DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

Large-scale Image Processing using Amazon EC2 Spot Instances

Youngsol Koh; Purdue University; West Lafayette, IN, USA
Yung-Hsiang Lu; Purdue University; West Lafayette, IN, USA

Abstract

Many cloud providers such as Microsoft, Amazon, and Google offer scalable computing environment with pay-per-use. However, processing large-scale data using on-demand cloud instances may still be too costly. Archival data, unlike real-time streams, does not have strict time constraints. Thus, it does not require continuous processing and occasional suspension can be tolerated. Some cloud vendors (such as Amazon) introduces spot instances that use spare instances with dynamic pricing. Spot instances offer the same performance as on-demand instances at greatly reduced prices but spot instances may be terminated at short notice. As a result, processing programs may not finish when using spot instances. This paper introduces a cost-effective system to process large-scale image data using Amazon EC2 (Elastic Compute Cloud) spot instances and Amazon Simple Storage Service (S3). This system uses a check-pointing method to store progress so that processing can resume later if the spot instances are terminated. Even though using spot instances may prolong the total execution time, our experiments demonstrate that with appropriate bidding strategies, the execution time can be almost the same as using on-demand instances, while saving up to 85% cost.

Introduction

Image and video data is growing rapidly in recent years. It is common for a person to have multiple digital cameras, in a mobile phone, a laptop, and a desktop display. Furthermore, low-cost network cameras are being deployed to observe natural scenes, traffic, or to detect unauthorized entry of restricted areas. These digital cameras can produce vast amounts of data. The data may be processed in two ways: streaming or archive. Streaming data is processed while the data is captured, i.e,, real-time. Archival data is stored as files for offline processing. Archival data, unlike streaming data, does not have strict timing requirements. Thus, occasional delay in processing archival data may be tolerable.

Cloud computing [2] can be a cost-effective way to process large amounts of data. Cloud vendors usually charge by hours. The “pay-per-use” model allows users to pay for only the computational resources needed. On-demand cloud instances may still be too costly when processing large amounts of data. Some cloud vendors (such as Amazon Elastic Compute Cloud, EC2) provide *spot instances* which allow users to set bidding prices. The cost of a spot instance will never exceed the bidding price set by users. However, the availability of spot instances is not guaranteed. It is possible that a user obtains no spot instance when the market price exceeds the bidding price. Furthermore, when the market price rises above the bidding price, an allocated spot instance may be terminated at short notice. When this occurs, the running programs on the spot instance are stopped and the intermediate results are lost.

Many efforts have been taken to use spot instances due to the cost advantages. The financial company, AlphaSense, uses spot instances to process millions of documents and has reduced the cost of processing from 5,000 dollars to 80 dollars [6]. Taifi *et al.* [10] use spot instances for high performance computing (HPC) and develop message passing interface (MPI) in volatile environment. Ben-Yehuda *et al.* [1] analyze spot price mechanism to construct a price estimation model. Many papers focus on modeling spot prices based on the price history to prevent interruptions. This paper introduces a system to process large-scale archival data with possible interruptions.

This paper presents a system that uses Amazon EC2 spot instances to process large amounts of visual data. The processing may take many hours and the spot instances may be terminated before the programs finish. The system uses a check-pointing method to store the progress on Amazon S3 (Simple Storage Service). Sometimes before a spot instance is terminated, a warning is issued. When this system receives the warning, a check-point is created. The system also saves check-points periodically because such a warning is not always available. When a spot instance becomes available again, the system resumes processing from the latest check-point. We use Amazon S3 to store large-scale data and the intermediate results for several reasons. First, it is inefficient to upload large-scale data to EC2 because the data may be accessed via multiple instances or vanishes due to the interruptions. Second, the data transfer rate between EC2 and S3 is high and free of charge. One experiment [12] shows that the maximum throughput between EC2 and S3 is up to 50 MB/s.

