Hybrid Spot Instance based Resource Provisioning Strategy in Dynamic Cloud Environment

Naidila Sadashiv
Dept. of Computer Sci. and Engg.
Acharya Institute of Technology
Bangalore, India 560 107
Email: sadashiv@acharya.ac.in

Dilip Kumar S M
Dept. of Computer Sci. and Engg.
University Visvesvarya College of Engg.
Bangalore, India 560 001
Email: dilipkumarsm@gmail.com

R S Goudar
Redknee
Bangalore, India 560 045
Email: rsgoudar1@gmail.com

***Abstract*—Utilization of resources to the maximum extent in large scale distributed cloud environment is a major challenge due to the nature of cloud. Spot Instances in the Amazon Elastic Compute Cloud (EC2) are provisioned based on highest bid with no guarantee of task completion but incurs the overhead of longer task execution time and price. The paper demonstrates the last partial hour and cost overhead that can be avoided by the proposed strategy of Hybrid Spot Instance. It aims to provide reliable service to the ongoing task so as to complete the execution without abruptly interrupting the long running tasks by redefining the bid price. The strategy also considers that on-demand resource services can be acquired when spot price crosses on-demand price and thereby availing high reliability. This will overcome the overhead involved during checkpointing, restarting and workload migration as in the existing system, leading to efficient resources usage for both the providers and users. Service providers revenue is carefully optimized by eliminating the free issue of last partial hour which is a taxing factor for the provider. Simulation carried out based on real time price of various instances considering heterogenous applications shows that the number of out-of-bid scenarios can be reduced largely which leads to the increased number of task completion. Checkpointing is also minimized maximally due to which the overhead associated with it is reduced. This resource provisioning strategy aims to provide preference to existing customers and the task which are nearing the execution completion.**

***Index Terms*—Cloud Computing; Resource Provisioning; Spot Instances; Bidding; Checkpointing; Reliability;**

I. Introduction

Cloud computing is an emerging paradigm that has revolutionized the consumption model of resources. With this paradigm, customers are able to access resources and services on the fly based on a pay-as-you-go model. Resources can also be booked in advance in order to be assured of the availability of desired resources. This is similar to market oriented system where maximizing the profit is the objective. Due to the dynamic nature of cloud computing where in the demand and supply of resource is difficult to predict, many of the resource are left *idle*. In order to utilize the idle resources efficiently, Spot Instance (SI) is introduced by Amazon [1].

Amazon is the pioneer in the introduction of the Spot Instance. The aim of SI is to generate revenue from the under utilized resources and there by consider the economy aware applications. Users who seek the service make a bid based on the current SI price that is publicly available for different type of instances. These instances are launched using the highest bid strategy. If the new bidding price is more than the current spot price, then the current user's Spot Instances will be terminated in no time and is termed as out-of-bid. Hence the pricing of these resources are volatile and depends on the supply and demand in the market. In such a dynamic pricing environment which is prone to out-of-bid situation shall involve fault tolerant techniques. Most commonly used fault tolerant techniques are migration, job duplication and checkpointing-based approach. They involve in saving the state of an application or a process during execution and restoring the saved state, for which a large storage capacity and time is required [6], [14], [15], [17]. Fault tolerance is achieved by these approaches but at a very high cost which should be shared by the providers as well as users [3], [16]. Users are attracted to the minimum cost of the resource but end up paying more, and sometimes it exceeds the on-demand price. Varying types of applications like economy aware applications, compute intensive, as well in data analytic scenarios, such as execution of MapReduce tasks [9], [14] and HTC are now aiming to reap the benefits of SI. Hence providers should provide more flexibility such that users are given a chance for their task completion rather than abrupt task termination [11], [19]. Users owe these resources at nominal price but not for free of cost.

Our main objective is to efficiently utilize the idle resources by avoiding the overhead during checkpointing and out-of-bid time. In this paper we propose a Hybrid Spot Instance (HSI) resource provisioning strategy that gives a chance to the existing users to continue with resource usage at out-of-bid time by redefining the running task's bid. We also present the overhead that the service provider faces which will be addressed by using the HSI. User's task will be migrated to on-demand scheme when SI price meets the on-demand price.

