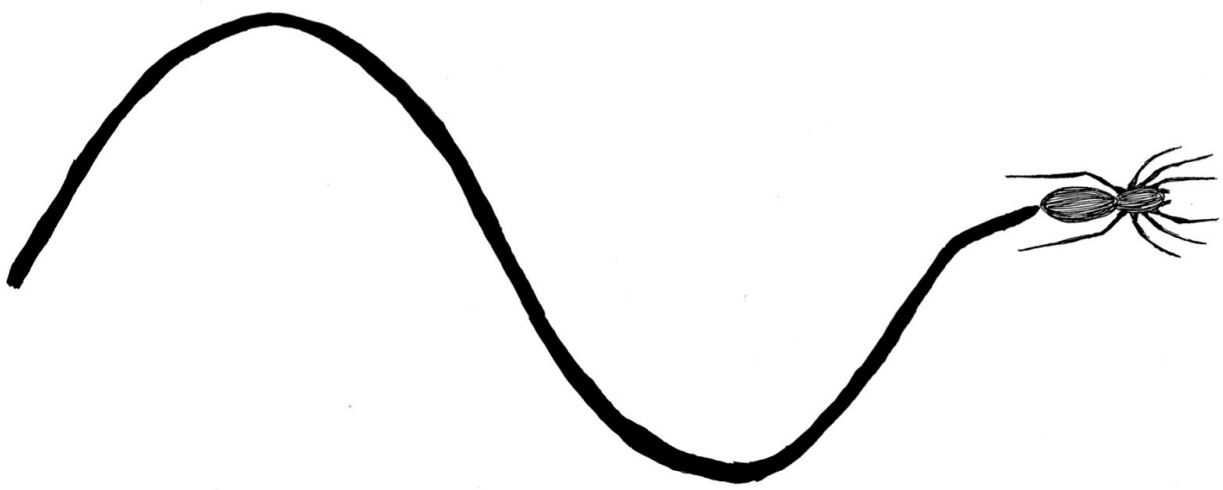
Effectively Using AWS EC2 Spot Instances to Reduce the Cost of Computation



Acksin, LLC
https://www.acksin.com
hey@acksin.com

Abstract

Spot instances provide a way to reduce the cost of running an EC2 (Elastic Compute Cloud) infrastructure by providing an alternate to OnDemand and Reserved instance types. They provide savings of up to 85% on OnDemand costs by providing no SLA (Service Level Agreement). However, with some best practices in architecting your applications you can achieve major cost savings while ensuring high availability.

Abstract
Introduction
What are Spot Instances?
What the Benefits of using Spot Instances?
What are some of the issues with Spot instances?
What are some use cases for Spot Instances?
    Batch Jobs
    AutoScaling API and Frontends
    Proof of Concept
    Staging Environments
    Continuous Integration and Testing
Architectures for Running on Spot Instances
    Combining OnDemand with Spot Instances
    Batch Processing
Bidding on Spot Instances
Conclusion
References