This system is evaluated using a human detection program [3] implemented in OpenCV. The data is obtained by storing snapshots of public locations from five network cameras. Over 200,000 images (approximately 21GB) are analyzed and the processing time is 36 hours using on-demand EC2 c3.large instance without interruptions. When the system uses spot instances, on average, the execution time is nearly the same as using on-demand instances, up to 85% cost savings with proper bidding prices.

This paper has the following contributions: (1) The paper presents a system for processing large-scale image data using cloud spot instances. By setting the bidding prices strategically, this system can reduce the costs by 85% at less than 5% performance degradation. (2) The system performs periodic check-points to save the progress in S3 because spot instances may be terminated at short notice or no notice at all. This system demonstrates that spot instances can be a cost-effective option for processing large amounts of data.

Cloud Computing for Analyzing Archival Data

Cloud computing is a service that may reduce the efforts to manage computing resources such as networks, servers, and storage. It offers essential characteristics such as on-demand service,

DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

broad network access, resource pooling, rapid elasticity and measured service [4]. These characteristics release users from dealing with expensive investments in hardware. There are several benefits of using cloud computing for handling large-scale archival data. First, users do not have to concern about the size of data since cloud computing offers scalability and can easily increase or decrease the computing resources as well as storages. Second, users pay only the amounts of time when resources are used. It may significantly reduce the cost to handle archival data. Amazon EC2 is one of the cloud computing services. Many companies use Amazon EC2 to process large-scale data. According to case studies in Amazon Web Service (AWS) [6], Netflix, one of the largest video streaming companies, uses Amazon EC2 to provide media streaming services to more than 57 million people in nearly 50 countries. With scalability, Netflix is able to quickly employ thousands of servers as well as terabytes of storage in short amount of time. Netflix uses EC2 not only for streaming media but also analyzing users' streaming experience over 10 PB data. AlphaSense, a financial services company, uses spot instances to index millions of documents to provide useful information for investors and has reduced the cost of re-indexing of the large-scale dataset from 5,000 dollars to 80 dollars [6]. In this paper, we study large-scale image processing using Amazon EC2 spot instances. There are three types of services in Amazon EC2, on-demand instances, reserved instances, and spot instances.

- On-demand instances. Users pay only fixed hourly rates without commitments. They are more expensive than reserved instances and spot instances.
- Reserved instances. They require long-term commitments for a year or more. It offers a reduced rate up to 75% [7]. Users may pay upfront costs to reduce hourly rates [7].
- Spot instances. Users can specify bidding prices which are the maximum prices that users are willing to pay. Spot instances are less reliable than both on-demand and reserved instances. They can be terminated when the bidding prices are lower than the market prices or when EC2 does not have spare resources. Upon termination, all data in the instance vanishes.

Spot instances are launched when the bidding prices exceed the market prices. Since archival data does not need continuous processing, it may be the best option to reduce cost. Amazon usually issues warnings 2 minutes before spot instance interruptions. The intermediate results can be saved in S3 before the termination because there are no data transfer charge between EC2 and S3. The process to create spot instances is similar to create on-demand instances. Users can create the spot instance requests in Amazon web console. In the spot instance requests, users can specify the types of the instances, zones, and the bidding prices. Users can monitor the trend of the market prices. Amazon offers cloud services in multiple regions such as Europe, Asia, and America. In this paper, among 3 regions in America (North Virginia, North California, and Oregon), we specifically study using the spot instances in Oregon (us-west-2). There are 3 available zones in Oregon. Each zone has different pricing. If one of market prices of the zones is below the bidding prices, the spot instances will be launched in the zone that meets the market prices. Figure 1 shows the 18 hour history of c4.large spot instance on June 25, 2015 [8]. At 2pm, in us-west-2c, the price is below 0.020 dollars per hour while the others (2b and 2c) have the prices about 0.035 dollars per hour. By our observation, the price for c4.large instance in us-west-2c is usually the lowest price for the instance. The price for c4.large on-demand instance is 0.11 dollars per hour and the reserved instance is 0.082 dollars per hour without upfront payment. The cheapest available option in c4.large reserved instance is 0.0692 dollars per hour, still more expensive than the spot instance. When processing large-scale archival data, storage cost is also one of the important factors. Amazon S3 [9] is scalable object storage and users pay only the space used. There is no charge for data transfer between EC2 and S3. The data transfer rate between EC2 and S3 is fast [12]. When processing data in spot instances, it is possible to lose the data due to interruptions. It is inefficient to upload such large data every time. Thus, we choose S3 to store large-scale archival data.