The rest of this paper is organized as follows. In Section II we discuss some of the recent related works and highlight their limitations and drawbacks. In Section III we present the Hybrid Spot Instance algorithm. In Section IV we show the simulation results with discussion and in Section V we present the concluding remarks.

978-1-4799-5958-7/14/$31.00 ©2014 IEEE

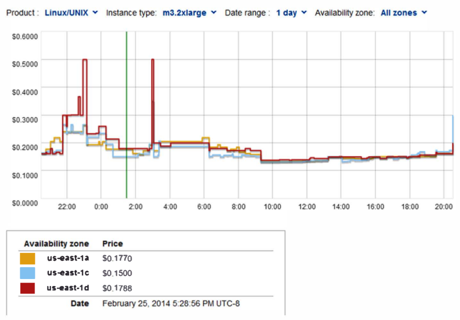
II. RELATED WORK

In this section, we discuss some of the research works related to resource allocation, performance and cost benefits with more focus on SI bidding for different types of applications in Cloud Environment.

Andrzejak et al. [2] have proposed a decision model based on probabilistic approach to help users to make a bid for the resources such that required budget or a deadline is meet. The work in [5] proposes an elastic spot instances where instead of abruptly terminating the SIs, the provider scales down their capacity proportionally to the increase in the price. The authors in [13] have proposed a bidding strategies for Spot Instances to minimize the cost and volatility of resource provisioning with the help of a constrained Markov decision. To address the problem of dynamically fluctuating resource demands, many resource allocation policies have been proposed [4], [7], [8], [10], [18], [20] wherein cloud resources may be offered as distinct types of virtual machines. The challenge faced in this environment is the allocation of data center resources to each spot market and thus gain profit.

These approaches discusses about predicting the optimal bid price for task completion and maximizing the profit. The limitations in these cases is finding the relationship between past and future failures or availability for setting up the optimal price. Also, profit maximization under some assumptions are considered with less or no focus on user reliability and the price for resource consumption. Users need to make bid such that they get resources for task completion and for this some hints on bidding value will be valuable. In [12] a job is modeled as a deadline constraint fixed computation oriented for which a dynamic bidding policy is proposed that minimizes the average cost of job completion. This dynamic bidding algorithm is compared with average bid and random bid policies. Analytical and closed-form results are obtained through a Markov spot price evolution. The authors in [17] have compared several static checkpointing schemes in terms of both price and task completion time and proposed a dynamic checkpointing strategies based on hour-boundary checkpointing, rising edge-driven checkpointing, checkpointing with adaptive decision and checkpointing combinations. Partial improvement based on the delayed termination is done which reduce the monetary cost, while improving reliability. Another bidding based approach is a multifaceted resource provisioning approach which estimates the future spot prices. Fault tolerance techniques namely migration, job duplication and their comparison with checkpointing in terms of deadline violations are discussed in [14]. Even though the cost as suggested may be lowered, it can be reduced much drastically by minimizing or avoiding the checkpointing.

In view of the above related works, some have considered profit maximization, timely completion of task, fault tolerance or providing reliable system but not as a cohesive model. This has motivated to propose a Hybrid Spot Instance approach to address these along with providing preference to the existing SI user. Last partial hour which is not charged to the user, is a burden for provider and is eliminated by the proposed HSI.


Product : Linux/UNIX Instance type: m3.2xlarge Date range : 1 day Availability zone: All zones

	Availability zone	Price
🟧	us-east-1a	$0.1770
🟦	us-east-1c	$0.1500
🟥	us-east-1d	$0.1788
	Date	February 25, 2014 5:28:56 PM UTC-8

Fig. 1. Spot Instance Price for m3.2xlarge
Courtesy: Amazon EC2 [1]

III. SPOT INSTANCE IN AMAZON EC2

A. Spot Instance

Amazon provides instances in 8 different regions for Linux/Unix, Windows and SUSE Linux enterprise server [1]. The instances are categorized as general purpose, compute optimized, memory optimized and micro instances. Unused resources under Amazon are shelled out based on highest bidding. Spot instance price is dynamic in nature which is affected based on the supply and demand in the market. Spot prices for the range of instance in different regions under different product is freely made as shown in Fig. 1. for Linux/Unix product of instance type m3.2xlarge. The following are the characteristics of Amazon EC2 SI:

- User is allowed to consume the SI resource when user bid price is more than spot price.
- Amazon EC2 charges on *hourly basis* for resource consumption.
- When the spot instance price is more than the current users bid then the tasks are abruptly terminated.
- Last partial hour is not charged for the customer if user is out-of-bid.
- Last partial hour is charged as full hour if user terminates the SI.
- No guaranty of the Quality of Service.

