A Novel Checkpointing Scheme for Amazon EC2 Spot Instances

Sunirmal Khatua[#1] and Nandini Mukherjee[#2]

[#]Department of Computer Science and Engineering, Jadavpur University, India

[1] skhatuacomp@caluniv.ac.in, [2] nmukherjee@cse.jdvu.ac.in

ABSTRACT

Amazon's spot instances allow customers to bid on unused Amazon EC2 capacity and run those instances for as long as their bid exceeds the current spot price. Customers may expect their services at lower cost with spot instances compared to on-demand or reserved. However, the reliability is compromised since the instances providing the service may become unavailable at any time. In this paper, we study various checkpointing schemes that can be used with spot instances. Also we devise some algorithms for checkpointing scheme on top of application-centric resource provisioning framework that increase the reliability while reducing the cost significantly.

INTRODUCTION

Amazon EC2 offers their computing resources in the form of on-demand, reserved and spot instances [3]. While on-demand and reserved instances has a fixed hourly charge, spot instances provide the ability for customers to purchase compute capacity with no upfront commitment and at a variable hourly rate with a customer-defined upper bound (bid) on the rate. Spot Instances are available only during the time when the spot price is below the customer defined bid.

Thus spot instances make the resources unreliable in nature and inappropriate for long running jobs like image processing, gene sequence analysis etc. At the same time, they offer the opportunity to accomplish such jobs at a much lower cost than on demand or reserved policies. Clearly, checkpointing (saving partially completed tasks to be resumed latter) may be a good option to make a tradeoff between the cost and reliability. In this paper, we investigate various checkpointing techniques for spot instances and evaluate their comparative performances.

RELATED WORK

An Application-centric resource provisioning framework along with the unified definition of an application is proposed in [1]. The framework analyzes each application and the available services from the providers to find optimal resource requirement for the application. It also monitors the deployed applications to reprovision the resources as the situation changes.

While [1] deals with a generalized multi-provider framework, various checkpointing schemes for EC2 spot instances are extensively studied in [2]. The key checkpointing schemes they propose include No Checkpointing (NONE), Optimal Checkpointing (OPT), Hourly Checkpointing (HOUR), Rising edge-driven Checkpointing (EDGE) and Adaptive Checkpointing (ADAPT). NONE does not take any checkpoint at all while OPT takes a checkpoint just prior to a failure. NONE and OPT provide two extreme results without any practical value. They are used to provide comparative study for the other realistic checkpointing schemes. HOUR takes a checkpoint just prior to the beginning of the next hour. EDGE takes a checkpoint at every increase of the current spot price. ADAPT analyzes the overhead of taking and skipping a checkpoint at a decision point and takes a checkpoint accordingly.

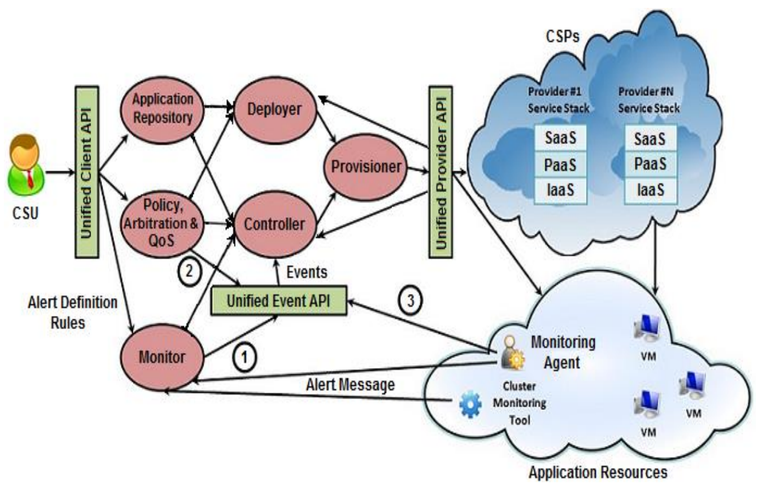

Fig. 1 : Application-centric Resource Provisioning Framework

A NOVEL CHECKPOINTING SCHEME FOR APPLICATION CENTRIC RESOURCE PROVISIONING

In this section we propose a novel checkpointing scheme for spot instances on top of application-centric resource provisioning framework, called Application Centric Checkpointing (ACC). The application-centric resource provisioning framework is shown in Fig 1.

In this paper, we devise a new event generation scheme for the framework that generates three events, namely E_{ckpt}, $E_{terminate}$ and E_{launch} to support spot instances. E_{ckpt} is used for taking checkpoint, $E_{terminate}$ is used to terminate a spot instance forcefully and E_{launch} is used to relaunch a previously terminated spot instance. We define two bid values for the purpose - one for the application (A_{bid}) and other for the spot instance (S_{bid}). S_{bid} is sufficiently large and is used in the request for spot instance. Clearly, the value is maintained at such a high level, that Amazon will never terminate the spot instances due to out-of-bid situation. On the other hand, A_{bid} is used by the monitoring subsystem to maintain user's budget.