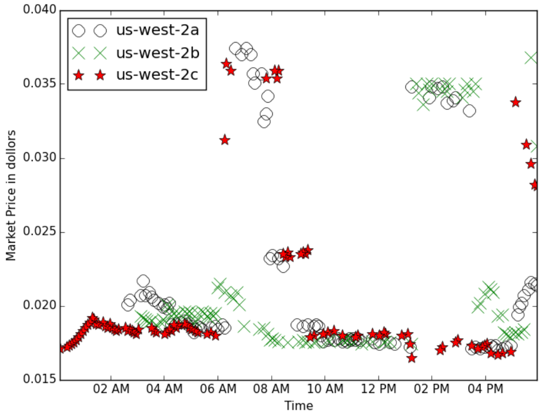

Figure 1. *Eighteen hour price history [8] of c4.large instance on June 25, 2015. Each zone has different pricing over time.*

Related Work on Spot Instances

Many papers conduct research on Amazon spot instances to reduce computation cost. Yi *et al.* [13] focus on adaptive price check-pointing methods to minimize costs and interruptions. Based on the price history of spot instances, the authors propose several check-pointing strategies such as hour-boundary check-pointing, rising edge driven check-pointing, check-pointing with adaptive decision and check-pointing combinations. They do not build a system to process data and the authors simulate the result based on the history of spot prices. Taifi *et al.* [10] establish models to predict running time of HPC applications based on previous spot price history. Ben-Yehuda *et al.* [1] analyze spot price mechanism to construct a price estimation model. The authors claim that spot prices are not market-driven as mentioned by Amazon rather the spot prices are randomly set within a tight price range in

Authors	Main focus
Yi [13]	Adaptive price check-pointing methods to minimize costs and interruptions
Taifi [10]	High performance computing and message passing interface tools for volatile environment
Ben-Yehuda [1]	Analysis on spot price mechanism to construct a price estimation model
This paper	System for large-scale image processing using spot instances with strategic bidding and a check-pointing method to save data

Table 1. Comparisons among the related work

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

a hidden reserve price mechanism. Many papers focus on modeling spot prices based on the price history to prevent interruptions. In contrast, this paper introduces a system that can handle interruptions by saving check-points.

Human Detection

There are various methods to detect humans. One method is using histogram of oriented gradients (HoG) developed by Dalal and Triggs [3]. The authors use linear SVM as a baseline classifier. To generate HoG descriptors, the authors divide an image into small cells and accumulate 1-D histogram of gradient directions in the cells. Contrast-normalization is performed to prevent illumination variation. The authors test the HoG method with two different data sets. One is the MIT pedestrian data set and the other is the INRIA. Tuzel *et al.* [11] use covariance matrices as object descriptors that do not lie on a vector space. These covariance matrices can be represented as a connected Reimannian manifold. The authors detect humans by classifying points in a Riemannian manifold and test their algorithms using the INRIA human database. Schwartz *et al.* [5] propose a human detection algorithm that utilizes multiple features such as edges, textures, and colors. The descriptors created by large amounts of information result in high-dimensional spaces. They employ Partial Least Squares (PLS) analysis to reduce dimensionality. The authors test the method with the INRIA, the DaimlerChrysler, and the ETHZ pedestrian datasets.

