AMAZON CLOUD

Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market.

Abstract

The study aims at determining the attributes of managing Peak Loads of Leasing Cloud Infrastructure Services from a Spot Market. The advent of cloud computing promises to provide computational resources. To deal with dynamically fluctuating resource demands, market-driven resource allocation has been proposed and recently implemented by public Infrastructure-as-a-Service (IaaS) providers like Amazon EC2.

Cloud resources are offered in distinct types of virtual machines (VMs) and the cloud provider runs an auction-based market for each VM type with the goal of achieving maximum revenue over time. However, as demand for each type of VMs can fluctuate over time, it is necessary to adjust the capacity allocated to each VM type to match the demand in order to maximize total revenue while minimizing the energy cost. We consider the case of a single cloud provider and address the question how to best match customer demand in terms of both supply and price in order to maximize the providers revenue and customer satisfactions while minimizing energy cost.

Research Summary

There are three major cloud service models: software as a service (SaaS), infrastructure as a service (IaaS) and platform as a service (PaaS). Cloud service pricing models are categorized into pay per use, subscription-based and hybrid, which is a combination of pay-per-use and subscription pricing models.

Software as a Service

Software as a service vendor host the applications, making them available to users via the internet. With SaaS, businesses don't have to install or download any software to their existing IT infrastructures. SaaS ensures that users are always running the most up-to-date versions of the software. The SaaS provider handles maintenance and support.

Platform as a Service

Platform as a service offers developers a platform for software development and deployment over the internet, enabling them to access up-to-date tools. PaaS delivers a framework that developers can use to create customized applications. The organization or the PaaS cloud vendor manage the servers, storage and networking, while the developers manage the applications.

Infrastructure as a Service

Infrastructure as a service is used by companies that don't want to maintain their own on-premises data centers. IaaS provides virtual computing resources over the Internet. The IaaS cloud vendor hosts the infrastructure components that typically exist in an on-premises data center, including servers, storage and networking hardware, as well as the hypervisor or virtualization layer.

How Do the 3 Cloud Computing Service Models Differ?

The three cloud service models mainly differ in what they offer out of the box. SaaS is cloud-based software that companies can buy from cloud providers and use. PaaS helps developers build customized applications via an application programming interface (API) that can be delivered over the cloud. IaaS helps companies build the infrastructure of cloud-based services, including software, content or e-commerce websites to sell physical products.

Infrastructure as a Service

Pros of IaaS include:

Cost efficient: IaaS makes it easier, faster and more cost-effective for organizations to operate workloads because they don't have to buy, manage and support the underlying infrastructure.

Scalability: The cloud infrastructure ensures that companies have access to all the resources they need when they need them.

Cons of IaaS include:

Security: In an IaaS environment, organizations relinquish control over cloud security to the third-party vendor. So even though a data breach might not directly affect a company's data, the compromised system could still endanger the its operations.

Technical issues: Some organizations may experience downtime with IaaS that they can't control. Any problems the provider experiences could limit companies' access to the applications and data they need to operate on a daily basis.

Use cases for IaaS include:

Website hosting: Organizations can save money using IaaS tools rather than traditional web hosting to run their websites.

Deploying software: Companies can use IaaS to deploy and run common business software, such as SAP and Salesforce.

Testing and development: IaaS enable developers to more easily scale up development and test environments.

Medium and large businesses that have the necessary IT resources should think about using infrastructure as a service. The almost complete control that IaaS provides means they can create highly customized technology stacks that meet an organization's specific business requirements. IaaS also makes it easy to adapt the technology if business requirements change.

Objectives

One of the important issues facing cloud computing is scheduling, which affects system performance.

