A Prediction Approach to Define Checkpoint Intervals in Spot Instances

Jose Pergentino A. Neto[1(✉)], Donald M. Pianto[2], and Célia Ghedini Ralha[1]

[1] Department of Computer Science, University of Brasília, Brasília, DF, Brazil
paraujo@aluno.unb.br, ghedini@unb.br
[2] Statistics Department, University of Brasília, Brasília, DF, Brazil
dpianto@unb.br

Abstract. Cloud computing providers have started offering their idle resources in the form of virtual machines (VMs) without availability guarantees. Know as transient servers, these VMs can be revoked at any time without user intervention. Spot instances are transient servers offered by Amazon at lower prices than regular dedicated servers. A market model was used to create a bidding scenario for cloud users of servers without service reliability guarantees, where prices changed dynamically over time based on supply and demand. To prevent data loss, the use of fault tolerance techniques allows the exploration of transient resources. This paper proposes a strategy that addresses the problem of executing a distributed application, like bag-of-tasks, using spot instances. We implemented a heuristic model that uses checkpoint and restore techniques, supported by a statistical model that predicts time to revocation by analyzing price changes and defining the best checkpoint interval. Our experiments demonstrate that by using a bid strategy and the observed price variation history, our model is able to predict revocation time with high levels of accuracy. We evaluate our strategy through extensive simulations, which use the price change history, simulating bid strategies and comparing our model with real time to revocation events. Using instances with considerable price changes, our results achieve an 94% success with standard deviation of 1.36. Thus, the proposed model presents promising results under realistic working conditions.

Keywords: Cloud computing · Virtual Machine · Spot instance
Fault tolerance · Checkpoint · Machine learning · Statistical model

1 Introduction

Cloud Computing provides an environment with high scalability and flexibility and affordable pricing for users. It has emerged as an important community resource and is considered a new paradigm for the execution of applications with high levels of security and reliability [2,8]. According to [2], Cloud Computing is an environment where services and resources are available in a distributed way and on demand over the Internet. Furthermore, services and resources are

M. Luo and L.-J. Zhang (Eds.): CLOUD 2018, LNCS 10967, pp. 84–93, 2018.
https://doi.org/10.1007/978-3-319-94295-7_6

available in a virtual, abstract and managed platform with resources dynamically scalable in a transparent way to the user. A particular class of service is Infrastructure as a Service (IaaS), which offers a set of Virtual Machines (VMs) with different types and capacity, allowing users to choose instances according their needs.

Recently, cloud providers have been adopting a business model that offers the idle capacity of VMs resources in the form of transient servers [18]. These VMs are offered without a guarantee of their availability at considerably lower prices compared to normal servers. Amazon offers transient servers, namely Spot Instances (SIs), using a dynamic pricing model which uses a market model based on users' bids [14]. A user acquires a VM when the maximum value they are willing to pay, their bid, exceeds the current price of the instance. A bid fault (BF) occurs when the current price of the VM is above the user's bid price. Using these transient instances may weaken running applications because the environment is volatile, even if the user does not want to lose the instance. In [15], the authors state that, to effectively use transient cloud servers, it is necessary to choose appropriate fault-tolerance mechanisms and parameters.

In this paper, we present an heuristic model for efficient usage of transient instances, providing a novel strategy that uses machine learning to predict Time To Revocation (TTR) using historical data patterns and bid strategies. This approach allows the definition of fault tolerant mechanisms with appropriate parameters, reducing costs even more. These parameters are computed based on the features explained later in the paper (Sect. 4), addressing the impact of the checkpointing interval on the SIs to reduce monetary costs and the application's total execution time. The remainder of this paper is structured as follows. First, in Sect. 2, we briefly discuss related work in this domain. In Sect. 3 we describe the proposed heuristic for the definition of the parameters in a checkpoint and restore FT technique, based on a machine learning model that analyzes historical price changes of SIs. In Sect. 4 we present the results of exhaustive experiments that comply with our proposal. Finally, in Sect. 5 we summarize this study with conclusions and discuss elements for future research.

2 Related Work

Researchers have been doing significant work in the cloud computing area. The use of computing services as a utility is defined as "on demand delivery of infrastructure, applications, and business processes in a security-rich, shared, scalable, and computer based environment over the Internet for a fee" [13]. This model brings benefits to providers and consumers, especially because users can choose services and resources according to their application needs. It reduces both the monetary costs and idle resources, making it possible to provide them to other consumers. In recent years, a lot of research has been done regarding the use of FT techniques in cloud environments. Migration is considered as a FT mechanism in [6,16,17,19], and in [5] a framework is proposed to migrate servers

between VMs in an efficient way. Using a set of proposed virtualization techniques, [17] offers a platform that uses migration mechanisms to offer the cheapest environment, using unsecured spot instances instead of on-demand servers.