B. Why Checkpointing?

Spot prices are dynamic in nature due to which out-of-bid scenario becomes very common. Out-of-bid occurs when user bid price is less that the bid made for the spot price. Fig. 2.a. shows the scenario when out-of-bid occurs. Amazon does not charge for the last partial hour rather it is given for free to the out-of-bid user. The number of last partial hour, its associated cost and the number of tasks that are abruptly terminated during the time an application execution faces an out-of-bid situation is shown in Fig. 4 through 5 considering two different instance type. Algorithm 1 explains the working of spot instances and Algorithm 2 explains how the last partial hour and its cost is computed. During this scenario, the task under execution gets terminated and its current state will be

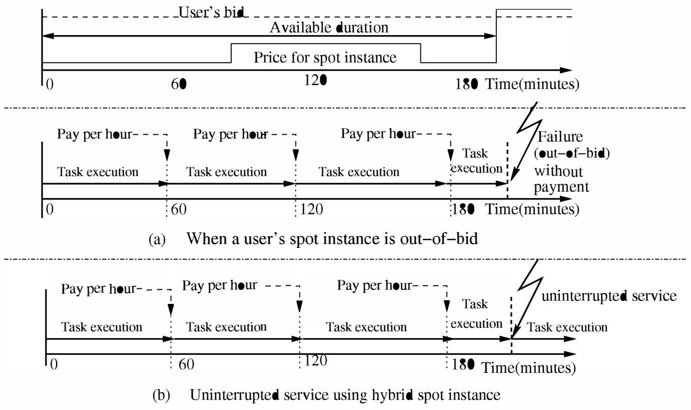


Fig. 2. Task execution with and without using HSI

lost. To avoid this, checkpointing is done to save the state and will be used later. Different kinds of checkpointing are been carried out at feasible time to avoid the loss. By taking up these checkpointing strategies, fault tolerance can be obtained to a larger extent which involves time, cost and storage overhead to both the providers and users. Tasks that have executed for a long time and which are about to be completed will be affected very much even if checkpointing is used. If checkpointing is performed just before out-of-bid, then it will be useful but it is difficult to predict the time of failure.

Algorithm 1: Spot Instance

```
Data: active-spot-instance, max-bid, user-bid-price,
      stretch-bid-price, stretch-bid-flag,
      stretch-till-on-demand, on-demand-price,
      hybrid-si-flag, spot-price
Result: cost, task-completed, time-consumed, Out-of-Bid
1  while user-task and active-spot-instance do
2      if user-bid-price $\geq$ spot-price then
3          task-remaining-time– ;
4          time-consumed++ ;
5          if time-consumed %60==0 then
6              cost+= user-bid-price ;
7          if task-remaining-time==0 then
8              task-completed++ ;
9      Out-of-Bid=1 ;
10     terminate-si() //Terminate the task on the instance ;
11     Partial Hour Overhead() ;
12 terminate-si() //Terminate the task on the instance ;
```

C. Hybrid Spot Instance based Resource Provisioning

We propose HSI based resource provisioning and fault tolerant approach to increase the reliability of existing Spot Instance users by allowing them to continue at the time of out-of-bid scenario as shown in Fig. 2.b. When the user requests for the Spot Instance, based on the price history they need to specify if service continuation is required when out-of-bid

Algorithm 2: Partial Hour Overhead

```
Data: spot-price, task-remaining-time
Result: partial-hour-cost, partial-hour
1  while Out-of-Bid AND active-spot-instance do
2      if task-remaining-time $\neq$ 0 then
3          partial-hour+=time-consumed % 60 ;
4          partial-hour-cost+=((time-consumed % 60)
           spot-price) $\div$ 60 ;
```

situation occurs by stretching the user bid till checkpointing is done or redefine the bid value till it reaches on-demand price. Similar to this is dynamic bidding that is used in ebay to prevent rebidding for products.

