecondaryPrivateIpAddressCount": integer,
    "SubnetId": "string"
  }
  ...
],
"Placement": {
  "AvailabilityZone": "string",
  "GroupName": "string",
  "Tenancy": "default|"dedicated|"host"
},
"RamdiskId": "string",
"SpotPrice": "string",
"SubnetId": "string",
"UserData": "string",
```

```
"KernelId": "string",
"KeyName": "string",
"Monitoring": {
  "Enabled": true|false
},
"WeightedCapacity": double,
"TagSpecifications": [
  {
    "ResourceType":
    "customer-gateway|"dhcp-options|"image|"instance|"internet-gateway
   |"network-acl|"network-interface|"reserved-instances|"route-table"
   |"snapshot|"spot-instances-request|"subnet|"security-group|"volume
   |"vpc|"vpn-connection|"vpn-gateway",
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
    ],
    ...
  }
  ...
],
"SpotPrice": "string",
"TargetCapacity": integer,
"TerminateInstancesWithExpiration": true|false,
"Type": "request|"maintain",
"ValidFrom": timestamp,
"ValidUntil": timestamp,
"ReplaceUnhealthyInstances": true|false
}
```



How (NOT) to Manage Spot Fleets

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole":
"arn:aws:iam::123456789012:role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "KeyName": "my-key-pair",
      "SecurityGroups": [
        {
          "GroupId": "sg-1a2b3c4d"
        }
      ],
      "InstanceType": "m3.medium",
      "SubnetId": "subnet-1a2b3c4d, subnet-3c4d5e6f",
      "IamInstanceProfile": {
        "Arn":
"arn:aws:iam::123456789012:instance-profile"
      }
    }
  ]
}
```

← 2 what?

← Magic number

← Magic number

← Magic number

← Magic number

← Duplicate number

← Only one instance type == bad

← Very nested, no trailing commas



```
# Request a Spot fleet
resource "aws_spot_fleet_request" "cheap_compute" {
  iam_fleet_role      = "arn:aws:iam::12345678:role/spot-fleet"
  spot_price          = "0.03"
  allocation_strategy = "diversified"
  target_capacity     = 6
  valid_until        = "2019-11-04T20:44:20Z"

  launch_specification {
    instance_type = "m4.10xlarge"
    ami           = "ami-1234"
    spot_price    = "2.793"
    placement_tenancy = "dedicated"
  }

  launch_specification {
    instance_type = "m4.4xlarge"
    ami           = "ami-5678"
    key_name      = "my-key"
    spot_price    = "1.117"
    availability_zone = "us-west-1a"
    subnet_id     = "subnet-1234"
    weighted_capacity = 35

    root_block_device {
      volume_size = "300"
      volume_type = "gp2"
    }
  }
}
```

How to (Better) Manage Spot Fleets

- Terraform (TF) has **variables**, you can document and reuse magic numbers
- TF has a `remote_state` thing, you can lookup other magic numbers for subnets and security groups
- TF doesn't need nesting and has a more forgiving syntax



How to (Best?) Manage Spot Fleets

```
module "mesos-slaves" {  
  source = "git::ssh://git@git/terraform-modules/mesos_spot_cluster"  
  cluster = "mycluster"  
  region = "${var.region}"  
  account = "${var.account}"  
  ecosystem = "${var.ecosystem}"  
  instances_data = "${file("instances_cpu_weighted.json")}"  
  account_id = "${var.account_id}"  
  ephemeralsubnets = "${element(split(",", data.terraform_remote_state.vpc.ephemeralsubnets), 0)}"  
  min_capacity = 3  
  max_capacity = 8  
  ami_type = "paasta-optimized"  
  initial_target_capacity = 3  
  instance_profile = "paasta"  
}
```

- No magic numbers ANYWHERE
- Only the inputs you actually care about (sane defaults)
- Reusable instance_data json



```
{
  "instance_data": [
    {
      "type": "c3.4xlarge",
      "price": "0.956",
      "weight": "0.15"
    },
    {
      "type": "c3.8xlarge",
      "price": "1.912",
      "weight": "0.31"
    },
    {
      "type": "c4.4xlarge",
      "price": "1.049",
      "weight": "0.15"
    },
    {
      "type": "c4.8xlarge",
      "price": "2.098",
      "weight": "0.35"
    },
    {
      "type": "m4.10xlarge",
```

- Adding a new instance type is easy
- TF will recreate the spot fleet if it detects changes
- Duplicate data is reduced to the absolute minimum
- Symlink this json as needed