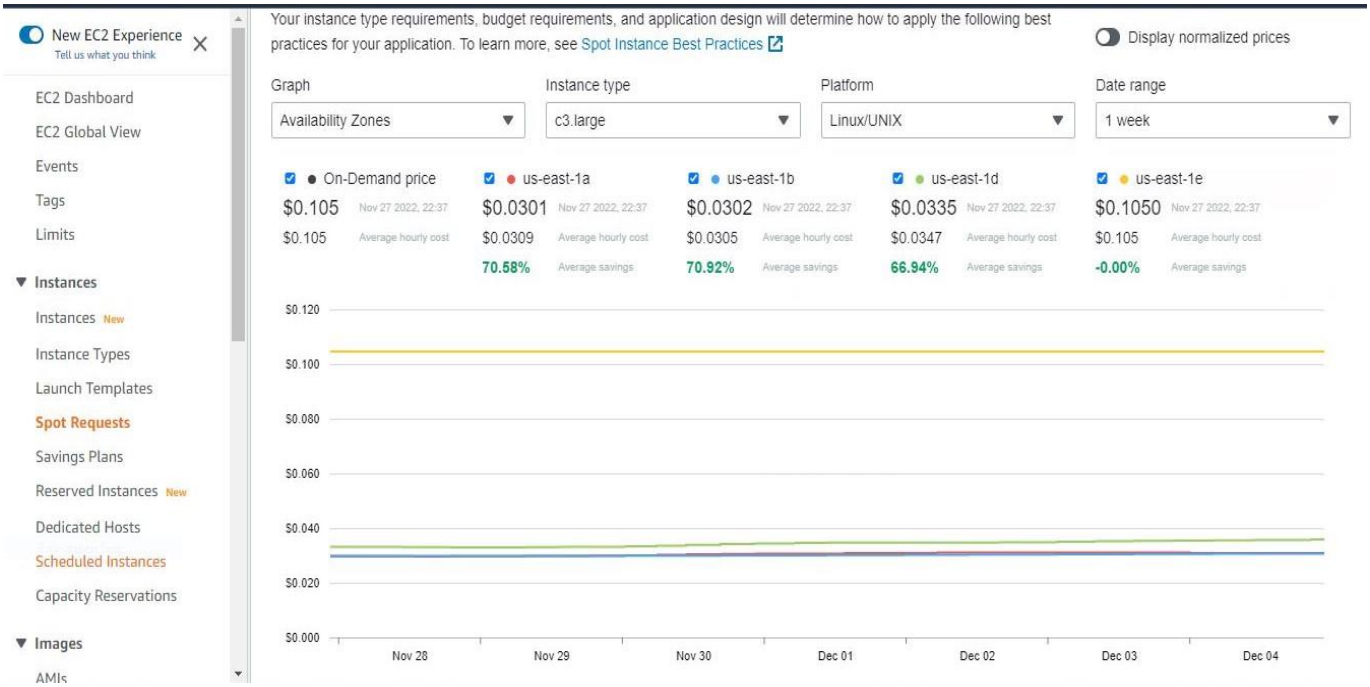
General objective

I. The objective is to help Cloud Solution Provider (CSPs) manage their resources more efficiently and effectively.
II. The objective of cloud computing is to provide easy, scalable access to computing resources and IT services.

Overview

Experiment outline.

Comparison of the Spot pricing history of the available spots in different time zones.



Relevant literature.

One of the important issues facing cloud computing is scheduling, which affects system performance. Scheduling mainly enhances the efficient utilization of resources and hence reduction in task completion time. Latency-critical workloads need be scheduled as soon as possible to avoid any queuing delays. The challenge is how to minimize the queuing delays of short jobs while maximizing cluster utilization.

Some scheduling approaches only assign resources statically, but do not regulate resources after the initial assignment. These methods fail to provide an effective

solution for managing the dynamic nature of cloud workloads. Most cloud service provider (CSPs) need to understand their workload properties to effectively predict the quality of service and optimize the specified metrics.

A resource management solution can provide important application utilization data, such as per-application cache occupancy and per-application memory bandwidth data. By incorporating the counters provided by the platform's Performance Monitoring Unit (PMU), such as cycles per instruction and low-level cache misses per instruction, a resource management solution makes it possible to access valuable information to enhance application performance monitoring.

Proposed Analysis.

Analyzing the Amazon historical spot pricing using modern data science toolsets.