Algorithm 3: Hybrid Spot Instance

```
Data: active-spot-instance, user-bid-price,
      stretch-bid-price, stretch-bid-flag,
      stretch-till-on-demand, on-demand-price,
      hybrid-si-flag, spot-price, second-chance=1
Result: user-bid-price
1  if Out-of-Bid AND active-spot-instance then
2      while hybrid-si-flag AND active-spot-instance do
3          if old-user OR smaller-task OR second-chance
           then
4              second-chance = 0 ;
5              if stretch-bid-flag then
6                  if user-bid-price $\leq$ stretch-bid-price and
                   spot-price $\leq$ stretch-bid-price then
7                      user-bid-price = spot-price ;
8                      Compute-Price() ;
9                      hybrid-si-flag = YES ;
10                 hybrid-si-flag = NO ;
11                 terminate-si() ;
12             if stretch-till-on-demand then
13                 if user-bid-price $\leq$ on-demand-price then
14                     user-bid-price = spot-price ;
15                     Compute-Price() ;
16                 if user-bid-price $\geq$ on-demand-price then
17                     spot-price-cross-on-demand-price++ ;
18                     break ;
19         terminate-si() //One chance of avoiding
           Out-of-Bid over;
20 terminate-si() //Terminate the task on the instance ;
```

Hybrid Spot Instance is launched according to the request. During the task execution if HSI user faces out-of-bid situation then a minimum of one chance of updating the user bid is allowed. Loyal customers and the tasks that are nearing completion are given the preference to update their bid value so that they are in-bid. This strategy will ensure that the task is

Algorithm 4: Compute-Price

```
Data: user-bid-price, spot-price, task-remaining-time
Result: cost, time-consumed, task-completed
1 while user-task AND active-spot-instance do
2     task-remaining-time– ;
3     time-consumed++ ;
4     if time-consumed % 60 == 0 then
5         cost+= user-bid-price ;
6     if task-remaining-time == 0 then
7         task-completed++;
```

not terminated immediately but checkpointing will be done to save the state. This overcomes the overhead of performing frequent checkpointing or no-checkpointing which leads to loss of state. In case the user opts for redefining the user bid, then for every out-of-bid user bid is updated with spot bid which continues till the user bid reaches on-demand price as given in Algorithm 3. This strategy achieves fault tolerance and reliable services. Tasks will be completed without break and within the expected time. If the redefined bid crosses the regular on-demand price, then the user will be shifted to on-demand category of service which ensures better reliability than the regular Spot Instances. Algorithm 4 discusses about how the user is charged for the usage of hybrid based Spot Instance. By using HSI, abrupt task termination is avoided and also checkpointing can be minimized maximally. This strategy will be useful specially if the task is in its completion stage, as the waiting time and cost of getting the new instance may be more than the updated spot price. The strategy discussed is for Spot Instances from Amazon since their price history is freely available. However it can be implemented for other service providers in order to utilize their idle resources efficiently.

IV. Performance Evaluation

In this section, we evaluate the proposed Hybrid Spot Instance based resource provisioning and fault tolerant strategy with the effect of its mechanism, using trace-driven simulations. We have considered the real-time price history of Amazon EC2 Instances [1] for the simulation as Spot Instances. Simulation is carried out considering 100 heterogeneous applications which vary in size and the time required for completion is 100 - 500 minutes. The parameters considered are number of tasks completed, time taken for completion, price involved and number of out-of-bid situation avoided by considering the HSI for UNIX/Linux m1.small (EC 1), and m2.2xlarge (EC 41) instances.

A. Last Partial Hour Overhead

Amazon charges for the Instances on *hourly basis*. When out-of-bid occurs, the last partial hour is not charged to the user but rather given for free. This is an overhead to the provider. Fig. 3. shows the number of hours that is an overhead to the provider at the particular spot bid while running 100 applications on instance of type m1.small (EC 1). The cost