Authors	Method	Test Dataset	Images
Dalal [3]	Histogram of Oriented Gradients	MIT INRIA	200 1,805
Tuzel [11]	Classification on Riemannian Manifolds	INRIA	1,805
Schwartz [5]	Partial Least Squares	INRIA DaimlerChrysler ETHZ	N/A N/A N/A
This paper	Histogram of Oriented Gradients	Public cameras	200,000

Table 2. Comparisons of Human detection methods and datasets used for each method.

Method

System Overview

The system contains the following four components as illustrated in Figure 2.

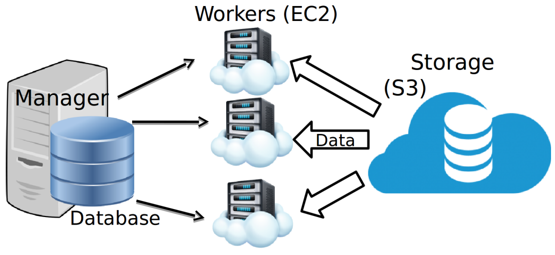

Figure 2. *Major components in the system. The manager sends meta-data to the workers. The meta-data is the information that the workers need to pull archival data from S3. All computations are performed in the workers.*

- Manager. The manager makes all decisions about the system, including what types of cloud instances to use (the number of cores and the amount of memory), and the bidding price. The manager tracks the states of the requested spot instances and sends the meta-data about the archive such as archive names.
- Worker. A worker is the computing engine where image processing occurs. A worker is a spot instance in EC2. The worker periodically stores check-points in S3.
- Database. The database stores the following information: the ID of each cloud instance and spot request, the assignments of the archival data to specific instances, and the state of the workers.
- Storage. The archival data is stored in S3 before processing because data transfer between S3 and EC2 is fast and free of charge. Eicken [12] performs the data transfer experiment. The results show the rate up to 12.6 MB/s for a single file download and the maximum throughput is 50 MB/s. Check-points are also stored here.

Please notice that the data does not pass through the manager nor the database. Instead, the data is moved between the EC2 workers and the S3 storage because of the available higher data rates.

Image Processing

The data is partitioned into five sets based on the sources (the snapshots from five network cameras are used). The system creates an operating system process for each set of data so that analysis can be performed in parallel when a cloud instance has multiple cores.

Check-point

When a spot instance is interrupted due to the low bidding price or insufficient capacity in EC2, Amazon issues a warning 2 minutes before the instance terminates. Each instance has its own meta-data such as hostname, instance id, and instance type. For a spot instance, it is possible to check the termination notice by accessing meta-data in the instance. If a spot instance is marked for termination, the meta-data will indicate the time it will be terminated. As the termination notice is issued, the system saves the progress on S3 after every image on each set of archival data is processed. The termination notice is not always provided. Thus, the system also saves intermediate results periodically. The progress will contain archive name and current frame information. Every instance has unique instance id. When a terminated spot instance is re-launched, the instance id is changed while the spot request id remains the same. From the spot request id, the system can find the new instance id and save it to the database to keep tracks of the instances. The system transfers the latest progress files from S3 to the spot instance. Each archival data resumes in parallel based on each progress file.

Instance Selection

Amazon offers a broad selection of instance types for different purposes. We use the human detection program from OpenCV that employs histogram of oriented gradient (HoG) descriptors to process data. Since it is computationally expensive, we select computation optimized instances shown in Table 3.

Bidding Strategy

We obtain the price history of the selected instances from Amazon. Amazon only publishes the price history for 3 months.

DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

Instance Type	vCPU	ECU	Memory (GiB)
c4.large	2	8	3.75
c4.xlarge	4	16	7.50
c4.2xlarge	8	31	15.00
c4.4xlarge	16	62	30.00
c4.8xlarge	32	132	60.00
c3.large	2	7	3.75
c3.xlarge	4	14	7.50
c3.2xlarge	8	28	15.00
c3.4xlarge	16	55	30.00
c3.8xlarge	32	108	60.00

Table 3. Computation optimized instances. vCPU is a hyperthread of an Intel Xeon core. ECU is Amazon EC2 Computing Unit that is relative measure of the integer processing power of an Amazon EC2 instance.