In a transient server perspective, much of the research focuses on exploring efficient ways to use SIs, either using techniques to define best user bids to avoid revocation [6,9,15,19] or providing FT mechanisms to guarantee application execution without data loss [17,21,22]. In the area of FT in SIs, recent works have focused on providing users a transparent environment with a high level of reliability, ensuring the execution of applications without data loss. Use of FT mechanisms are even more significant in transient instances than conventional servers where high availability is present. Even with additional overhead on running processes due to the necessity of save states, as shown in Fig. 1, checkpoint and restore is one of the most used FT mechanisms [3] and is explored in [4,10,22].

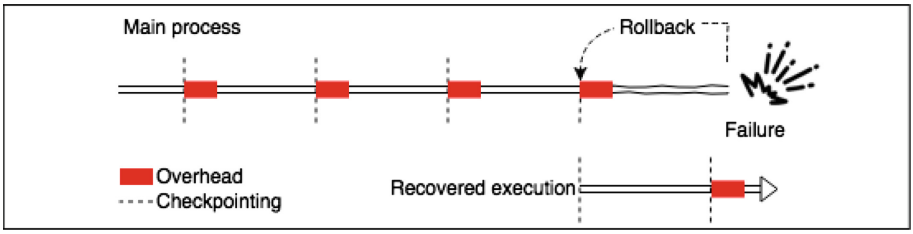

Fig. 1. Checkpoint intervals and restoration on failure of a single process execution.

From a checkpoint FT technique perspective, an important element is the checkpointing interval, because it affects total execution time. A process that executes in 30 h with 10 min of overhead checkpointing and an interval of 1 h will add 290 min of total execution time. A well defined interval time leads to lower costs and better use of execution time. Larger intervals imply faster execution.

In [19,20], the authors use a static interval of 60 min, arguing that the billing window used by the service is one hour, ignoring the fact that memory and data intensive applications need time for the save process. Alternatively, other authors present a strategy in which interval time is defined through monitoring price changes [21]. When a new price is registered, a new checkpoint is created. Using a set of defined times, [22] uses continuous checkpoints in static intervals of 10, 30 and 60 min. What is common among these related works is that the defined intervals are not dynamic. An important point is understanding the impact of the checkpointing interval on the SIs in order to reduce the monetary costs and the application's total execution time. This is the focus of this work.

3 Proposal

In this section, we present a high-level overview of our heuristic with a machine learning model that analyzes historical and current prices of SIs and their changes

to define appropriate checkpoint intervals. Compared to existing similar studies, our scheme uses observed price change patterns of SIs in terms of hour-in-day (HID) and day-of-week (DOW), as proposed in [7,8]. The performance and availability of VMs in a cloud environment varies according to instance types, region, zone and different times of day. A pattern can be observed in Fig. 2, that shows, in a range from April to December of 2017, how many price changes occurred on each day of the week (a) and during each hour of the day (b) in an *m1.large* SI, grouped by zones in the *US-WEST* region. The number of changes peaks during weekdays, as opposed to weekends, and after 5pm (17 h).

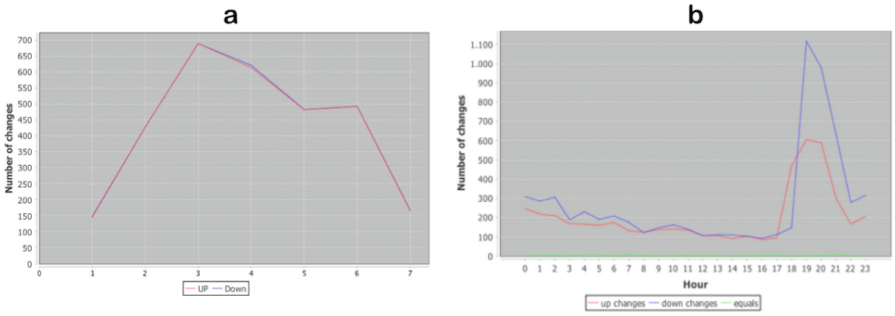

Fig. 2. Observed patterns in price changes in DOW and HID.

Compared to existing similar studies, our scheme uses a Case Based Reasoning (CBR) [1], that classifies price changes in order to comply with HID and DOW patterns. An intelligent system must be able to adapt to new situations, allowing for reasoning and understanding relationships between facts to recognize real situations and learn based on its experience. Applications that use CBR can learn through previous cases, adapting the knowledge database when evaluating new experiences.