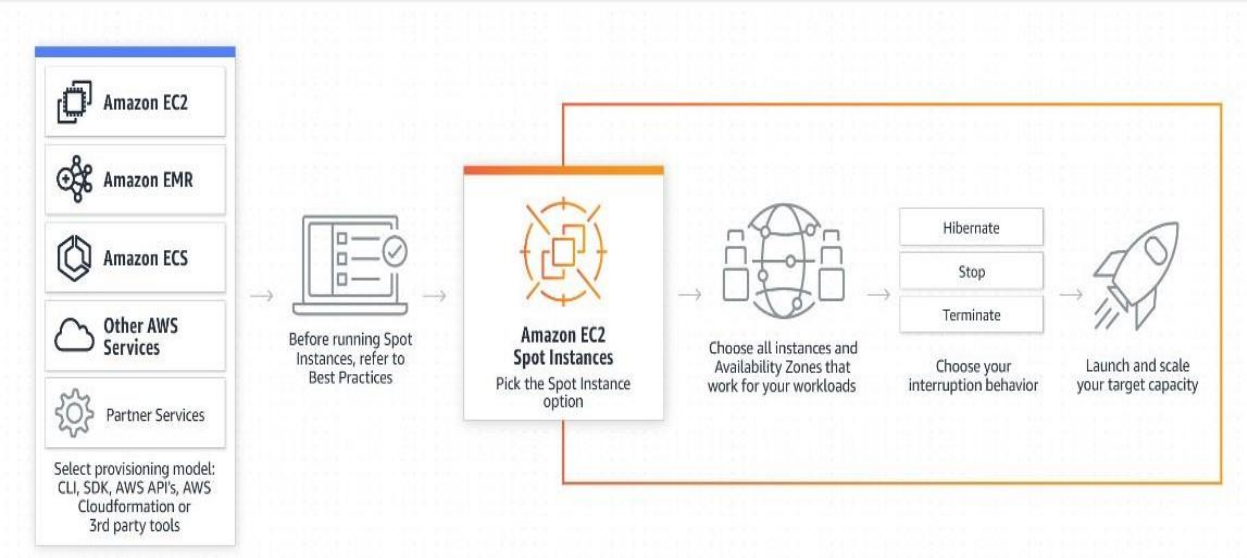
Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices. You can use Spot Instances for various stateless, fault-tolerant, or flexible applications such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and test & development workloads. Because Spot Instances are tightly integrated with AWS services such as Auto Scaling, EMR, ECS, CloudFormation, Data Pipeline and AWS Batch, you can choose how to launch and maintain your applications running on Spot Instances.

Moreover, you can easily combine Spot Instances with On-Demand, RIs and Savings Plans Instances to further optimize workload cost with performance. Due to the operating scale of AWS, Spot Instances can offer the scale and cost savings to

run hyper-scale workloads. You also have the option to hibernate, stop or terminate your Spot Instances when EC2 reclaims the capacity back with two-minutes of notice. Only on AWS, you have easy access to unused compute capacity at such massive scale - all at up to a 90% discount.

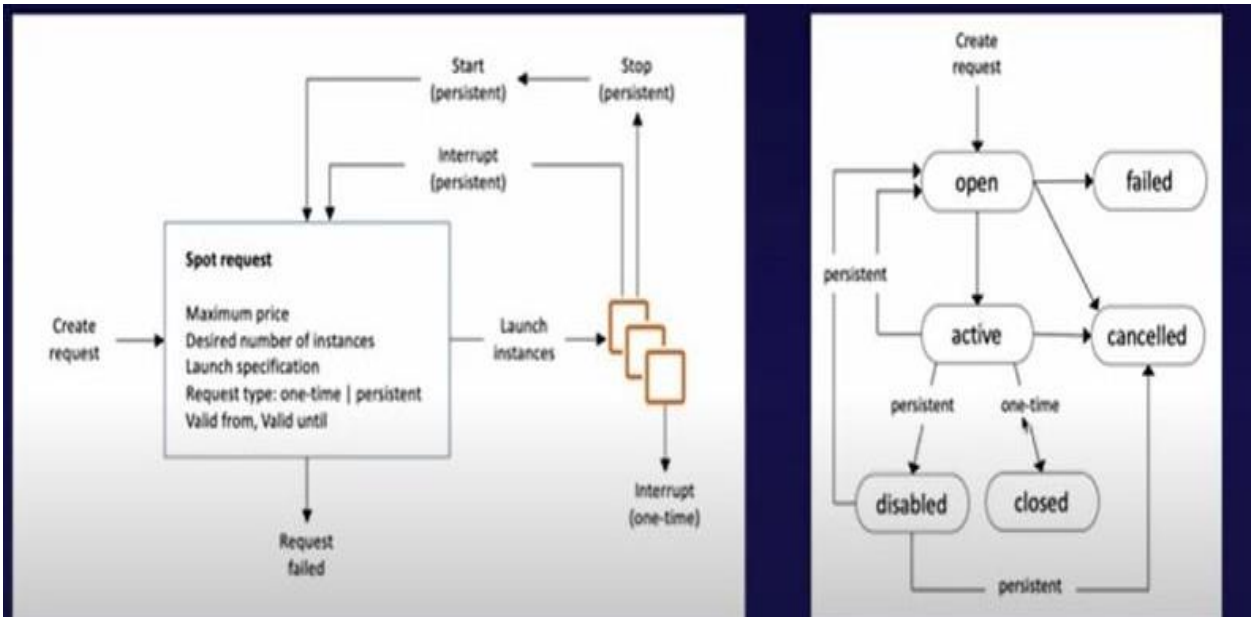
Systems setup.