Best Practices for “Production” SFRs

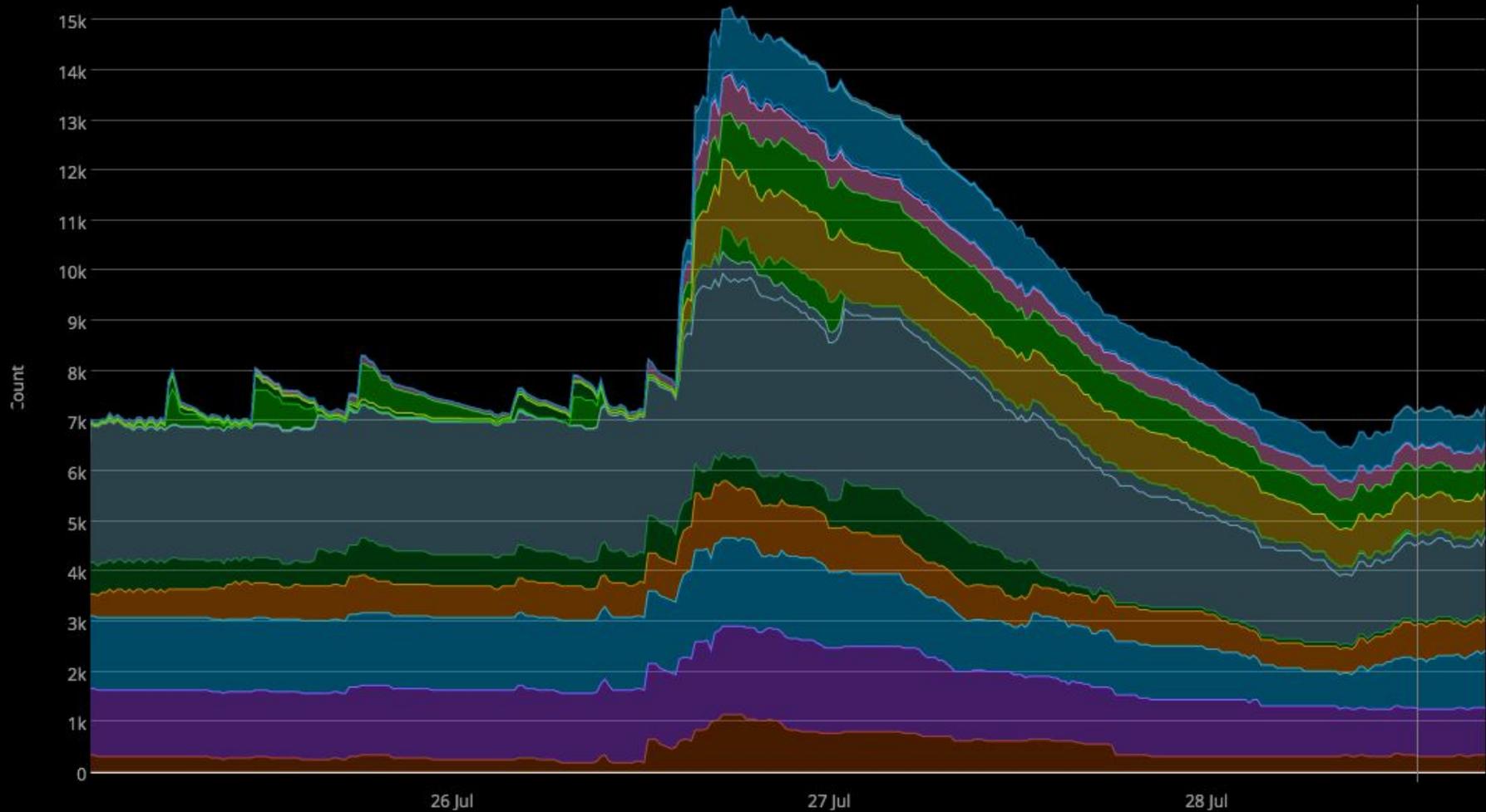
1. Get good at remaking the SFR



Mesos Slaves CPU Count per spot market - kwa

15m

uswest1



Best Practices for “Production” SFRs

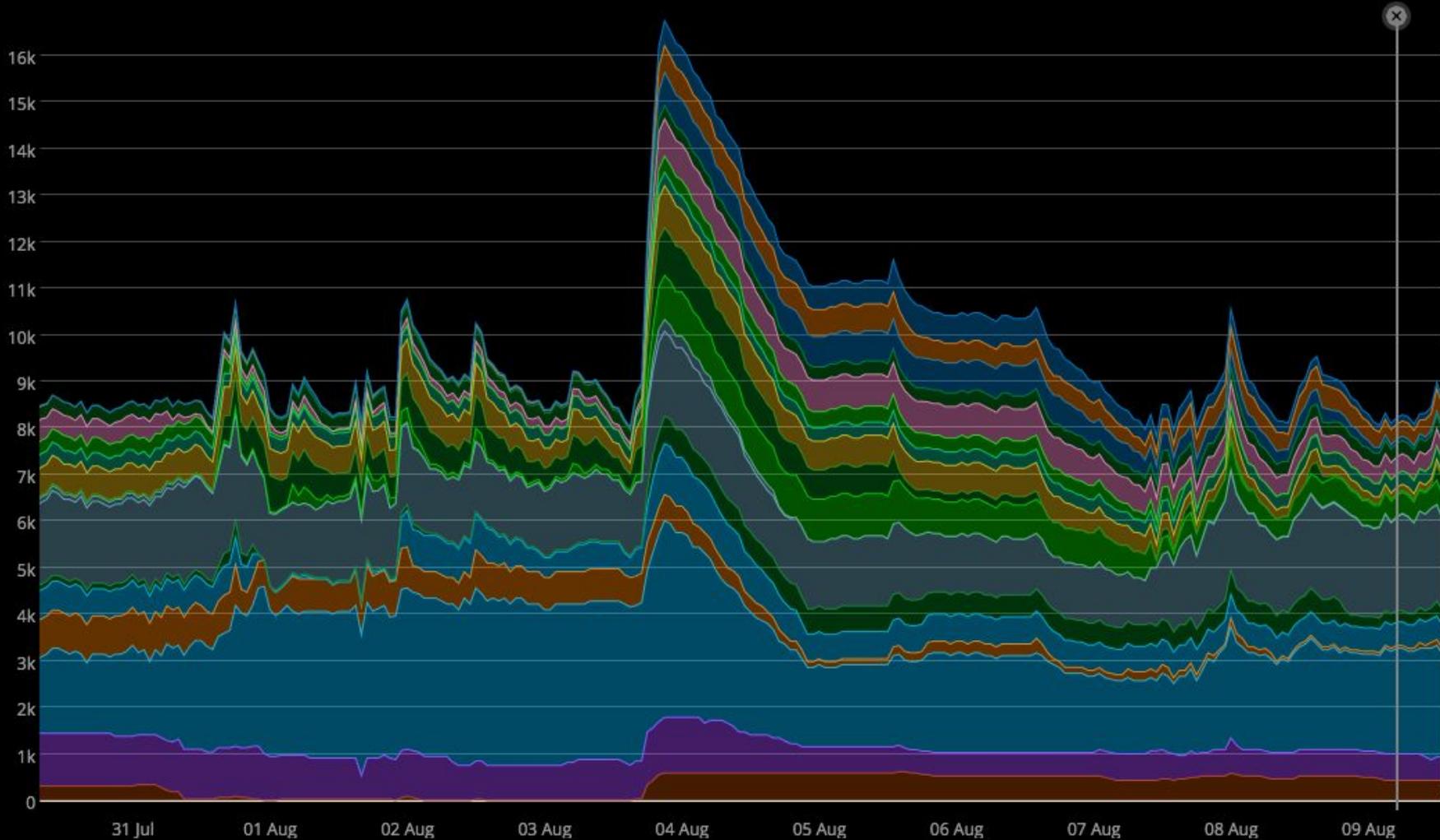
2. Diversify, Diversify, Diversify

- Pick “diversified” over “lowest_price”
- Diversification only can be applied when instances are launched!
- Remember spot markets are az+instance_type
- Weighting is key to keeping capacity up in a diverse fleet



