2012 IEEE Fifth International Conference on Cloud Computing

Increasing Spot instances reliability using dynamic scalability

Wesam Dawoud
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
wesam.dawoud@hpi.uni-potsdam.de

Ibrahim Takouna
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
ibrahim.takouna@hpi.uni-potsdam.de

Christoph Meinel
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
christoph.meinel@hpi.uni-potsdam.de

***Abstract*—Traditionally, Infrastructure as a Service (IaaS) providers deliver their services as *Reserved* or *On-Demand* instances. *Spot Instances* (SIs) is a complementary service that allows customers to bid on the free capacity at the provider data centers. Therefore, the decrease in the free capacity may result in terminating instances abruptly. To ensure fair trading, the provider does not charge customers for the interrupted partial hours. However, SIs price history traces analysis shows that uncharged time could rise up to 30% of the instance total run time, which means a reduction in the provider's profit.**

In this paper, we propose *Elastic Spot Instances* (ESIs) approach. It is a trade-off between the price and the total run time, where instead of abruptly terminating the SIs, the provider scales down their capacity proportionally to the increase in the price. Our approach delegates the task of interrupting the instances into the customers, but at the same time keeps the control on the provider side to isolate SIs' impact on the other services at overloaded time. Our approach doesn't imply an additional overhead or complex modification to current IaaS, while it consumes interfaces that are available by most of nowadays virtualization technologies.

***Keywords*-IaaS; Spot Instances; Reliability; Elasticity;**

I. Introduction

Amazon is the first cloud provider to come up with SIs [1] purchasing system to sell the spare capacity. The price of SIs changes dynamically according to free capacity and actual demand. Requests with bid price higher than or equal the current spot price will be served. On the other hand, if the current prices exceeded the user bid, provider will terminate out-of-bid instances abruptly. SIs reduce the prices from %38 to %44 of the *On-Demand* prices [2]. However, SIs customers are supposed to modify their applications to manage the abrupt termination of SIs.

To manage SIs termination, customers can implement fault tolerant architectures like MapReduce, Grid, Queue-Based, and Checkpointing [1]. The first three architectures imply major modification to customers' applications. On the other hand, Checkpointing is a simple traditional fault tolerant technique. It keeps application execution progress by storing the current state (i.e., snapshot) of the running instance into a persistent storage. Nevertheless, bad Checkpointing strategies could impact the performance drastically [3]. For instance, frequent Checkpointing results in a high cumulative overhead (i.e., computation is paused at checkpointing time). On the other hand, infrequent checkpointing results in a high overhead caused by the high recovery time (i.e., much computation should be repeated again).

The main goal of this paper is to reduce the Checkpointing overhead in SIs environment. This is motivated by the following facts: first, Checkpointing is a simple fault tolerant technique that does not require major modifications to customers' applications. Second, Checkpointing could be integrated to the other fault tolerant architectures to increase their reliability. Finally and most importantly, if customers have the control to get checkpoints exactly before terminating VMs instances (i.e., Optimum Checkpointing), then the provider can eliminate the concept of unpaid partial running hours, which on consequently increases the provider profit. In this paper, we investigate Amazon EC2 SIs. However, our approach is generic and increases the chance of the new cloud providers to compete in the market of cloud computing infrastructure.

In the next section, we study Amazon EC2 SIs implementation. In section III, we discuss our proposed ESIs approach. In section IV, we show the preliminary steps that are necessary to evaluate our approach. In section V, we present related work done to improve the trade-off between price, reliability, and total run time of applications on SIs. Finally, in section VI, we conclude and point out to our future work.

II. Amazon EC2 SIs

Amazon EC2 infrastructure is distributed into Regions. Each Region is separated into many Availability Zones to prevent failure propagation. This infrastructure mainly delivers *Reserved* and *On-Demand* Instances. The spare capacity is sold as SIs. The SIs, as well as the *Reserved* and *On-Demand* instance, could be one of many types depending on resources capacity (e.g., High-CPU Medium Instance "c1.medium", High-Memory Extra Large Instance "m2.xlarge", etc...).

Spot Instance price is determined by the type, the Region, and the operating system. Unlike Zhang et al.'s [4] assumption, in our approach we assume that a physical machine hosts only instances of the same type and operating system.

978-0-7695-4755-8/12 $26.00 © 2012 IEEE
DOI 10.1109/CLOUD.2012.58

We support our assumption by observing CPU architecture of each EC2 instance type.