Creation Of Spots



Spots Instance Creation Process

A Spot Instance is an instance that uses spare EC2 capacity that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance is called a Spot price. The Spot price of each instance type in each Availability Zone is set by Amazon EC2, and is adjusted gradually based on the long-term supply of and demand for Spot Instances. Your Spot Instance runs whenever capacity is available.



Spot Fleets

A Spot Fleet is a set of Spot Instances and optionally On-Demand Instances that is launched based on criteria that you specify. The Spot Fleet selects the Spot capacity pools that meet your needs and launches Spot Instances to meet the target capacity for the fleet.

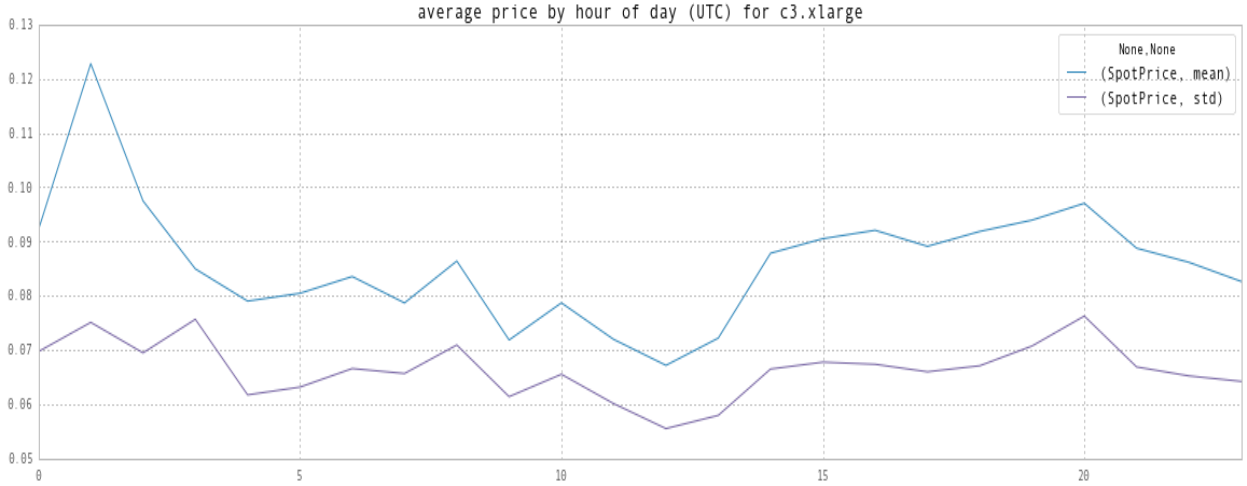
By default, Spot Fleets are set to maintain target capacity by launching replacement instances after Spot Instances in the fleet are terminated. You can submit a Spot Fleet as a one-time request, which does not persist after the instances have been terminated. You can include On-Demand Instance requests in a Spot Fleet request.