Mesos Slaves per spot market - kva

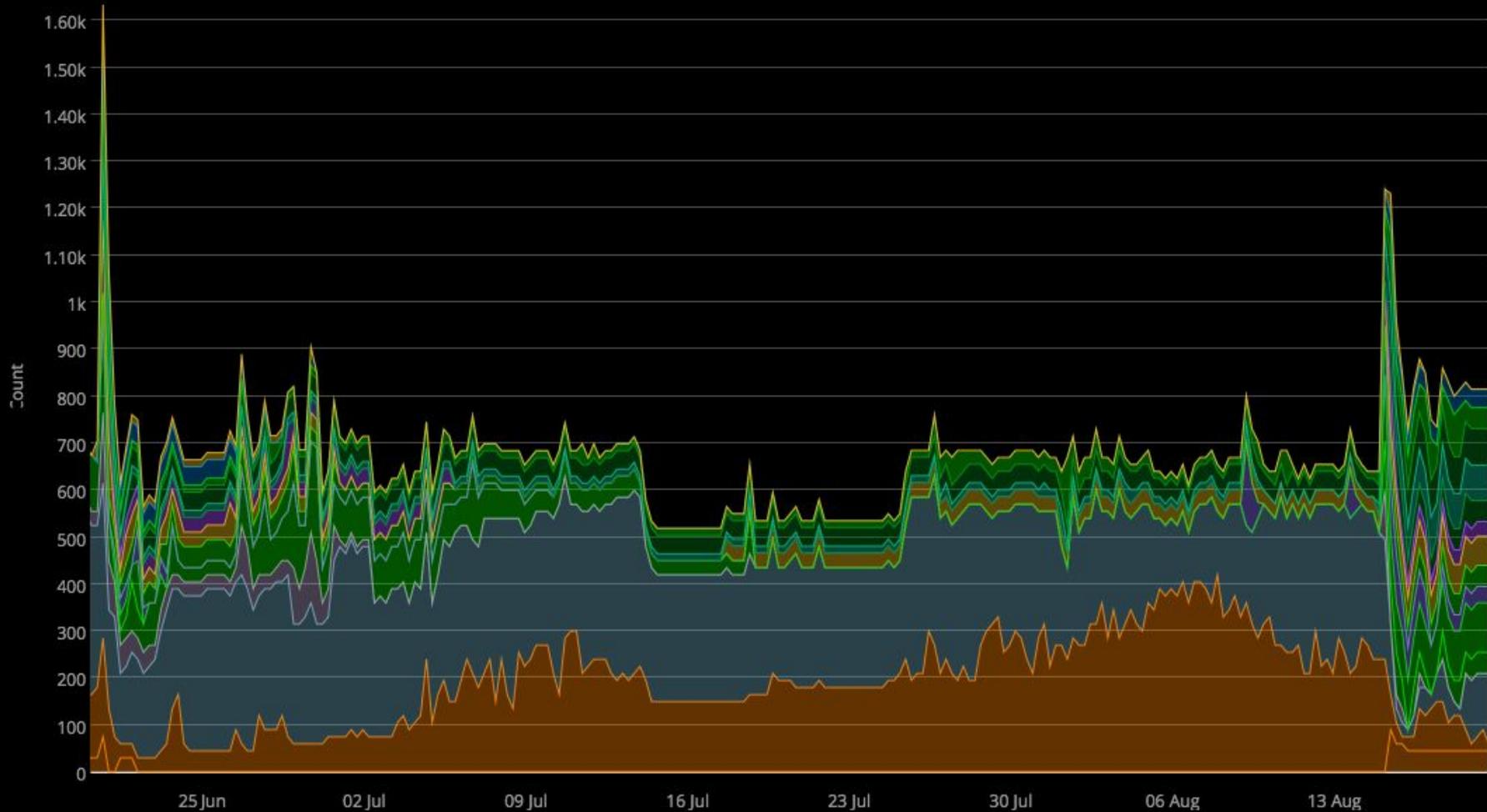
1h



Mesos Slaves CPU Count per spot market - kwa

6h

uswest1



```
{
  "type": "r4.4xlarge",
  "price": "2.128",
  "weight": "0.15"
},
{
  "type": "r4.8xlarge",
  "price": "4.235",
  "weight": "0.31"
},
{
  "type": "r4.16xlarge",
  "price": "8.520",
  "weight": "0.63"
}
```

Best Practices for “Production” SFRs

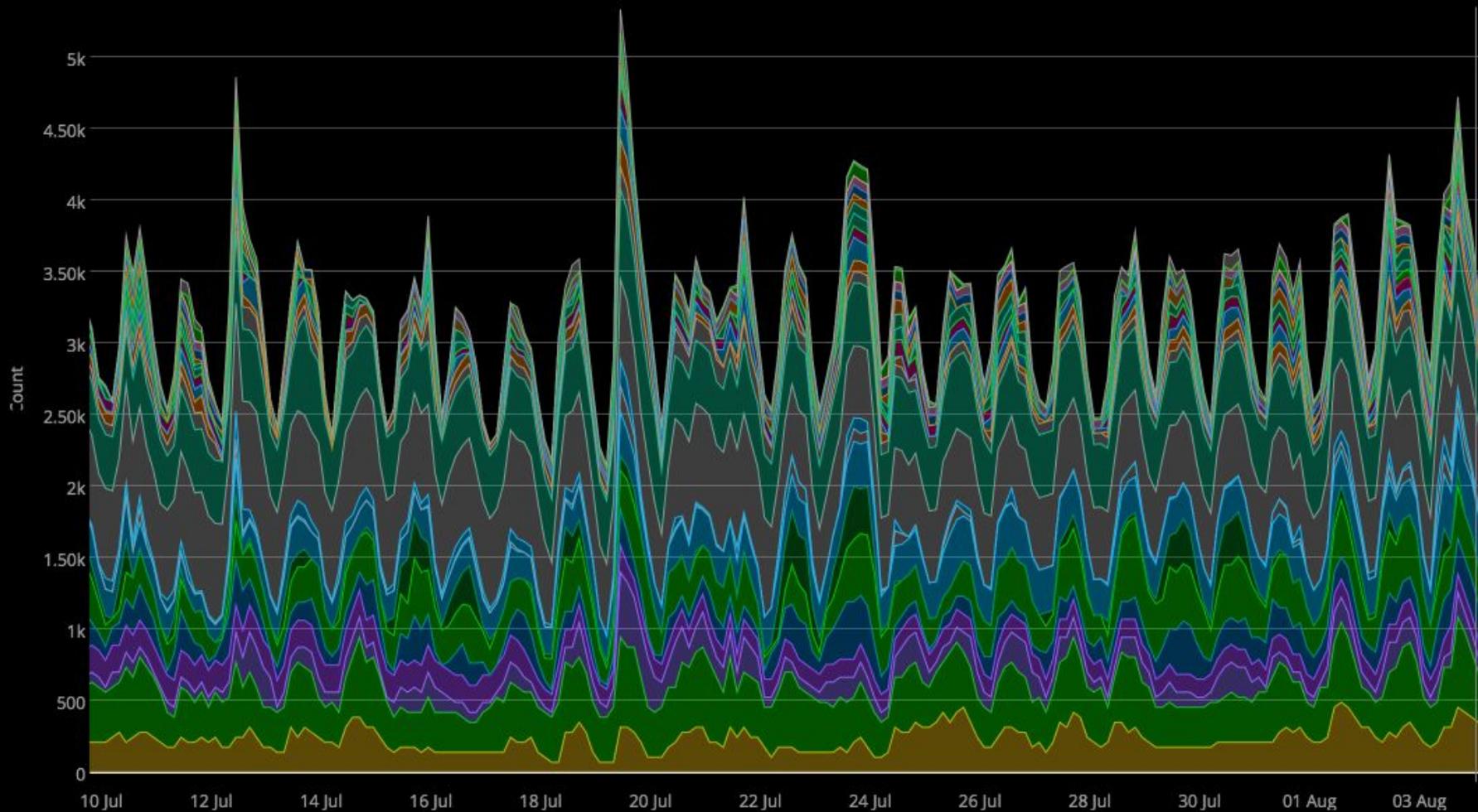
2. Force Availability Zone (AZ) “Balancing”