History of market prices for computation optimized instances are shown in Table 4 in page 5. A spot instance will be launched in one of the zones that satisfy the market price. Most of the average market prices are 75% lower than the on-demand prices. However, most of the median values are 80% cheaper than the on-demand prices. The average price is higher than the median price due to frequent spikes in the market prices. Thus, we decide to set the spot bidding price as 20% to maximize the cost reduction, 25%, and 50% to reduce the interruptions.

Evaluation

Dataset

This study uses snapshots obtained from five network cameras in public locations. The data is available to anyone connected to the Internet. A program is used to detect humans and count the number of people; for privacy, this study does not recognize faces. Figure 3 shows several examples of the snapshots. About forty thousand images are saved from each camera at one image every 10 seconds over 5 days. Totally, 200,000 images are used for this study.

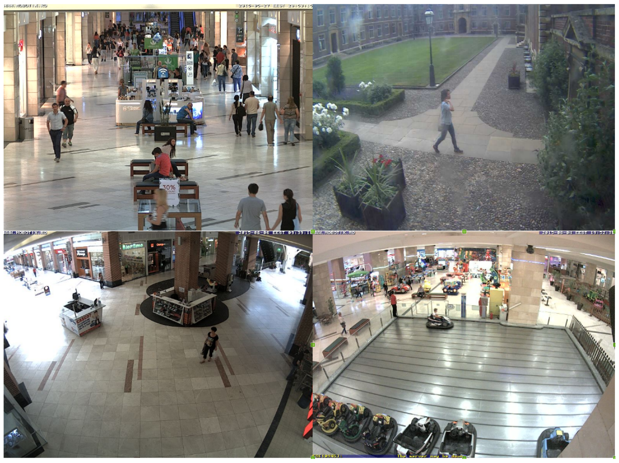

__Figure 3.__ The datasets used for evaluation. All the datasets are collected via public cameras. Except for the image on top right, cameras are located in shopping malls. The image on the top right is captured in a university.

Results

We can reduce the processing cost up to 90% using spot instances. Table 5 shows the results of the processing time and the cost when we use the on-demand instances. When we use the smallest instance, c3.large, to process the data, the execution time is more than 36 hours and the cost is 3.885 dollars. When the largest instance, c4.8xlarge, is used, it takes about 2 hours and 30 minutes and the cost is 7.052 dollars. For each spot instance, we select the bidding prices as 20%, 25%, and 50% of the on-demand instances based on the spot instance price history as shown in Table 4. Figure 4 shows the results of the execution time when the spot instances are used and Figure 5 shows the computation costs at the multiple bidding prices. The execution time is measured

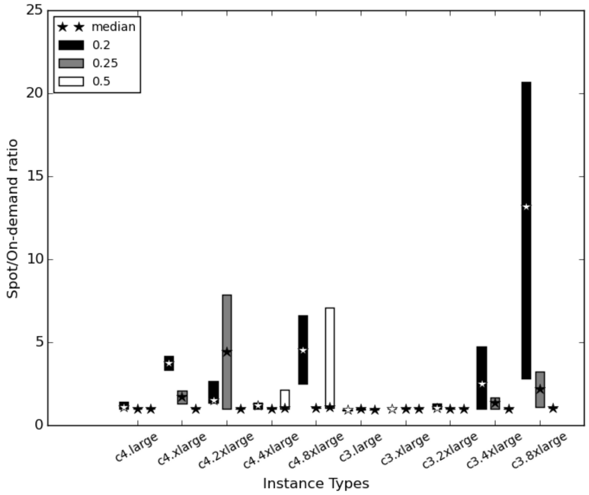

__Figure 4.__ The execution time of the workers at different bidding prices (0.2,0.25, and 0.5 of the on-demand instances). Horizontal axis: types of instance, vertical axis: execution time normalized to on-demand at different bidding prices.