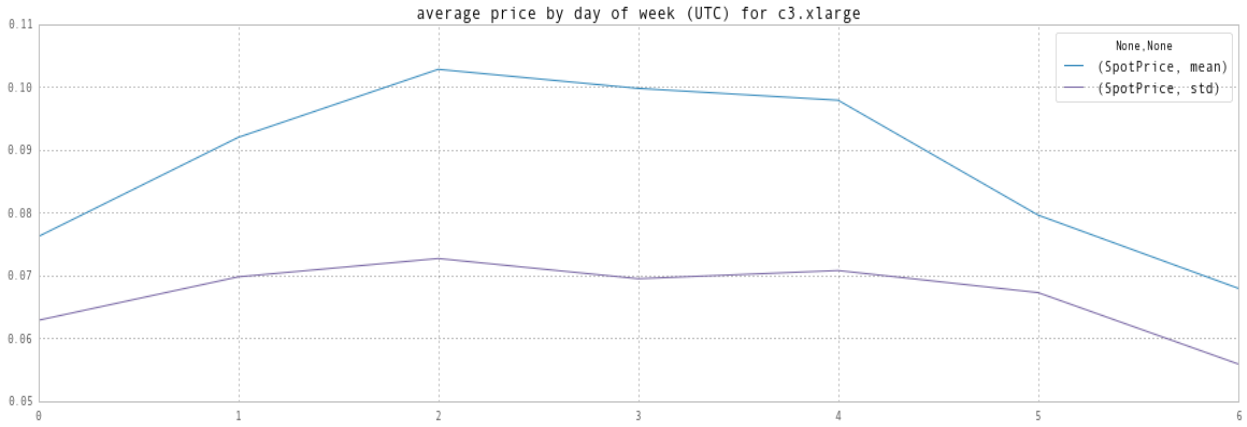
Benchmarking

Amazon Web Services Cloud Computing provides a number of different pricing models ranging from pay-per-use (on-demand), fixed (reserved), and auction-based (spot). The spot model is on the order of 8–10x cheaper as compared with on-demand, but with a twist: since instances in the spot market are auctioned, you may or may not "win" the auction. Even if you win and get the instances you asked for, you may subsequently loose them with little notice depending on the next auction and how others are bidding.

AWS provides data on spot market prices for each instance type in each availability zone (AZ) for the trailing 90 days. When the market price exceeds your bid price, then your resources will be lost with very little notice. So, you have to set a bid price balancing volatility and cost.

Simulations.