Mesos Slaves CPU Count per spot market - kwa

3h

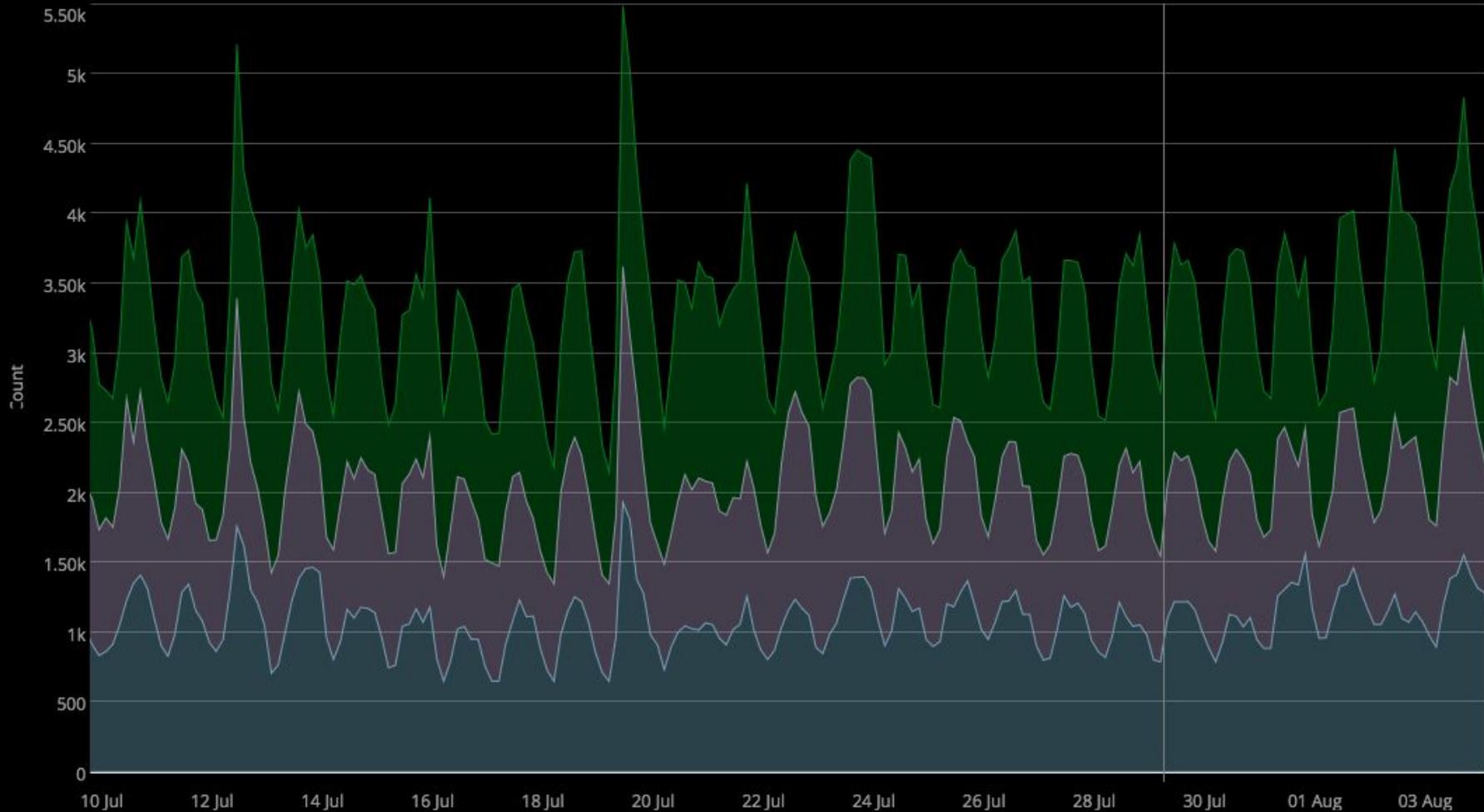
useast1



Mesos Slaves CPU Count per spot market - kwa

3h

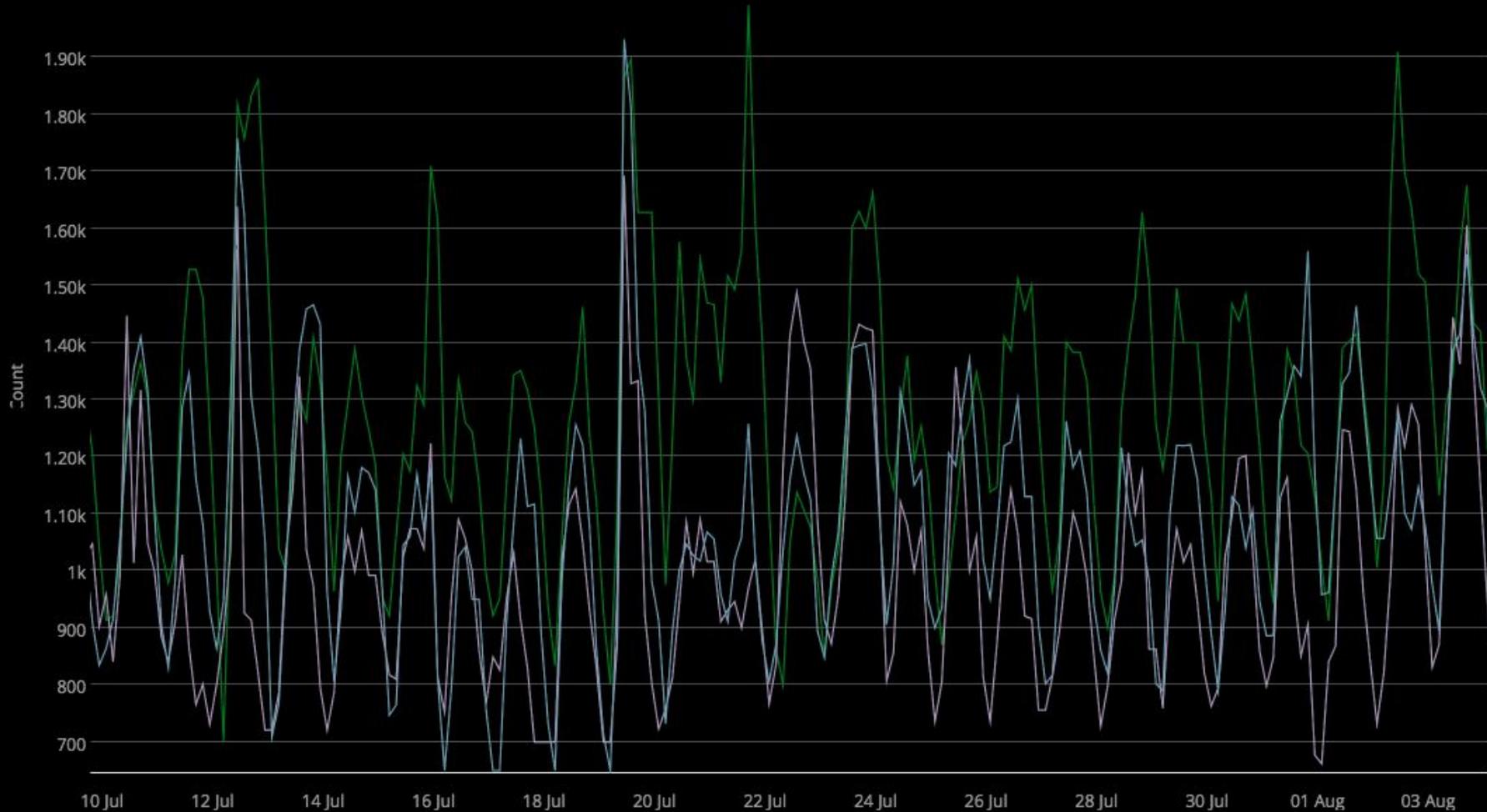
useast1



Mesos Slaves CPU Count per spot market - kwa

3h

useast1



Best Practices for “Production” SFRs

3. Just Bid High (2X the on-demand price?)



Bidding Recommendations

Instance type: m4.10xlarge

Minimum reliability requested: 100% over 30 days!

Availability Zone	Recommended Bid	Cost per hour (\$)	Saving over on-demand (%)	Saving over reserved (%)
us-west-1a	\$10.20	\$0.57	80.51%	74.73%
us-west-1b	-	-	-	-
us-west-1c	\$30.00	\$0.54	81.55%	76.09%

Explanation:

Recommended bid: The amount we think you should bid in each AZ to meet the minimum reliability requirement specified.

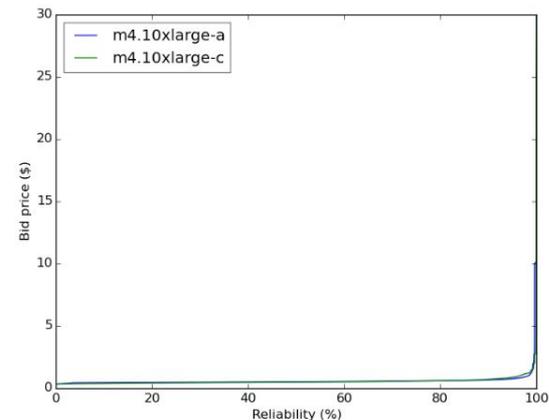
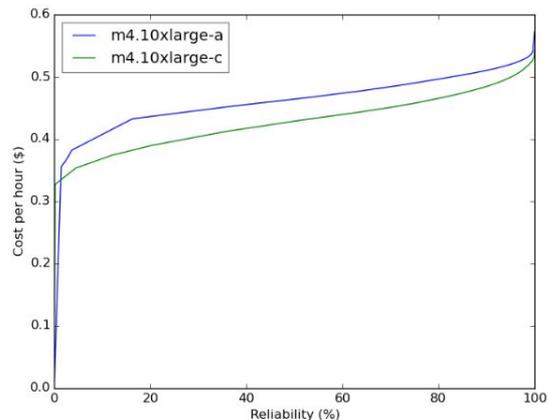
Cost per hour: The actual cost that you will end up paying if you bid according to our recommendation.

Saving over on-demand: The percentage of money you will save using spot instance (using our recommended bid) compared to on-demand price.

Saving over reserved: The percentage of money you will save using spot instance (using our recommended bid) compared to reserved-instance price.

- Bid high!
- Stay reliable
- Still save \$\$\$

Price-Reliability Curves



Cost-Reliability curves is a pictorial representation of the tradeoffs between reliability and cost of getting that reliability. Users can look at these graphs to get an idea about how much they will have to pay for getting higher reliability.