III. Elastic Spot Instances (ESIs)

Current SIs implementation reacts with the increase demand on *On-Demand* and *Reserved* instances by increasing the SIs price. As a result, out-of-bid SIs are terminated abruptly. A simple solution is to apply high bids strategy and accept a higher total price. However, monitoring spot instances price history traces shows that even such a technique don not guarantee continues running of SIs.

Our propose ESIs approach is a trade-off between the price and the total running time. For instance, user can keep running instance few minutes more, even with a higher price, to finish a specific job within a deadline. At the same time, ESIs user can decide about the limit of the price after which he/she will terminate the VM instance safely (i.e., Checkpointing or planned terminating). The strength of our approach lies in delegating VM instances termination into the user without influencing the other services (i.e., *Reserved* and *On-Demand* instances) performance.

Implementing our approach requires the following modifications to current SIs purchasing algorithm: first, provider should determine min and max price for each instance type. Second, instead of terminating out-of-bid SIs, the provider scales down instances' capacity to a value proportional to the increase in the SIs price. Third, running instances will be charged per second, while terminating VM instances, at our approach, is the user decision. On the light of these modifications, algorithm calculates VM capacity according to user bid and current SI price.

Algorithm 1 ESIs's purchasing algorithm

Input: max_price, min_price, current_price, min_cap, and user_bid
Output: VM_capacity

// Calculate the scaling step size
$scale\_step \leftarrow 100/(1000 * (max\_price - min\_price) + 1)$
// Calculate next capacity of VM
if $user\_bid \geq current\_price$ **then**
  $VM\_capacity \leftarrow 100$
else
  if $user\_bid < current\_price$ **then**
    $VM\_capacity \leftarrow 100 - scale\_step * 1000 * (current\_price - user\_bid)$
  end if
  //To prevent VM from starving
  if $VM\_capacity < min\_cap$ **then**
    $VM\_capacity \leftarrow min\_cap$
  end if
end if

In addition to user bid and current price, the algorithm considers max and min price of SI's type. According to [2], the price history of most SIs types, except for some types in US-East data center, could be modeled as a Mixture of Gaussian distributions with three or four components with a high fit. This gives the impression that SIs already have soft minimum and maximum price thresholds for each instance type. Moreover, to prevent VM instances from starving, we propose having a minimum capacity of the VM resources. Actually, this value should be carefully calculated. In our extended research, we intend to study calculating this value depending on the individual physical hosts' utilization. This can exploit the free capacity of physical hosts within a high price zone.

Algorithm 1 shows that the provider will not have the control to terminate the SIs. At first glance, it may seem that customers will be complacent and can simply use a very low bid strategy to guarantee a continued run with a low price. However, if we take the example of "US-West, Linux, High-CPU Medium" instance, the probability density function shows that 99.8% of the prices fall between 0.076 and 0.084. Therefore, scale_step value in algorithm 1 is calculated as $100/(1000 * (0.084 - 0.076) + 1) = 11.11$, which means that whenever the Spot Price surpasses user bid with 0.001, the capacity of the instances will scale down to $(100 - 11.11) \approx 89$. If a user submitted a low bid, for example 0.077, he will be charged 0.077 per hour for a full capacity instance (i.e., 100%). However, when the market price jump to 0.081, the instance capacity will be scaled down to $100 - 11.11 * (0.081 - 0.077) \approx 56\%$. In spite of the fact that the instance still being charged 0.077 per hour, its capacity is scaled to almost 60%. By this concept, at some value of the SI price, even users with low bids have to decide between keeping VM instances running or terminating them safely.

IV. ESIs modeling

To evaluate our approach, we have to model the VM instance performance with different values of resources capacities. In this section, we model a VM instance runs on Xen 4.1 hypervisor. The physical server has 2.8 GHz Intel Quad Core i7 Processor and 8GB of physical memory. The workload is CPU-intensive workload generated by EP Embarrassing Parallel, which is one of NAS Parallel Benchmarks (NPB) [5]. The throughput is measured by Million Operations Per second (MOPs).