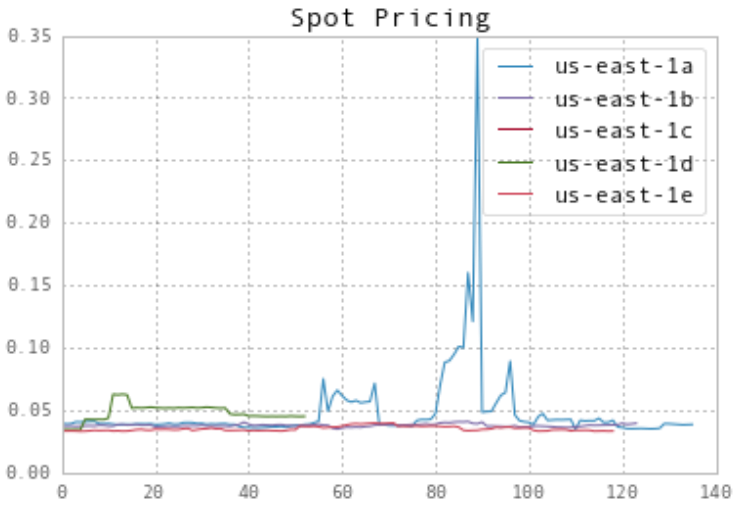




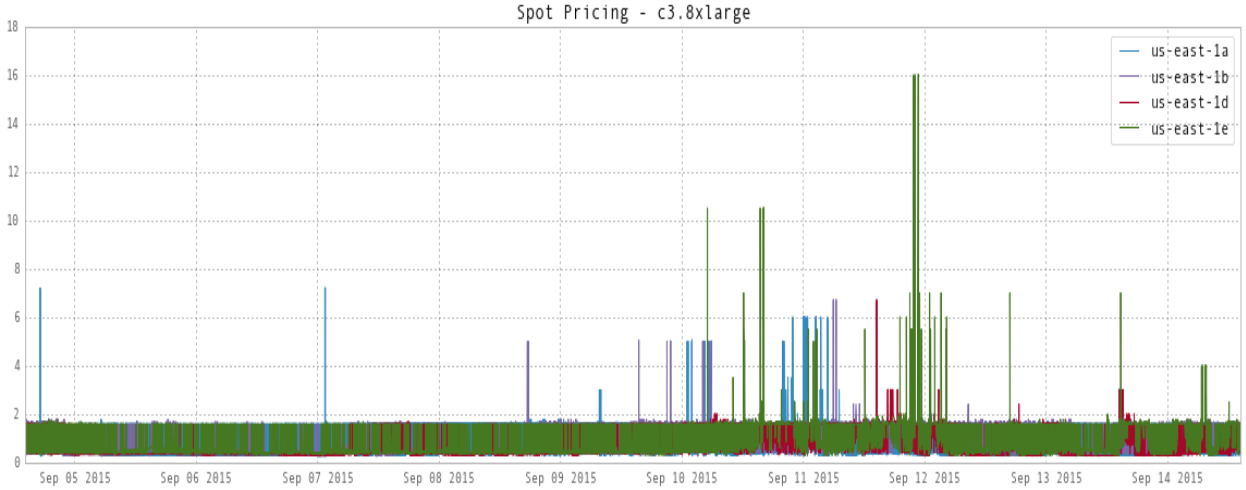
Results

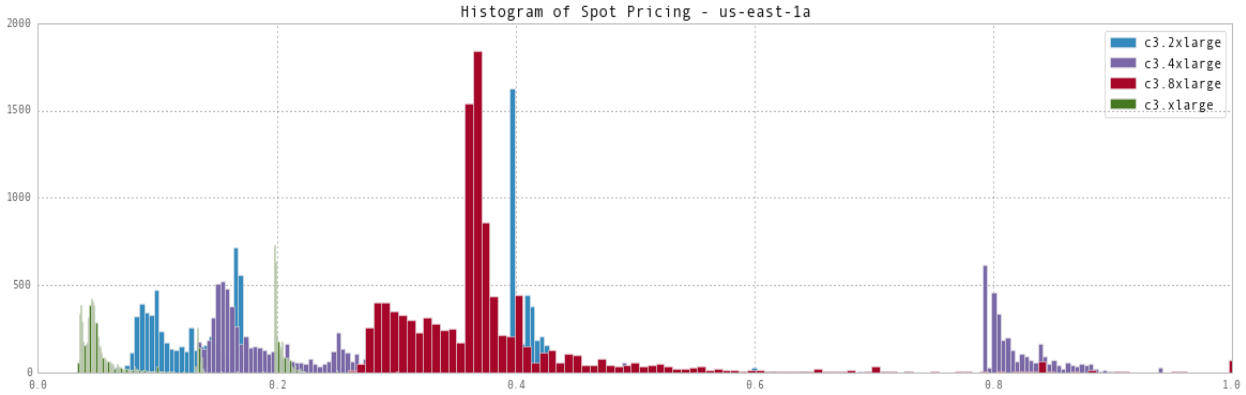
Measurements/Metrics tables

AvailabilityZone	SpotPrice
us-east-1a	0.000996
us-east-1b	0.000001
us-east-1c	0.000000
us-east-1d	0.000046
us-east-1e	0.000003



Visualizations.





Evaluation.

Simulation studies using real cloud workloads indicate that under dynamic workload conditions, we achieve higher net income than static allocation strategies and minimizes the average request waiting time.

Conclusion

Analyzing the historical spot pricing is easy using modern data science toolsets and very simple calculations and aggregations can lead to insight that can save you significant money.

Caution: The pricing changes week to week and even day to day. If Netflix decides they need to re-encode all their movies for the new AppleTV, then the price and availability statistics of the instances will change dramatically. Your mileage will vary and that's why it is important to build algorithms that can adapt to market fluxuations.

Future directions:

Use more advanced techniques to find recurring patterns in the data that you can then take advantage of.

Bibliography

Reference citations (Chicago style - AMS/AIP or ACM/IEEE).

[1] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "Analysis and lessons from a publicly available google cluster trace," University of California, Berkeley, CA, Tech. Rep, 2010.

[2] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," ACM SIGCOMM Computer Communication Review, 2008.

[3] "Aws developer forums: Amazon elastic compute cloud," https://forums.aws.amazon.com/forum.jspa?forumID=30.