Some key points are:

- Lower-right corner is desirable: High reliability with low cost

- Top-left corner is undesirable: Low reliability with high cost

- The crossover points between different AZs are important since they can change your preference of an AZ

Bidding Recommendations

Instance type: c4.4xlarge

Minimum reliability requested: 100% over 30 days!

Availability Zone	Recommended Bid	Cost per hour (\$)	Saving over on-demand (%)	Saving over reserved (%)
us-west-1a	\$11.20	\$1.41	-27.31%	-63.25%
us-west-1b	-	-	-	-
us-west-1c	\$11.20	\$1.34	-21.16%	-55.35%

Explanation:

Recommended bid: The amount we think you should bid in each AZ to meet the minimum reliability requirement specified.

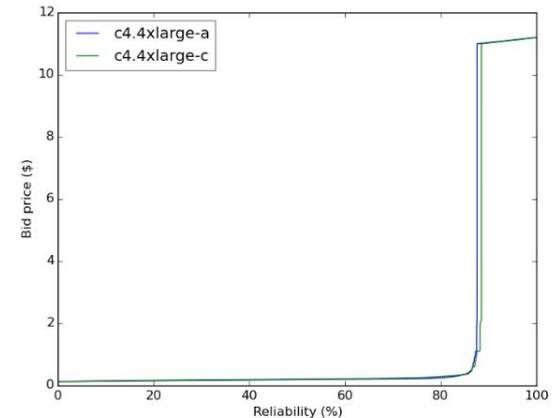
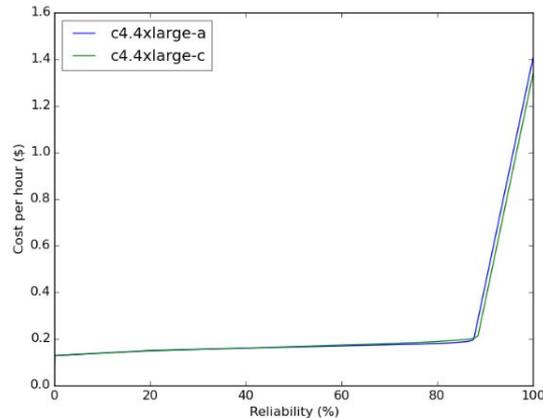
Cost per hour: The actual cost that you will end up paying if you bid according to our recommendation.

Saving over on-demand: The percentage of money you will save using spot instance (using our recommended bid) compared to on-demand price.

Saving over reserved: The percentage of money you will save using spot instance (using our recommended bid) compared to reserved-instance price.

- This time bidding super high is a bum deal
- Don't bid *that* high

Price-Reliability Curves



Cost-Reliability curves is a pictorial representation of the tradeoffs between reliability and cost of getting that reliability. Users can look at these graphs to get an idea about how much they will have to pay for getting higher reliability.