Introduction

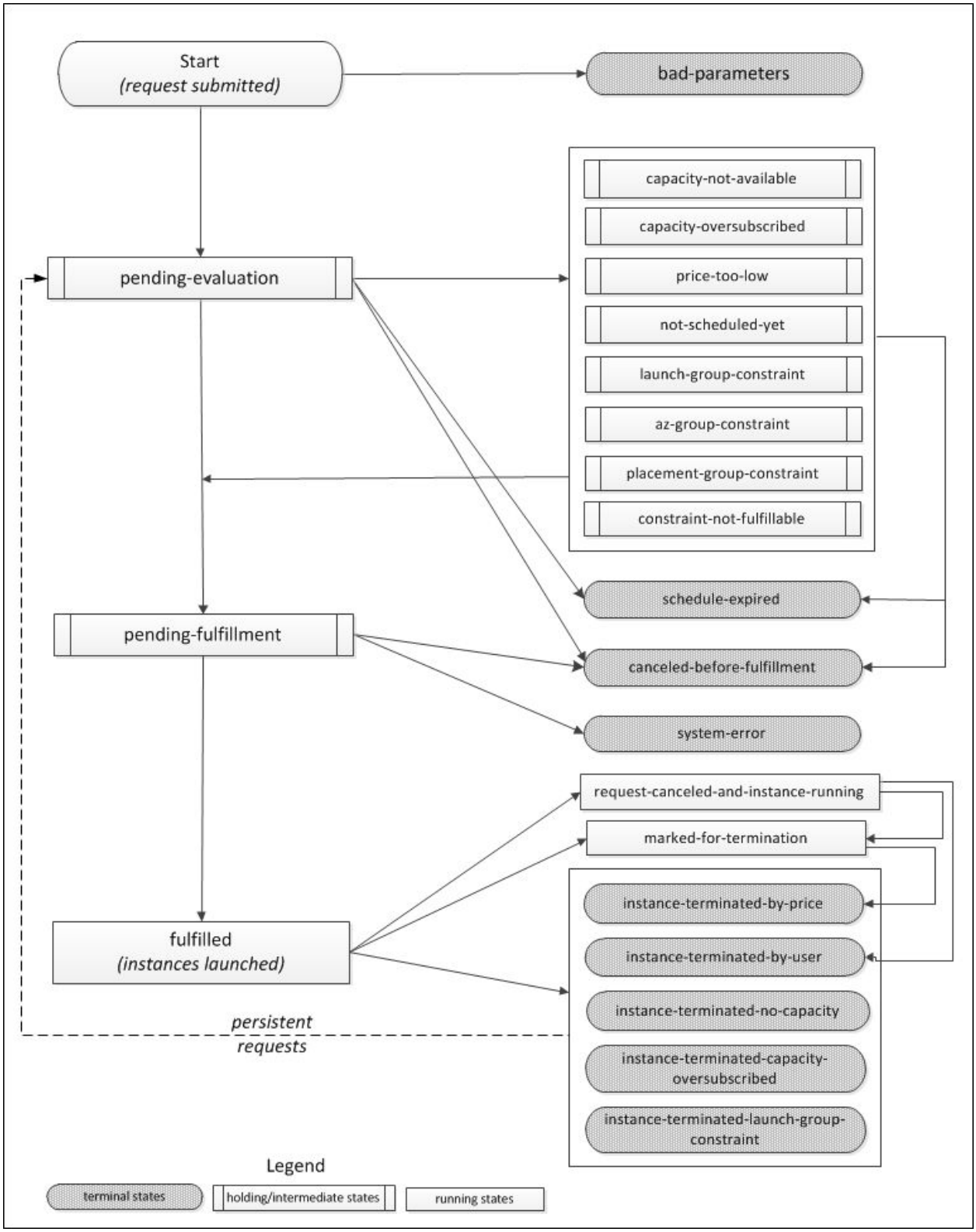
The cost of running an EC2 (Elastic Compute Cloud) infrastructure becomes expensive when using just OnDemand instances. Even using Reserved instances might not give you a benefit if the load you experience is temporary or cyclical. We believe that by using Spot instances with some best practices you can realize large amounts of savings and attain high availability in multiple situations. In this white paper we will cover several things including what are EC2 Spot Instances, best use cases for Spot Instances and how to build your application's architecture to use Spot Instances effectively.

What are Spot Instances?

EC2 instances provided by AWS (Amazon Web Services) come in three different flavors: OnDemand, Reserved, and Spot. Spot Instances are a marketplace provided by AWS which allows you to bid on unused instances. Whereas OnDemand and Reserved provide a 99.95% SLA, Spot instances provide no SLA at all. However, Spot instances reduce the cost by up to 85% and these savings can be realized with some modifications to your architecture. These instances are provided at a price called the *spot price* which changes constantly based on supply and demand. You get the instance if your bid is higher than the current spot price and will lose the instance if you are outbid by someone else at a higher spot price.

Spot instances come in three different varieties: One-time, Persistent, Spot Block.