[4] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," in IEEE Int. Conf. on Cloud Comp. (CLOUD), 2010.

[5] Navraj Chohan et. al., "See spot run: Using spot instances for mapreduce workflows," in USENIX HotCloud, 2010.

[6] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in Proceedings of the USENIX HotICE workshop, 2011.

[7] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, and K. Pentikousis, "Energy-efficient cloud computing," The Computer Journal, 2010.

Code & Data

Reference Documentation - Software commands, inputs, installation.

The toolset uses iPython, a web-based interactive programming environment similar to Mathematica or Matlab; Pandas for data manipulation; and Mathplotlib for graphing.

An AWS account.

Data Sources - Links, downloads, access information.

The spot pricing dataset is assumed to be in json in the data/ directory

Source Code - Listings, documentation, dependencies (open-source).

```
instance_types  = ['c3.xlarge', 'c3.2xlarge', 'c3.4xlarge', 'c3.8xlarge']
region = 'us-east-1'
number_of_days = 10

end = !date -u "+%Y-%m-%dT%H:%M:%S"
end = end[0]
start = !date -v "-{number_of_days}d" -u "+%Y-%m-%dT%H:%M:%S"
start = start[0]
print "will process from " + start + " to " + end
```

```
import sys
import boto as boto
import boto.ec2 as ec2
import datetime, time
import pandas as pd
import matplotlib.pyplot as plt
pd.set_option('display.mpl_style', 'default') # Make the graphs a bit
prettier
%pylab inline
%matplotlib inline

ec2 = boto.ec2.connect_to_region(region)
```

```
df = pd.read_json('../data/m1.xlarge.json')
df = df.set_index('Timestamp')
df[:5]
```

```
df.groupby('AvailabilityZone').var()


for key, grp in df.groupby(['AvailabilityZone'], as_index=False):
    plt.plot(grp['SpotPrice'], label=key)

plt.legend()
plt.title('Spot Pricing')
plt.show()


#
# process the output and convert to a dataframe
#

l = []
for instance in instance_types:
    sys.stdout.write("*** processing " + instance + " ***\n")
    sys.stdout.flush()
    prices = ec2.get_spot_price_history(start_time=start, end_time=end,
instance_type=instance)
    for price in prices:
        d = {'InstanceType': price.instance_type,
             'AvailabilityZone': price.availability_zone,
             'SpotPrice': price.price,
             'Timestamp': price.timestamp}
        l.append(d)
    next = prices.next_token
    while (next != ''):
        sys.stdout.write(".")
        sys.stdout.flush()
        prices = ec2.get_spot_price_history(start_time=start, end_time=end,
instance_type=instance,
                                            next_token=next )
        for price in prices:
            d = {'InstanceType': price.instance_type,
                 'AvailabilityZone': price.availability_zone,
                 'SpotPrice': price.price,
                 'Timestamp': price.timestamp}
            l.append(d)
        next = prices.next_token

    sys.stdout.write("\n")

df = pd.DataFrame(l)
df = df.set_index(pd.to_datetime(df['Timestamp']))




plt.figure(1, figsize(20,5))
```

```
for azName, azData in
df[df.InstanceType=='c3.8xlarge'].groupby(['AvailabilityZone'],
as_index=False):
    plt.plot(azData.index, azData['SpotPrice'],label=azName)
plt.legend()
plt.title('Spot Pricing - c3.8xlarge')
plt.show()
```

```
plt.figure(1, figsize(20,5))
for inName, inData in df[df.AvailabilityZone=='us-east-
1a'].groupby(['InstanceType'], as_index=False):
    plt.hist(inData['SpotPrice'], bins=1000,label=inName)
    plt.xlim([0,1])
plt.legend()
plt.title('Histogram of Spot Pricing - us-east-1a')
plt.show()
```

```
eight = df[df.InstanceType=='c3.xlarge']
eight.groupby(eight.index.hour).agg([mean, std]).plot(title='average price by
hour of day (UTC) for c3.xlarge')
```

```
eight = df[df.InstanceType=='c3.xlarge']
eight.groupby(eight.index.dayofweek).agg([mean, std]).plot(title='average
price by day of week (UTC) for c3.xlarge')
```