Some key points are:

- Lower-right corner is desirable: High reliability with low cost

- Top-left corner is undesirable: Low reliability with high cost

- The crossover points between different AZs are important since they can change your preference of an AZ

Best Practices for “Production” SFRs

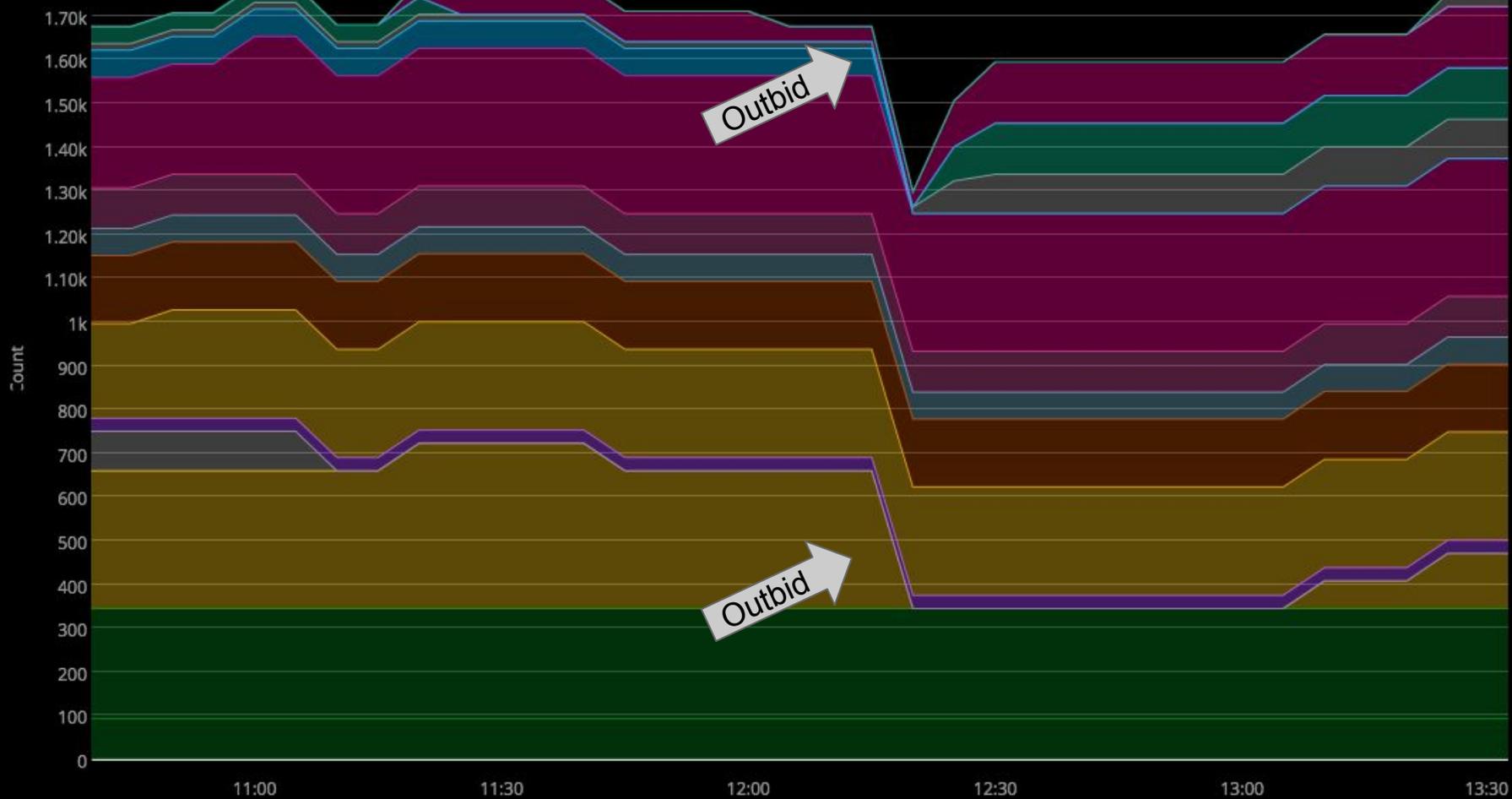
4. Deal with terminations



Mesos Slaves CPU Count per spot market - kwa

5m

uswest1



Mesos Slaves CPU Count per spot market - kwa

1m

uswest1



If you're new to Mesos

See the [getting started](#) page for more information about downloading, building, and deploying Mesos.

If you'd like to get involved or you're looking for support

See our [community](#) page for more details.

Maintenance Primitives

Operators regularly need to perform maintenance tasks on machines that comprise a Mesos cluster. Most Mesos upgrades can be done without affecting running tasks, but there are situations where maintenance may affect running tasks. For example:

- Hardware repair
- Kernel upgrades
- Agent upgrades (e.g., adjusting agent attributes or resources)

Frameworks require visibility into any actions that disrupt cluster operation in order to meet Service Level Agreements or to ensure uninterrupted services for their end users. Therefore, to reconcile the requirements of frameworks and operators, frameworks must be aware of planned maintenance events and operators must be aware of frameworks' ability to adapt to maintenance. Maintenance primitives add a layer to facilitate communication between the frameworks and operator.

Terminology

For the purpose of this document, an “Operator” is a person, tool, or script that manages a Mesos cluster.

Support mesos primitives for availability #5435

 **Open** alanbover wants to merge 5 commits into `mesosphere:master` from `alanbover:alan/MARATHON-3216`

 Conversation **35**

 Commits **5**

 Files changed **11**



alanbover commented on Jul 11

First-time contributor



This PR makes marathon drop those resource offers from agents that are set in maintenance.

Find more information here: <https://jira.mesosphere.com/browse/MARATHON-3216>



Support mesos primitives for availability

521266f



janisz suggested changes on Jul 11

[View changes](#)

Re-use those same primitives for dealing with spot termination

Inspect this url for termination events:

<http://169.254.169.254/latest/meta-data/spot/termination-time>



General Advice Summary

- Diversify as much as you can
- Lock spot fleets per-az
- Set a spot_market mesos attribute
 - `"%{::ec2_instance_type}-%{::aws_availability_zone}"`
- Respond to maintenance events as best as you can



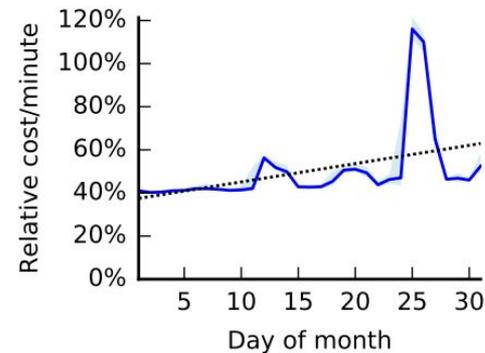
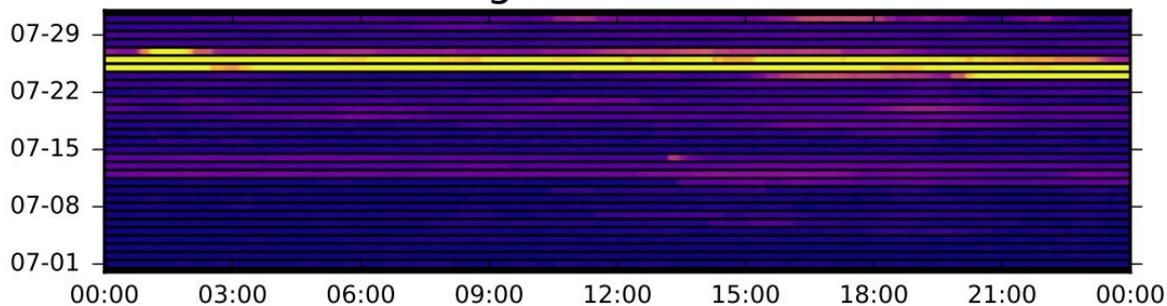
Profit?