At the beginning, the VM instance runs with its full capacity (i.e., 100%). As seen in figure 1, the throughput is 37.92 MOPs and the execution time is 56.6 seconds. The same workload is run many times but for different capacities of the VM's CPU. In our experiment, we use Xen *Credit Scheduler* as an actuator for setting the CPU capacity limit of the VM. The *Credit Scheduler* has a non work-conserving mode, which prevents an overloaded VM

from consuming the whole CPU capacity of the host and consequently degrading the other VMs performance. For each CPU capacity, we recorded both the MOPs number and the total execution times.

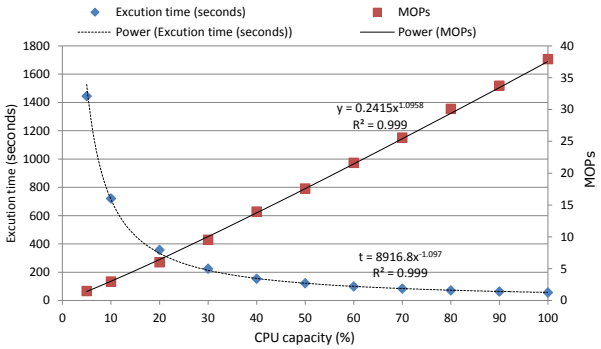

Figure 1. VM's model against CPU-intensive workload

As seen in figure 1 the instance's throughput changes linearly with the virtual CPU (vCPU) capacity according to the following equation:

$$0.2415 * x^{1.0958} \quad (1)$$

Where x is the vCPU capacity. A similar model should be built for each VM instance type. These models will be fed to our intended simulation.

V. Related work

Towards improving the trades off between the total price and the total run time of SIs, Andrzejak et al. [6] have proposed probabilistic decision model that considers user bid, budget, and the job deadline. However, it has been shown by Mazzucco et al. [7] that there is no correlation between Spot prices and the time. Moreover, the artifact changes in the SIs price [8] make it difficult to build consistent models that describe the SIs market behavior for the long run.

Yi et al. [3] employed Checkpointing and migration as fault tolerance techniques. They examined many Checkpointing strategies on the light of normalized $Price \times Time$ for different bid values and different types of instances. Moreover, after each instance's interruption, their approach decides the new type, location, and price that reduce the total running time.

In addition to Checkpointing and migration techniques, Voorsluys et al. [9] integrated job duplication technique. This integration increases the probability that jobs finish within their deadlines. However, as concluded by authors, job duplication yields much higher costs.

VI. Conclusion & Future work

The proposed ESIs architecture does not require many modifications to the current Cloud Computing Infrastructure. However, it has benefits for both of the provider and the customer. On the provider side, our approach increases the provider's revenue where it eliminates the concept of the partial hours. For the customer, the proposed approach boosts the Checkpointing strategy to the optimum level.

In this paper, we concentrate on CPU-intensive applications, where the CPU is the real player in power consumption. However, in the future, we will consider other resources and different combinations of the real workload. Our approach introduces new spot instances market, which should be clearly explained to the clients. Currently, we are implementing a simulator that validates our approach and help explaining the potential modification to current bidding strategies. Moreover, we study the impact of the ESIs approach on the other hosted services in the cloud infrastructure.

References

[1] "Amazon EC2 Spot Instances." [Online]. Available: http://aws.amazon.com/ec2/spot-instances/

[2] B. Javadi and R. Buyya, "Comprehensive Statistical Analysis and Modeling of Spot Instances in Public Cloud Environments," The University of Melbourne, Melbourne, Tech. Rep., 2011.

[3] S. Yi, A. Andrzejak, and D. Kondo, "Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances," *IEEE Transactions on Services Computing*, Jul. 2011.

[4] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," p. 1, Mar. 2011.

[5] NASA, "NAS Parallel Benchmarks (NPB)." [Online]. Available: http://www.nas.nasa.gov/Resources/Software/npb.html

[6] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, Aug. 2010, pp. 257–266.

[7] M. Mazzucco and M. Dumas, "Achieving Performance and Availability Guarantees with Spot Instances," in *13th International Conference on High Performance Computing and Communications (HPCC-2011)*, Banff (Canada).

[8] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and T. Dan, "Deconstructing Amazon EC2 Spot Instance Pricing," Technion Israel Institute of Technology, Haifa, Tech. Rep., 2011.

[9] W. Voorsluys and R. Buyya, "Reliable Provisioning of Spot Instances for Compute-intensive Applications," *Computing Research Repository*, vol. abs/1110.5, 2011.