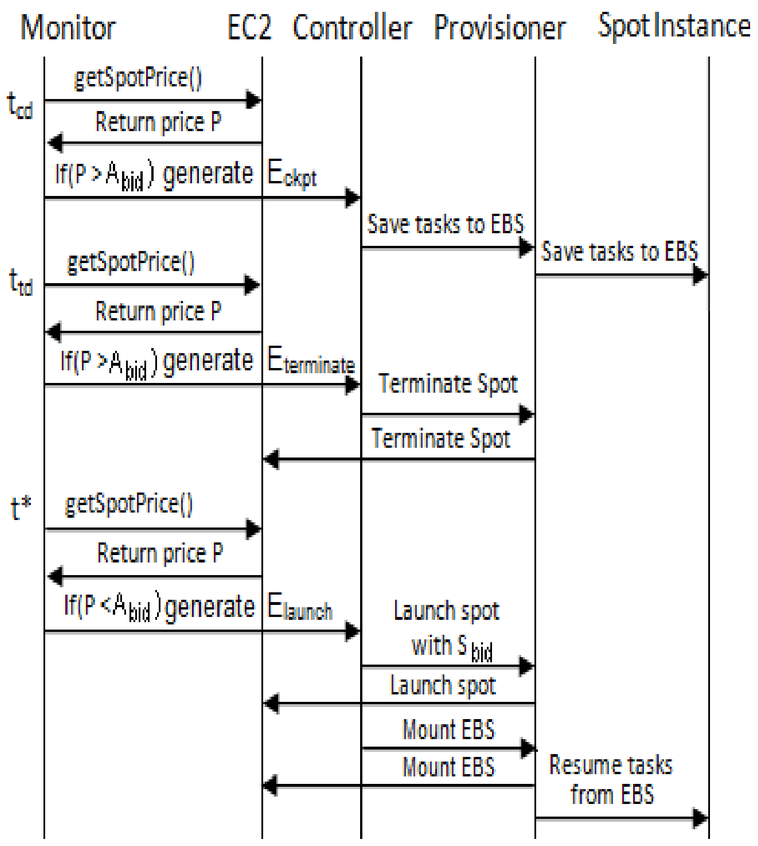

Fig. 2 : Application Centric Checkpointing Scheme

The ACC scheme is depicted in Fig 2 where the monitoring subsystem generates the above mentioned events at various decision points. Since the cost of spot instance is not changed during an instance hour and is fixed at the beginning of that instance hour, the decision points should be relative to the beginning of the next instance hour. Accordingly we define two decision points just prior to each hour boundary. Considering t_h as an hour boundary, the decision points for checkpoint (t_{cd}) and terminate (t_{td}) are defined as $t_{cd}=t_h-t_c-t_w$ and $t_{td}=t_h-t_w$. Here t_c is the time needed to take a checkpoint and t_w is the waiting time to get the current spot price. The Monitor module will generate E_{ckpt} at t_{cd} if the current spot price exceeds A_{bid} and will generate $E_{terminate}$ at t_{td} if the current spot price is still above the A_{bid}. It will generate E_{launch} at the start of each available period of a spot instance with respect to A_{bid}.

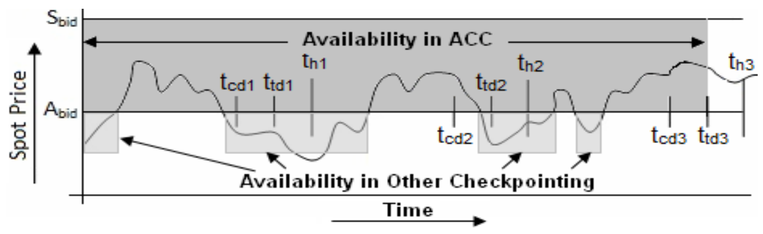

Fig. 3 : Availability of a spot instance

The novelty of the scheme is illustrate in Fig 3. ACC will generate neither E_{ckpt} nor $E_{terminate}$ for the hour boundary t_{h1} since the current spot price is below A_{bid} at both the decision points. That means, it will neither take a checkpoint nor terminate the spot instance at t_{h1}. It will generate E_{ckpt} but not $E_{terminate}$ for the hour boundary t_{h2} since the current spot price is above A_{bid} at t_{cd2} and below A_{bid} at t_{td2}. That means, it will take a checkpoint but will not terminate the spot instance at t_{h2}. Similarly, for the hour boundary t_{h3}, it will generate both E_{ckpt} and $E_{terminate}$ since the user will have to pay above A_{bid} for the next hour. So, it will take a checkpoint as well as terminate the spot instance at t_{h3}. Clearly, availability is increased and more continuous in ACC compared to other checkointing schemes as shown in Fig. 3.