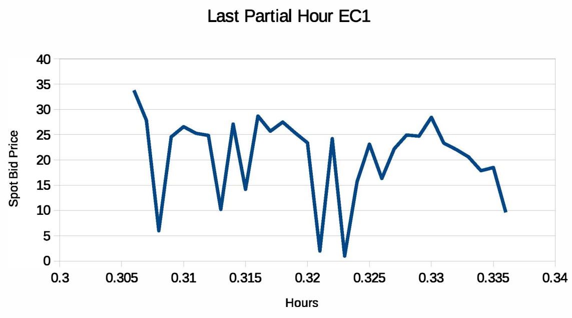


Fig. 3. Number of partial hours for 100 applications on EC1

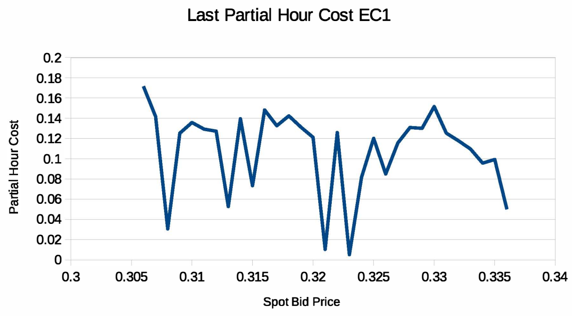


Fig. 4. Cost of last partial hour for 100 applications on EC1

associated for those last partial hour is shown in Fig. 4. Fig. 5. shows the number of tasks that are out-of-bid and hence get terminated abruptly.

B. Simulation Results

In the following, we show the impact of using the proposed fault tolerance strategy on the performance of the task and compare it with checkpointing based strategies namely : *base checkpointing, hourly checkpointing and no checkpointing*. In base checkpointing, just before out-of-bid the checkpointing is performed. At the end of every hour checkpointing is carried out in case of hourly checkpointing and it is totally absent in case of no-checkpointing. Applications of both fixed size and variable size are executed on different EC2 instances. The

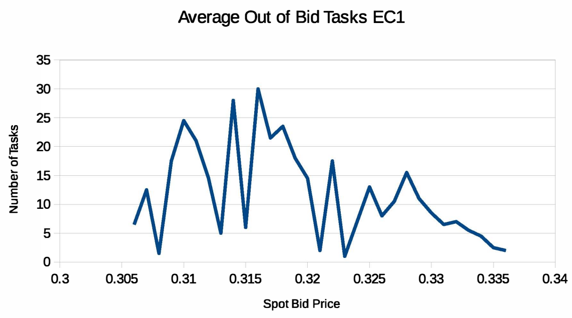


Fig. 5. Number of out-bid-count for 100 applications on EC1

parameters considered for the simulation are as below.

Number of Tasks Completed: HSI allows the tasks to continue the execution after out-of-bid condition is reached by redefining the user bid. As a result, the number of tasks getting executed is more. If no checkpointing scheme is used, the tasks are abruptly interrupted and have to restart for which the user has to invest for resources which violates the reason for using the SI. The Fig. 6 shows the comparison of HSI results with checkpointing based on base strategy where in the checkpointing is taken just before out-of-bid occurs, hourly checkpointing and when no checkpointing strategy is used on instance of type m1.small (EC 1). Assumptions regarding the resource availability immediately after the task is out-of-bid is done in [17]. Contrary to this assumption is, if the resources are available at out-of-bid time at the same bid price then the task should not have been interrupted. However, the number of tasks completed on instance of type m2.2xlarge (EC 42) is shown in Fig. 7. The comparison shows that the number of tasks completed is almost similar to other strategies.

Time Taken for Tasks Completion: The time taken for execution of the 100 tasks under instance of type m1.small (EC 1) and instance of type m2.2xlarge (EC 42) is shown in Fig. 8. Execution time of each task is 500 minutes. We have simulated for 200 and 500 tasks having different execution time but not reproduced here due to space limitations. The task is executed on more than 40 different categories of EC2 instances and have computed the time required for execution. According to HSI existing tasks are given more preference and hence are not interrupted rather user bid redefined. This leads to task completion within the actual required time of 500 minutes as observed from Fig. 9. Using base, hourly and no checkpoint strategies, the time taken for task completion is more with large difference when compared hybrid strategy. In HSI, resources are used without break which eliminates the burden both on the provider and the user regarding checkpointing and maintenance cost.