- Is it worth living with this instability?



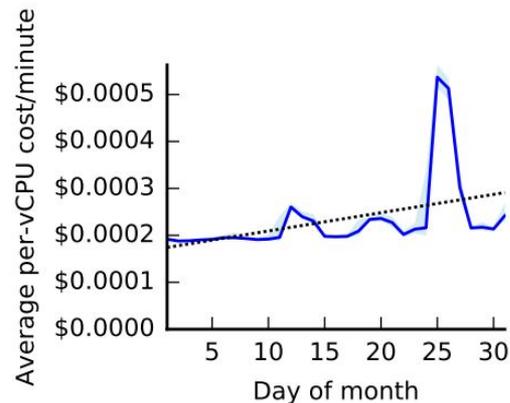
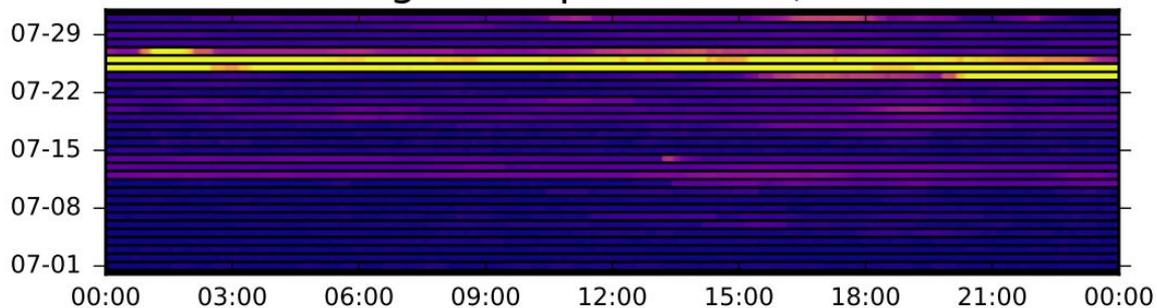
PaaS cost relative to RI data for July 2017