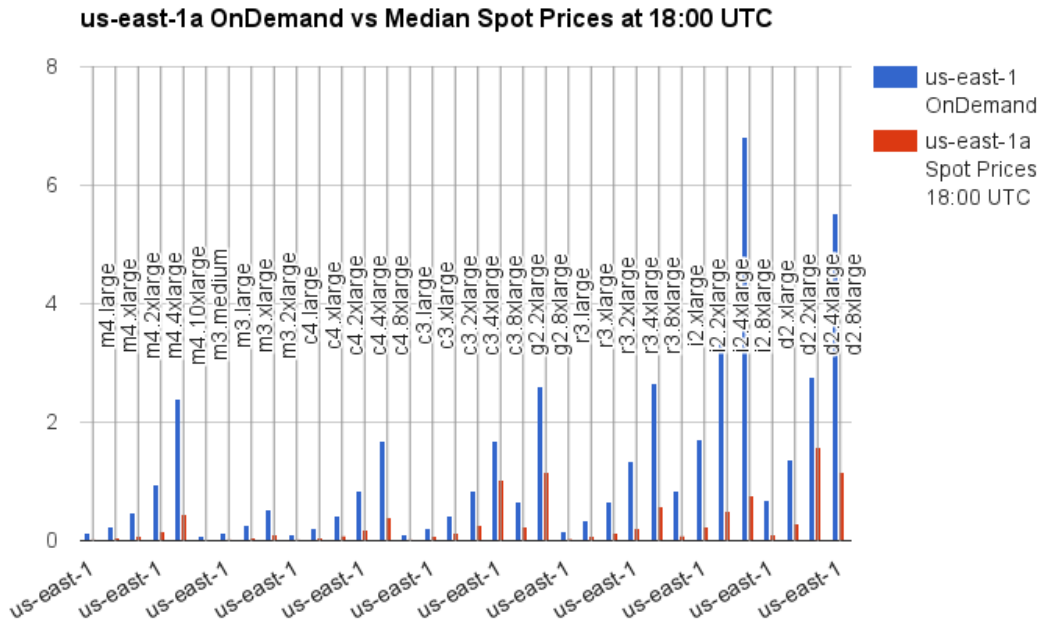
- One-time Spot Instances runs when a bid is fulfilled. When it is terminated it does not attempt to come back up unless you manually bid for an another instance.
- Persistent Spot Instances are machines that come up when a bid is fulfilled. If the instance is terminated because the bid is too low EC2 will attempt to fulfill the request until the instance is started again.
- Spot Block Instances are instances that run for a specific duration from 1 to 6 hours but they usually only have savings of up to 30-45%. These are useful if you need an alternative to Reserved Instances for a short duration of time. And since they are blocked for a duration you can ensure that these machines will not be terminated.



Spot Instance Bid Status. What happens when you bid on a Spot Instance.

What are the Benefits of using Spot Instances?

The major advantage of using Spot instances is the realization of significant reduction in cost over both OnDemand and Reserved Instances. And to reiterate, savings of up to 85% on OnDemand prices can be attained. The graph below shows the OnDemand prices in us-east-1 and the median Spot Prices in the availability zone us-east-1a at 18:00 UTC (2:00 PM EST) for the period of June 7, 2016, to June 14, 2016. As you can see the spot prices are significantly lower than their OnDemand price.



What are some of the issues with Spot instances?

As with any type of auction, there are risks involved with bidding in a dynamic marketplace. With Spot instances, common issues include overbidding, underbidding, and delays. Overbidding can results in higher cost threshold and increase the spot prices for everyone including you. By underbidding, you are susceptible to getting outbid. During times of peak demand, desired machines may be unavailable at a reasonable cost. These problems can increase costs and reduce the efficacy of the Spot instances.

A common danger lies in choosing your bidding strategy. Your bidding strategy can become a liability. Many companies have decided to use a Spot first infrastructure by making their bids significantly higher than the OnDemand prices thinking that spot prices will never go higher than the OnDemand price. They assume that by overbidding, they will have full control of the instance until they stop bidding. Unfortunately, in certain occurrences, the cost of Spot Instances can go well above the OnDemand price losing all cost savings. If the spot prices go higher than the OnDemand price, you can run up a large bill if you are are not careful with your bids.