Cost for Tasks Completion: Cost for task execution on instance of type m1.small (EC 1) is shown in Fig. 10. It is observed that the cost of execution is less incase of HSI than using other checkpointing strategy. This difference is due to the fact that out-of-bid is followed by rebid process in case of HSI. Depending on the demand for the resource, the new instance will be assigned which will consume time and cost to the existing user in case of non HSI strategies. In case of hybrid strategy if out-of-bid is occurring often then in order to be in race, the user bid will be updated according to the spot bid. If the user bid reaches the on-demand price then the current user is automatically shifted to on-demand service. This is very important as some time the user pays more than the on-demand price unknowingly and may not reap reliable service. The cost of executing 100 tasks on instance of type m2.2xlarge (EC 42) is shown in Fig. 11. It is observed from the graph that the cost involved for no-checkpointing strategy is more and increases with the spot price. The cost for base and hourly strategy is also more when compared to HSI, due to the reason that the task has to wait and enter into fresh bid

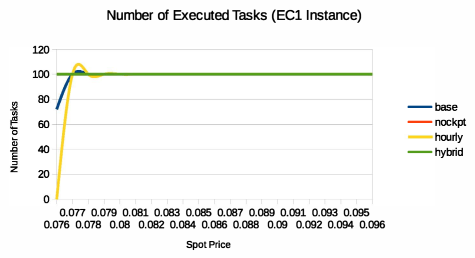


Fig. 6. Number of Tasks Completed for EC 1

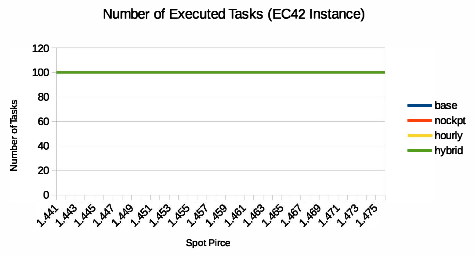


Fig. 7. Number of Tasks Completed for EC 42

process.

V. Conclusion

Spot Instance has emerged from Amazon as a resource provisioning model for delivering unused resources through highest bid concept with no guaranty of QoS. SI users require reliability and fault tolerance based on their bid price to facilitate usage of idle resources. This paper address how to improve reliability and throughput based on streach bid or rebid during resource allocation.

This paper has proposed the Hybrid Spot Instance based resource provisioning strategy for efficient utilization of idle resources. Simulation results show that providers revenue can be improved by overcoming the last partial hour overhead and also by minimizing the checkpointing through HSI. It will attract wide range of applications for the usage of spot instances, as reliable and fault tolerant services can be achieved.

We have considered Amazon price history for working and evaluating the performance of HSI. QoS at an economic price with efficient resource utilization is the foundation of

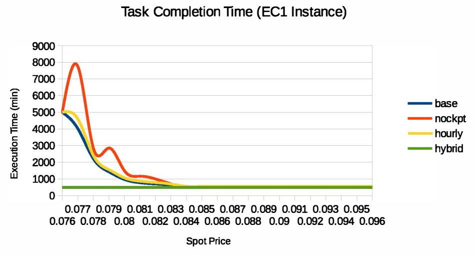


Fig. 8. Task Execution Time for EC 1

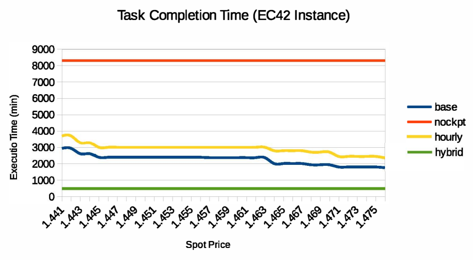


Fig. 9. Task Execution Time for EC 42

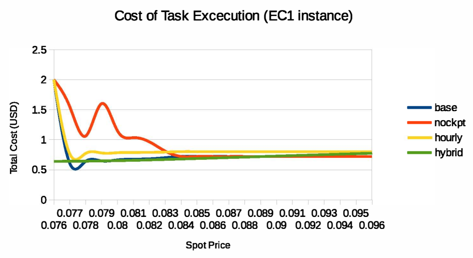


Fig. 10. Task Execution Cost for EC 1

our work. Therefore, it will be intresting to see the effects of HSIs approach on the different hosted services with real time dynamic workload in the Cloud Environment. We will investigate these problems in the future work.

VI. Acknowledgments

This research work is partly supported by the SERB - DST New Delhi, under grant reference number: NO SB/EMEQ-250/2013.