Using a set of spot price history traces P_t, it is possible to perform simulations and create a set of real cases Δ. Algorithms that recover spot prices and simulate an auction are needed to estimate the time which a SI stays with a user until its revocation. Using a $\text{Est}_{bid}(P_t, n)$ function, that calculates the median price over the previous n days, simulations can produce scenarios where the user's bid U_{bid} assumes the returned value from Est_{bid}. With these assumptions, a set of cases can be defined as $\Delta = \{\delta_1, \delta_2 \ldots, \delta_n\}$, in which δ is a structure with the following attributes: $\delta.instance\text{-}type$, representing a type of VM instance; $\delta.day\text{-}of\text{-}week$, an integer number that determine a day of week, being 1 for Sunday and 7 for Saturday; $\delta.hour\text{-}in\text{-}day$ with a range of 0–23 representing a full hour; $\delta.price$ with instance price value; and $\delta.time\text{-}to\text{-}revocation$, that retains how much time (in minutes) a VM was alive until TTR.

In addition, our model uses a Survival Analysis (SA) statistical model [12]. SA is a class of statistical methods which are used when the data under analysis

represents the time to a specific event. In this paper, we use a nonparametric technique which incorporates censored data to avoid estimation bias. The event under study is a BF and the time to this event, TTR, is treated as a random variable.

We would like to estimate the largest survival time (T_S) for which we have a high confidence (98%) that our SI will not be revoked. This time is defined by $T_S = \arg\max_t \{t \in \mathbb{R} \mid P(TTR > t) \geq 0.98\}$ and can be calculated as the 98th percentile of TTR (TTR_{98}) from estimated Survival Curves (SC). The SCs, $\hat{S}$, are estimated using the Kaplan-Meier estimator [11], in which t_i represents the upper limit of a small time interval, D_i the number of deaths within that interval, and S_i the number of survivors at the beginning of the interval. If no deaths occur in a given interval, the SC does not decrease.

$$\hat{S}(t) = \prod_{i:t_i \leq t} (1 - \frac{D_i}{S_i}) \tag{1}$$

Figure 3 shows SCs for some SIs on Saturday at midnight. TTR_{98} for the *c3.large* instance is a little less than 100 h, while TTR_{98} for the *m3.large* is much smaller.

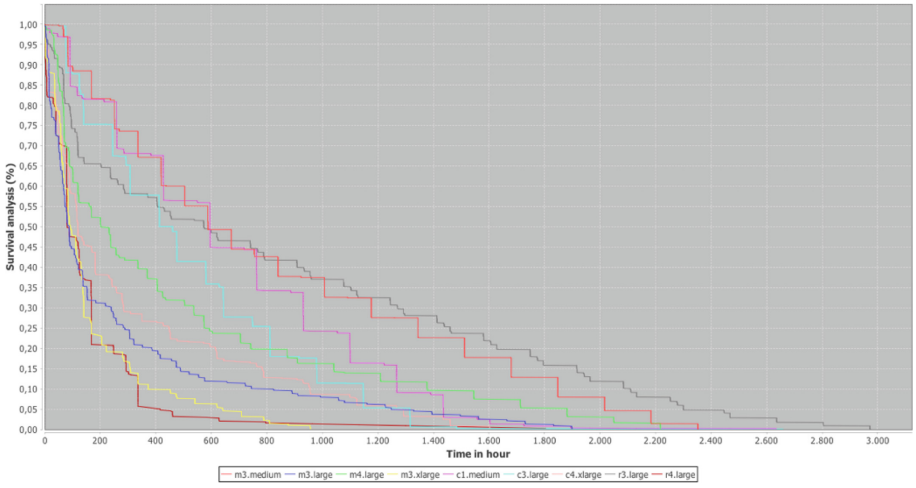

Fig. 3. Generated SC of selected SIs.

The main goal of our scheme is to find an optimal checkpointing interval that minimizes the total runtime and reduces costs. Using both CBR and SA, a SC is produced, allowing quantiles of the TTR to be used to find a probable T_S and use it as part of strategy, avoiding occurrences of BF.

4 Experimental Evaluation

We evaluate our proposed checkpointing scheme for 37 SIs in all zones of the *US-WEST* region. Using 6 months of real price changes, collected from April to December of 2017, our experiment simulates 389.576 scenarios with $U_{bid} = \text{Est}_{bid}(P_t, n) \mid n \in \mathbb{N} = \{1 \dots 7\}$, using all combinations of days and hours for the 13 weeks in October, November and December.