You lose the instance, i.e get evicted, if you get outbid by someone else. When this happens you have two minutes to backup any logs or other data. This can increase the time it takes to handle certain computational tasks if you are unprepared for these type of interruptions. Automating these tasks can ensure your backups occur within the time limit and will allow you to run your applications seamlessly between Spot instances. Fortunately, we provide a tool called SeeSpot which handles cleanup tasks and health checks for when a spot termination is sent. The tool checks every 5 seconds for updates. This tool also allows you to gracefully stop accepting new incoming connections if using the Spot instance as a frontend or API server.

Finally, because Spot Instances are based on supply and demand, at peak times you may not be able to start any Spot instances if the demand is too high. This can be inconvenient at best and disastrous at worst, depending on your needs and time constraints regarding instance use. Or it might force you to use a more expensive instance that is mismatched or overpowered for the task at hand. Spot Instances also may not start immediately especially if your bid is lower than the current spot price. This means that there may be a delay between your bid time and when your machine is available to do any computation.

What are some use cases for Spot Instances?

Spot instances can be used for multiple use cases but there are a few circumstances where they really shine. These use cases have been used in production environments and are effective for cutting costs.

Batch Jobs

Spot instances are great for batch jobs that need to run once or twice a day. An example is say you have a computationally expensive job that needs to run on a large instance, like a c4.8xlarge machine. Instead of running it on an OnDemand Instance, you can use AWS Lambda to trigger a One Time Spot Instance to launch during off-peak hours. When the instance launches it can run the run the batch job and terminate on completion. This allows you get maximum savings both by running during off-peak hours and by not using more pricey OnDemand Instances. Coordinating extra or higher loads with periods of low demand for instances, such as nights and weekend, can be beneficial. Nights and weekends tend to have a higher availability of instances.

AutoScaling APIs and Frontends

Spot instances can be used to handle auto-scaling when the load hits certain thresholds. This could be heavy client traffic on an AWS ELB (Elastic Load Balancer) or when a certain threshold is hit on a messaging queue. While scaling with OnDemand instances to handle this extra load is possible, it is not as cost effective. For example, if you have regular, anticipated high loads when Spot instance availability is high, like a consumer app during the weekend, you can take advantage of the savings associated with using Spot Instances.

Proof of Concept

When writing a proof of concept application a high availability instance might not be necessary. In this situation, you can use Spot Instances to prototype and build out the application until you are ready for deployment into production.

Staging Environments

Spot instances have been used for testing and running staging environments. These environments usually has low utilization but need to be similar to the production environments leading to a high cost to utilization ratio. These environments have benefited from reduced costs via Spot Instances.

Continuous Integration and Testing

If you are running your own Jenkins infrastructure, you may use the Jenkins EC2 Fleet Plugin to run your tests on Spot Instances to reduce costs. This also allows you to bring up and down machines with test load bringing down machines during periods where there is no test activity such as weekends and evenings.