References

[1] Amazon. http://aws.amazon.com/ec2/spot-instances. In *EC2 Spot Instance*, May 2014.

[2] A. Andrzejak, D. Kondo, and Sangho Yi. Decision model for cloud computing under sla constraints. In *IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, pages 257–266, Aug 2010.

[3] K. Chard and K. Bubendorfer. High performance resource allocation strategies for computational economies. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):72–84, Jan 2013.

[4] A. Danak and S. Mannor. Resource allocation with supply adjustment in distributed computing systems. In *IEEE 30th International Conference on Distributed Computing Systems*, pages 498–506, June 2010.

[5] W. Dawoud, I. Takouna, and C. Meinel. Increasing spot instances reliability using dynamic scalability. In *IEEE 5th International Conference on Cloud Computing*, pages 959–961, June 2012.

[6] Haikun Liu, Hai Jin, Xiaofei Liao, Chen Yu, and Cheng-Zhong Xu. Live virtual machine migration via asynchronous replication and state synchronization. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):1986–1999, Dec 2011.

[7] V. Marbukh and K. Mills. Demand pricing resource allocation in market-based compute grids a model and initial results. In *Seventh International Conference on Networking*, pages 752–757, April 2008.

[8] M. Mattess, C. Vecchiola, and R. Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. In *12th IEEE International Conference on High Performance Computing and Communications*, pages 180–188, Sept 2010.

[9] M. Spreitzer M. Steinder A. Tantawi N. Chohan, C. Castillo and C. Krintz. See spot run using spot instances for mapreduce workflows. In *2nd USENIX conference on Hot topics in cloud computing*, pages 1–7, June 2010.

[10] R. Boutaba Q. Zhang, E. G. urses and J. Xiao. Dynamic resource allocation for spot markets in clouds. In *11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, pages 1–7, June 2011.

[11] N. Samaan. A novel economic sharing model in a federation of selfish cloud providers. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):12–21, Jan 2014.

[12] Yang Song, M. Zafer, and Kang-Won Lee. Optimal bidding in spot instance market. In *Proceedings IEEE INFOCOM*, pages 190–198, March 2012.

[13] Shaojie Tang, Jing Yuan, and Xiang-Yang Li. Towards optimal bidding strategy for amazon ec2 cloud spot instance. In *IEEE 5th International Conference on Cloud Computing*, pages 91–98, June 2012.

[14] W. Voorsluys and R. Buyya. Reliable provisioning of spot instances for compute-intensive applications. In *IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 542–549, March 2012.

[15] S. Garg W. Voorsluys and R. Buyya. Provisioning spot market cloud resources to create cost-effective virtual clusters. In *11th international conference on Algorithms and architectures for parallel processing*, pages 395–408, Feb 2011.

[16] Yi-Min Wang, Yennun Huang, Kiem-Phong Vo, Pe-Yu Chung, and C. Kintala. Checkpointing and its applications. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing*, pages 22–31, June 1995.

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

[18] M. Zafer, Yang Song, and Kang-Won Lee. Optimal bids for spot vms in a cloud for deadline constrained jobs. In *IEEE 5th International Conference on Cloud Computing*, pages 75–82, June 2012.

[19] Jianfeng Zhan, Lei Wang, Xiaona Li, Weisong Shi, Chuliang Weng, Wenyao Zhang, and Xiutao Zang. Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers. *IEEE Transactions on Computers*, 62(11):2155–2168, Nov 2013.

[20] Han Zhao, Miao Pan, Xinxin Liu, Xiaolin Li, and Yuguang Fang. Optimal resource rental planning for elastic applications in cloud market. In *IEEE 26th International Parallel Distributed Processing Symposium*, pages 808–819, May 2012.

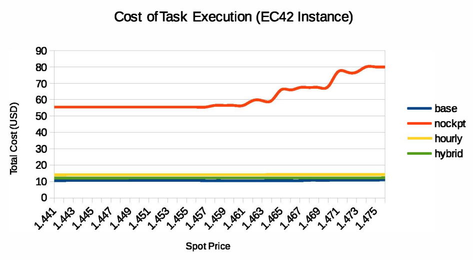


Fig. 11. Task Execution Cost for EC 42