Average cost: 51% of RI



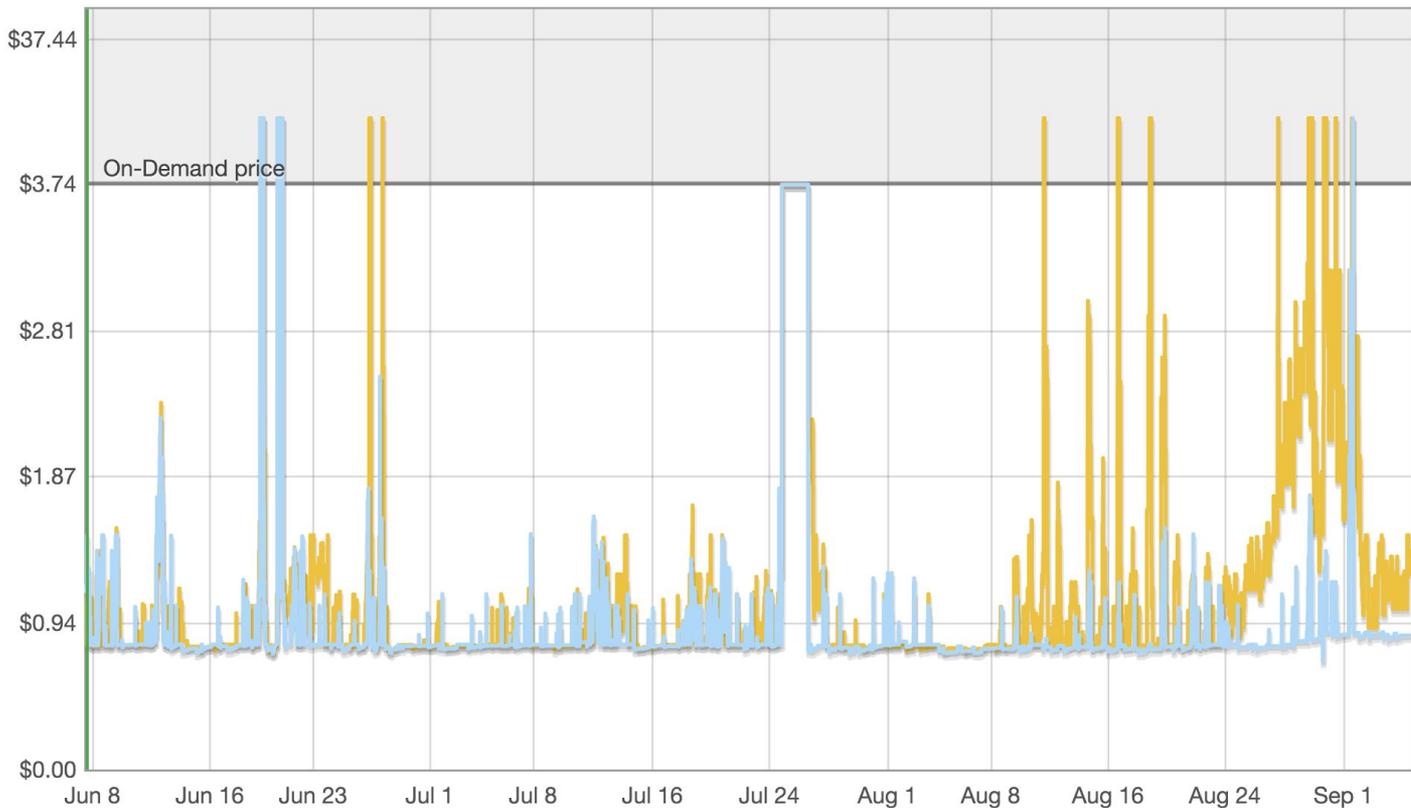
PaaS price per vCPU data for July 2017

Average cost per vCPU: \$0.0002



Spot Instance Pricing History

Product: Linux/UNIX (Amzn) Instance type: m4.16xlarge Date range: 3 months



Date
7/19/2017
5:07:39 AM UTC-0700

On-Demand price
\$3.7440

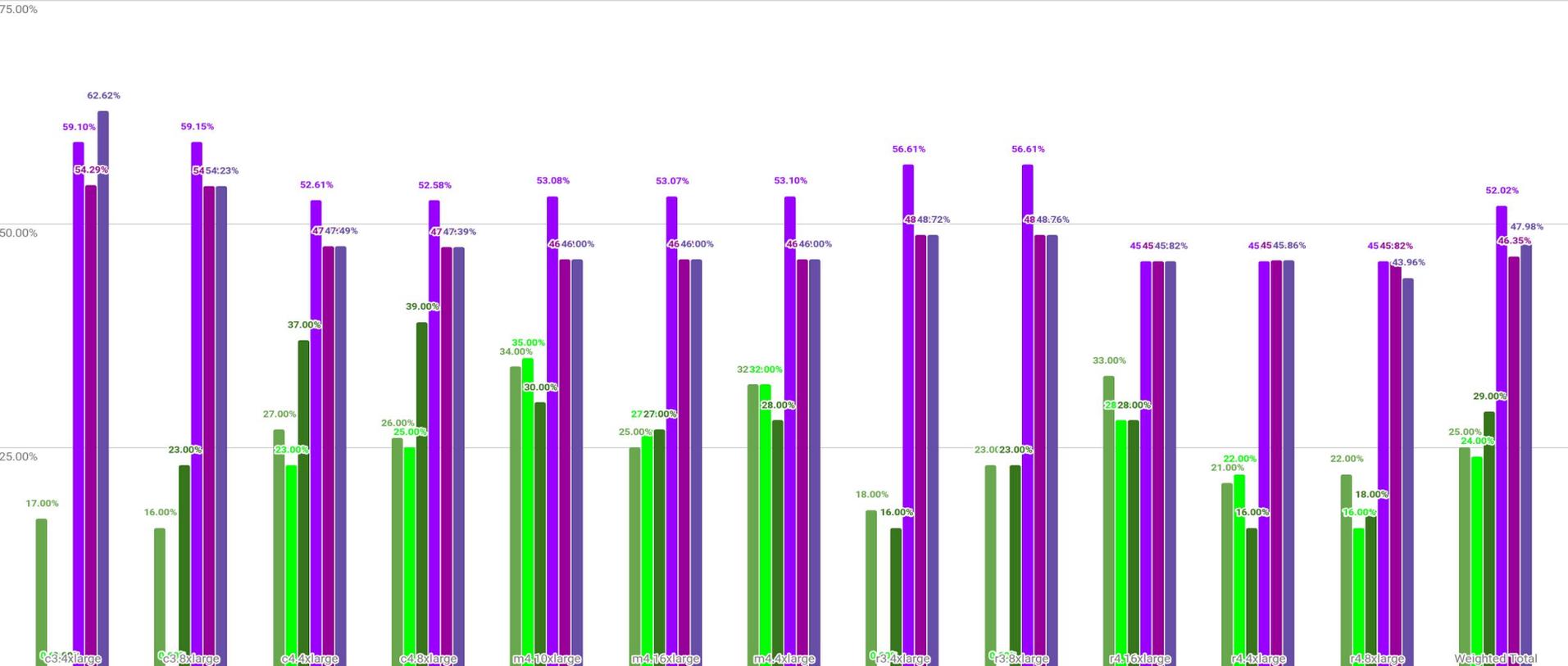
Availability Zone	Price
us-west-1a	\$1.2500
us-west-1b	\$0.8519



Yelp's Spot Mesos Cluster Cost for August 2017

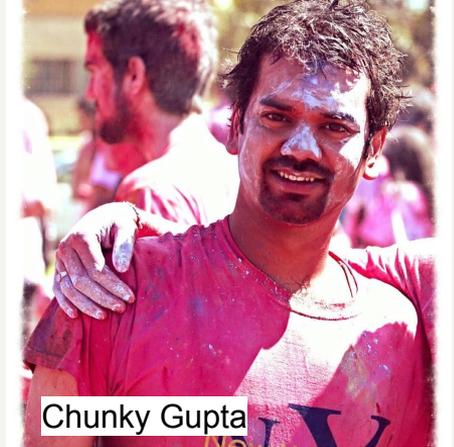
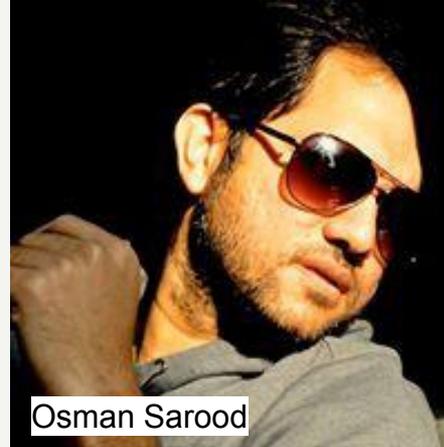
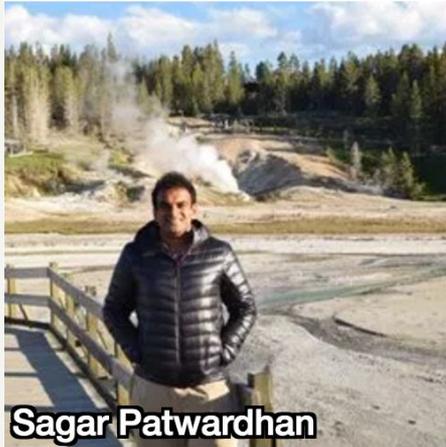
Prod clusters only, non-stateful services

■ Percent we pay vs OD uswest1
 ■ Percent we pay vs OD useast1
 ■ Percent we pay vs OD uswest2
 ■ Percent RI of OD uswest1
 ■ Percent RI of OD useast1
 ■ Percent RI of OD uswest2



	Type / Region	us-west-1	us-east-1	us-west-2	
What percent are we paying compared to 3-year Convertible RI Partial Upfront (For prod in August 2017)	<i>c3.4xlarge</i>	29%	0%	0%	
	<i>c3.8xlarge</i>	27%	0%	42%	
	<i>c4.4xlarge</i>	52%	49%	78%	
	<i>c4.8xlarge</i>	49%	53%	81%	
	<i>m4.10xlarge</i>	65%	77%	65%	
	<i>m4.16xlarge</i>	47%	59%	58%	
	<i>m4.4xlarge</i>	60%	70%	62%	
	<i>r3.4xlarge</i>	32%	0%	34%	
	<i>r3.8xlarge</i>	41%	0%	48%	
	<i>r4.16xlarge</i>	71%	62%	61%	
	<i>r4.4xlarge</i>	45%	49%	35%	
	<i>r4.8xlarge</i>	48%	34%	42%	
	Weighted Total		47%	51%	60%

Shoutouts - Yelp Spot Early Adopters



Shoutouts - Production (Operations)



Practical:

- <https://www.appneta.com/blog/aws-spot-instances/>
- <https://github.com/cristim/autospotting/>
- <https://www.cmpuete.io/>
- <https://spotinst.com>
- <https://autoscalr.com/2017/07/25/strategies-mitigating-risk-using-aws-spot-instances/>
- <https://github.com/yelp/paasta>

Academic:

- On the Viability of a Cloud Virtual Service Provider:
https://www.andrew.cmu.edu/user/cjoewong/CVSP_SIGMETRICS.pdf
- Cloud Spot Markets are not Sustainable:
https://www.usenix.org/system/files/conference/hotcloud16/hotcloud16_subramanya.pdf





kwa@yelp.com



[@YelpEngineering](https://twitter.com/YelpEngineering)



engineeringblog.yelp.com



github.com/yelp