Architectures for Running on Spot Instances

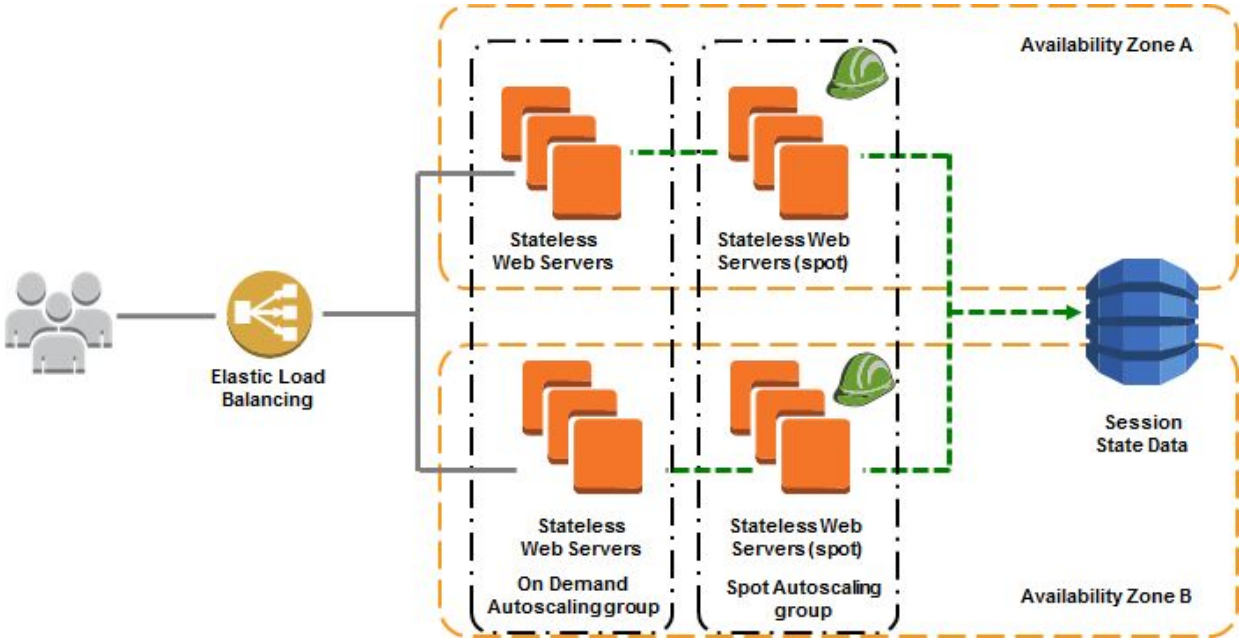
If you have an existing application that you'd like to convert to run on Spot instances, there are a few changes that need to be done to ensure that things run smoothly. This especially holds true when considering the no SLA nature of Spot Instances. To use Spot instances effectively we need to make sure that these are in place: resilient software architecture and an optimal bidding strategy to reduce server eviction. This section will show a few different architectures that can be used with Spot instances.

Combining OnDemand with Spot Instances

The first method of saving costs on EC2 instances is to simply split your architecture such that it uses both OnDemand and Spot instances. This allows for the following benefits:

- Uptime guarantees of OnDemand instances
- Cost savings of Spot instances.

Say you have a group of OnDemand/Reserved Instances that will be your main app servers and since these are OnDemand instances they have a high SLA. You can augment those instances with Spot Instances whenever a load hits a certain threshold. See the diagram below from this AWS Compute Blog post. By having a subset of your infrastructure in Spot instances you can ensure that your clients can reach your service without downtime but also allow for unexpected spikes in traffic with a reduced hit on your wallet.



In the diagram above the instances are behind an AWS ELB with OnDemand instances running in different availability zones. The example case has both OnDemand and Spot instances specified in the Autoscaling Groups to guarantee that load is handled. Furthermore, as you can see the servers are stateless and all transactional things happen in an external data source such as RDS, DynamoDB or your own custom installation running on an OnDemand machine. This architecture is in use at multiple large consumer app companies to handle unexpected spikes in traffic and when there is a need to do a large computational job that can be handled by scaling horizontally such as MapReduce jobs.

Some code to change If running behind an ELB is to have the Spot Instance monitor for a termination notification which gives two minutes for cleanup. During this period no new clients should be handled, existing clients should be flushed, the health check should be marked as failed and any data that is needed for later should be backed up. Again our tool SeeSpot handles these situations.

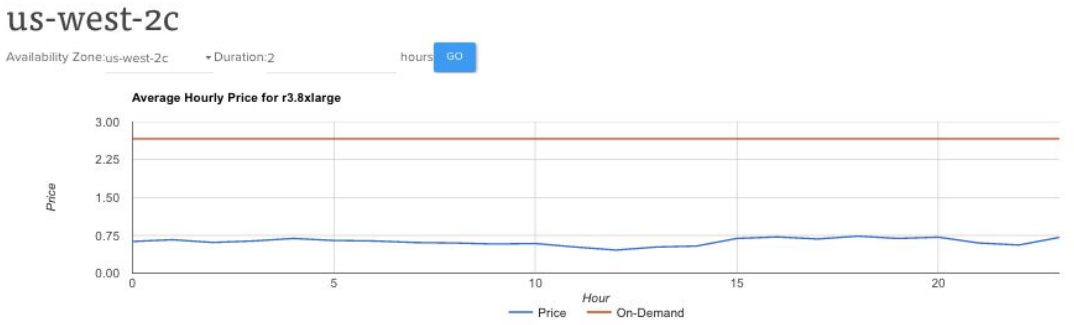
Batch Processing

Batch processing should have idempotent or transactional behavior. A termination event of a Spot instance in the middle of processing should not affect the end user and care