IMPLEMENTATION AND EVALUATION

In this section, we analyze and compare our proposed ACC checkpointing scheme with the existing checkpointing schemes. The experiments have been carried out on 64 spot instance types using same data set, parameters, algorithms and assumptions used in [2]. The metrics used for this purpose include job completion time, total monetary cost and the product of monetary cost & completion time as the basis for comparison.

Fig 4, Fig 5 and Fig 6 show the comparative study of ACC with other checkpointing schemes for the jobs under different user's bid (A_{bid}) from \$0.401 to \$0.441. The result shows that ACC outperforms all the practical checkpointing schemes. Even ACC scheme reduces this product metric over OPT (theoretical optimal claimed in [2]) by an average value of 5.56% over the OPT scheme.

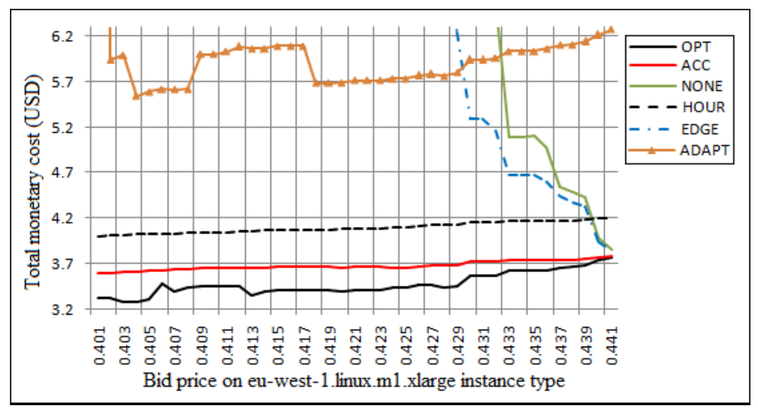

Fig. 4 : Total monetary cost of Job completion

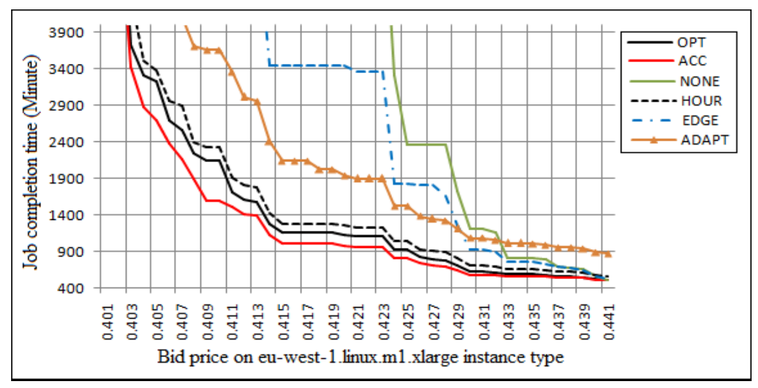

Fig. 5 : Job completion Time

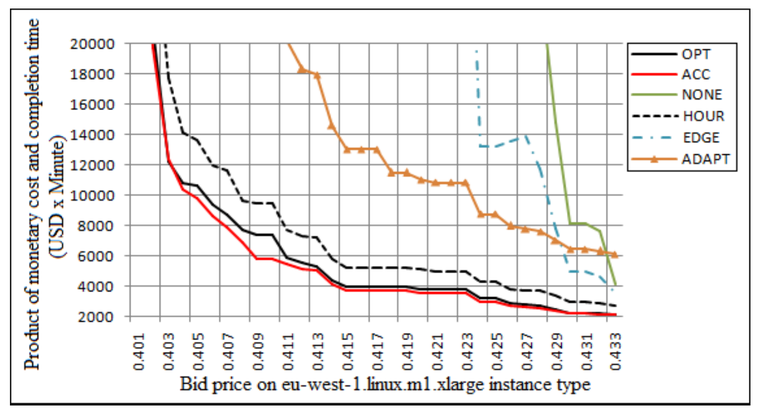

Fig. 6 : Product of total cost and completion time

The ACC scheme outperforms other checkpointing schemes due to the fact that it allows the jobs to be continued even when the spot price exceeds A_{bid} without affecting the job completion cost as shown in Fig 3.

CONCLUSION AND FUTURE WORK

Checkpointing plays an important role in reliability of job execution over EC2 spot instances. In this paper, we propose a checkpointing scheme on top of application-centric resource provisioning framework that not only increases the reliability but also reduces the cost significantly over the existing checkpointing schemes. In future, we want to investigate more on finding the optimal bid (A_{bid}) and the corresponding instance type for a given job.

REFERENCES

[1] S. Khatua, A. Ghosh and N. Mukherjee. Application-centric Cloud managemengt. In 9th IEEE/ACS International Conference on Computer Systems and Application

[2] S. Yi, A. Andrzejak, D. Kondo. Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances. In IEEE Transactions on Services Computing 2011. Volume: PP. Issue: 99.

[3] Amazon EC2 Purchasing Options. Available from: http://aws.amazon.com/ec2/purchasing-options/