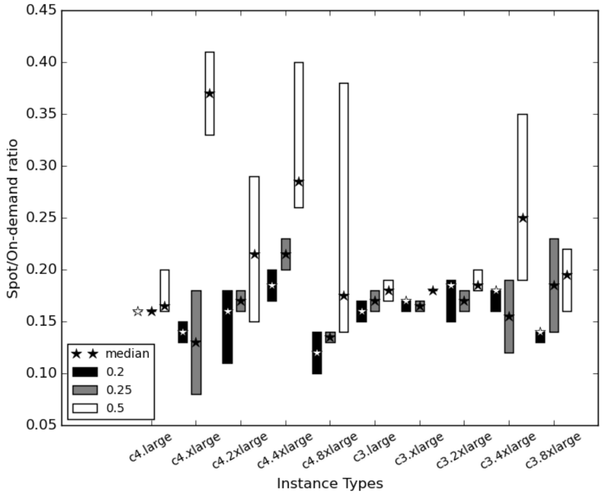

__Figure 5.__ The cost of the workers at different bidding prices (0.2,0.25, and 0.5 of the on-demand instances). Horizontal axis: types of instance, vertical axis: cost normalized to on-demand at different bidding prices.

DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

instance type	On-demand	us-west-2a			us-west-2b			us-west-2c		
	Price	mean (%d)	std	median (%d)	mean (%d)	std	median (%d)	mean (%d)	std	median (%d)
c4.large	0.110	0.064 (41.50)	0.178	0.018 (83.27)	0.034 (69.29)	0.060	0.019 (83.00)	0.028 (74.82)	0.039	0.019 (82.64)
c4.xlarge	0.220	0.049 (77.50)	0.086	0.039 (82.32)	0.053 (76.09)	0.065	0.040 (81.91)	0.047 (78.54)	0.051	0.040 (82.05)
c4.2xlarge	0.441	0.083 (81.12)	0.059	0.070 (84.20)	0.093 (78.85)	0.082	0.074 (83.24)	0.121 (72.46)	0.273	0.081 (81.54)
c4.4xlarge	0.882	0.205 (76.71)	0.200	0.153 (82.61)	0.221 (74.96)	0.183	0.161 (81.70)	0.240 (72.76)	0.234	0.160 (81.85)
c4.8xlarge	1.763	0.386 (78.11)	0.323	0.285 (83.86)	0.356 (79.81)	0.448	0.274 (84.48)	0.501 (71.58)	0.531	0.332 (81.16)
c3.large	0.105	0.028 (73.31)	0.025	0.025 (76.19)	0.027 (74.42)	0.024	0.023 (77.71)	0.030 (71.16)	0.026	0.026 (74.86)
c3.xlarge	0.210	0.063 (70.00)	0.160	0.040 (80.86)	0.061 (71.12)	0.128	0.040 (80.90)	0.062 (70.49)	0.087	0.045 (78.62)
c3.2xlarge	0.420	0.117 (72.19)	0.173	0.087 (79.36)	0.111 (73.62)	0.142	0.078 (81.50)	0.114 (72.76)	0.121	0.087 (79.21)
c3.4xlarge	0.840	0.213 (74.69)	0.364	0.156 (81.45)	0.182 (78.39)	0.148	0.151 (82.04)	0.205 (75.55)	0.293	0.151 (82.02)
c3.8xlarge	1.680	0.438 (73.95)	0.361	0.373 (77.80)	0.422 (74.87)	0.321	0.359 (78.61)	0.522 (68.95)	0.880	0.354 (78.90)

Table 4. Price history [8] of computation optimized instances from March to June. The table shows mean, standard deviation and median market prices of each zone. (%d) indicate the price difference compared to on-demand instance.