should be taken into writing applications that minimize the loss, especially of long running computations. To achieve this is very programming language and architecture specific but we recommend that you use messaging queues as intermediary points and have the applications do very specific tasks limiting the time it takes to run any individual task so a termination notification event would give enough time to complete the task.

Bidding on Spot Instances

Finally, we get to bidding for Spot Instances. Amazon does provide a way to look at spot pricing history and the get the current spot prices. However, the provided information doesn't specify or suggest an optimal bid that would ensure a high uptime while lowering the chance of eviction. It simply presents the current spot price, not what they will be in 5 minutes or even an hour.



Instance Type	OnDemand	Recommended Bid	Savings
t1.micro	N/A	N/A	0%
c1.medium	N/A	$0.032563631543207336	N/A
c1.xlarge	N/A	$0.072528195482005425	N/A
cc2.8xlarge	N/A	$0.37530220923300656	N/A
cg1.4xlarge	N/A	N/A	0%
cr1.8xlarge	N/A	$0.44297410729800474	N/A
hi1.4xlarge	N/A	$0.40683414436403853	N/A
m1.xlarge	N/A	$0.034490935167803709	N/A
m1.large	N/A	$0.020686795106454475	N/A
m1.medium	N/A	$0.014997862133151227	N/A
m1.small	N/A	N/A	0%
m2.4xlarge	N/A	$0.10797446343287835	N/A
m2.2xlarge	N/A	$0.045041204467611361	N/A
m2.xlarge	N/A	$0.035390914670121967	N/A
m3.medium	$0.067	$0.024980548178724565	63%
m3.2xlarge	$0.532	$0.11451349139849214	78%
m3.xlarge	$0.266	$0.057419154997645432	78%

AWS only provides one clue to help guide you through the bidding process. According to the AWS documentation, “Your bid price should be high enough to make it likely that your request will be fulfilled, but not higher than you are willing to pay. This is important because if the supply is low for an extended period of time, the Spot price can remain high during that period because it is based on the highest bid prices. We strongly recommend against bidding above the price for On-Demand instances.” This is good advice however it is advice rather than a bid to use. This is where we at Acksin step in with our tool ParkingSpot.

Acksin provides a Spot Instance bidding tool called ParkingSpot which can be accessed either programmatically or via a web interface that allows you to create an optimal bid that reduces the chance of eviction. We do this by looking at the bidding history currently encompassing over 30 million historic spot price points as well as current spot prices to find trends and recommend an optimal bid.

Since a high bid price increases spot prices for everyone we attempt to bid at a level that ensures a low chance of getting evicted but still bid at a price that ensures that the prices don’t increase significantly for everyone.

Furthermore, with programmatic access to the optimal bid prices you can develop applications that can bid on and start Spot instances when prices are low such as during off-peak hours such as nights and weekends.

Conclusion

As you can see Spot instances allow the ability to significantly cut costs. With some architectural changes and an effective bidding strategy, you can make Spot instances an effective part of your infrastructure handling which can handle multiple uses cases.

References

- https://www.acksin.com/parkingspot
- https://github.com/acksin/seespot
- https://www.cs.ucsb.edu/~ckrintz/papers/gca15.pdf
- http://www.princeton.edu/~liangz/paper/CloudBidding.pdf
- https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/
- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html
- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html
- https://aws.amazon.com/blogs/compute/an-ec2-spot-architecture-for-web-applications/
- https://wiki.jenkins-ci.org/display/JENKINS/Amazon+EC2+Fleet+Plugin