Instance Type	Execution Time	Cost
c4.large	30:56:40.55	3.410
c4.xlarge	16:07:00.78	3.740
c4.2xlarge	08:28:56.08	3.969
c4.4xlarge	04:32:23.22	4.410
c4.8xlarge	02:32:02.96	7.052
c3.large	36:26:36.15	3.885
c3.xlarge	18:36:33.49	3.990
c3.2xlarge	09:49:57.14	4.200
c3.4xlarge	05:13:43.79	5.040
c3.8xlarge	03:00:02.07	6.720

Table 5. The excution time and the cost of on-demand instances.

from the request of the spot instance to the completion of the program. It includes the waiting periods when the spot instances are not launched due to the low bidding prices or short in capacities in EC2 and interruptions. When we select the bidding prices as 50% of the on-demand instances, the execution time of most of the instances are nearly the same as the on-demand instances and the computation cost is only 15% of the cost of on-demand instances. The range of the cost is wide since the bidding price has more chance to meet the market price. For example, in c4.2xlarge instance, when the bidding price is 50%, the computation cost varies from 14% to 28% and its median is 21%. When the bidding price is 20% of the on-demand instance, the execution time increases up to 20 times of the on-demand instance. Table 6 shows the performance and the cost of c3.8xlarge instance. The instance completes the program 3 to 21 times slower when the bidding price is 20%. The median of the completion time is 13 times slower than the same type of the on-demand instance. While the performance of the instance varies in the wide range, the range of the cost is narrow. The minimum cost is 13% and the maximum cost is 14% of the on-demand instance. When we select the bidding price as 50% of the on-demand price, the execution time is nearly the same (4% to 6% times slower) as the on-demand instance. The minimum cost is 16% and the maximum cost is 22% of the on-demand instance.

Spot/On-demand	T_{min}	T_{med}	T_{max}	C_{min}	C_{med}	C_{max}
0.20	2.84	13.20	20.71	0.13	0.14	0.14
0.25	1.12	2.19	3.26	0.14	0.19	0.23
0.50	1.04	1.05	1.06	0.16	0.20	0.22

Table 6. Detailed result for c3.8xlarge instance. First column is the bidding price that is normalized by c3.8xlarge on-demand price. T stands for the time normalized to the on-demand execution time. C stands for the cost normalized to the on-demand cost. The subscripts, min, med, max are the minimum, median, and the maximum respectively.

Conclusion

We demonstrate a system for large-scale image processing using Amazon EC2 spot instances and S3. Using the spot instances, we can reach nearly the same performance as the on-demand instances with 85% reduced cost. With the check-pointing method, we successfully preserve intermediate results and are able to resume processing after interruptions. With the high bidding price, we can maintain the performance nearly the same as the on-demand instances. However, the cost varies in a wide range; it is still less than 50% of the on-demand instances. With the low bidding price, the execution time can be longer up to 20 times than the on-demand instances.

References

[1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation*, 1(3):16, 2013.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] P. Mell and T. Grance. The nist definition of cloud computing. 2011.

[5] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *Computer vision, IEEE 12th international conference on*, pages 24–31. IEEE, 2009.

[6] A. W. Service. All AWS case studies. `https://aws.amazon.com/solutions/case-studies/all/`.

[7] A. W. Service. Amazon EC2 Pricing. `https://aws.amazon.com/ec2/pricing/`.

[8] A. W. Service. Amazon EC2 Spot Instances Price History. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html`.

[9] A. W. Service. Amazon Simple Storage Service (S3). `https://aws.amazon.com/s3/`.

[10] M. Taifi, J. Y. Shi, and A. Khreishah. Spotmpi: a frame-

DOI: 10.2352/ISSN.2470-1173.2016.13.IQSP-226

work for auction-based hpc computing using amazon spot instances. In *Algorithms and Architectures for Parallel Processing*, pages 109–120. Springer, 2011.

[11] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1–8. IEEE, 2007.

[12] T. von Eicken. Network Performance Within Amazon EC2 and to Amazon S3. `http://www.rightscale.com/blog/cloud-industry-insights/network-performance-within-amazon-ec2-and-amazon-s3`.

[13] S. Yi, D. Kondo, and A. Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In *Cloud Computing (CLOUD),IEEE 3rd International Conference on*, pages 236–243. IEEE, 2010.