From the collected data, approximately 1 million cases were generated. Figure 4 shows a SC generated by the experiments. The value of TTR_{98} is indicated. We then repeated the same experiment, varying n in the Est_{bid} function between 1 and 7 days. Our results show that $n = 7$ is the best strategy, with all others performing worse.

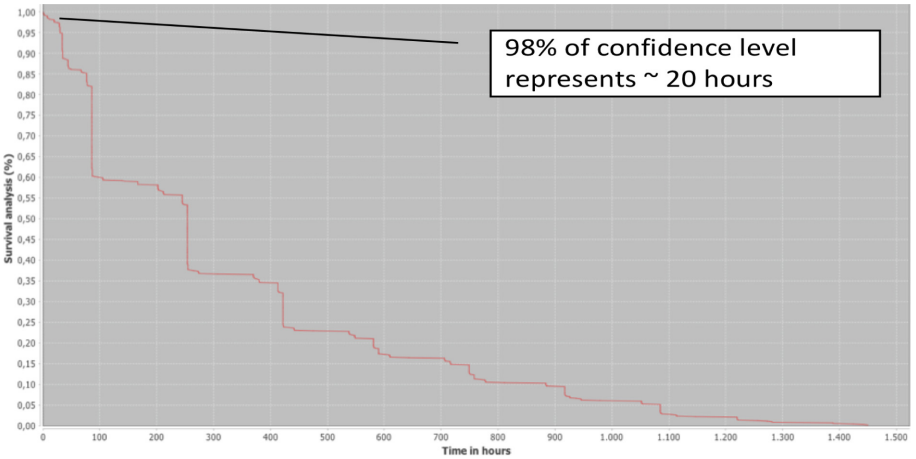

Fig. 4. SC of *c1.medium* on Sunday at 5am.

Each DOW and HID relationship has its own SC and a set of SC quantiles can be represented as a Survival Time Matrix (STM), as illustrated in Table 1, which shows TTR_{98} times, in minutes, obtained by 98th confidence level.

Table 1. Generated STM of *c1.medium* SI.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	95	129	76	220	142	100	86	130	75	706	636	569	201	168	119	330	270	208	150	90	65	151	117	94
2	83	134	78	221	159	102	84	107	68	134	123	173	129	93	68	118	95	167	109	75	69	210	121	101
3	103	129	73	226	143	100	63	154	80	445	360	309	132	189	223	156	250	192	133	74	66	345	142	101
4	108	136	80	138	111	90	199	172	79	124	360	300	215	149	102	61	261	199	141	81	68	361	194	154
5	173	138	87	119	73	102	69	128	80	717	634	574	461	438	372	318	249	191	138	85	63	227	160	128
6	85	70	69	198	134	81	67	128	78	591	253	471	169	109	123	231	189	130	89	93	64	84	80	70
7	73	120	60	207	114	99	77	134	74	680	575	346	120	111	220	238	210	164	116	89	67	130	77	120

To evaluate STM times, a set of experiments that compare STM values with real scenarios was created. Using the same experimental scenario, 80.808 simulations were executed. An example of the *c1.medium* instance is illustrated in Fig. 5. Very good results can be seen, with mean success rates around 80% with standard deviation of 1.36% points.

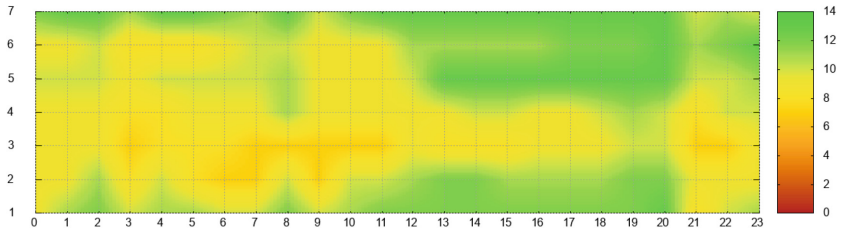

Fig. 5. Heat map of survival success compared of TTR times in *c1.medium* instance.

An unsecured gap can be observed on Monday and Tuesday between 1am and 10am, with 8 failures in 13 attempts, showing that another bid strategy is needed to increase the observed TTR and achieve better success rates. Considering all 37 instances' simulations, the success rate goes up to around 94%. This occurs because instances with expensive price values have stable variations, allowing long TTR times.

To achieve better results, a new strategy can be incorporated into the $\hat{S}$ function, where more recent results receive a greater weight. In this strategy, the time interval used in our experiment was reduced to 5 months, from April to August of 2017. With this change, a new STM was generated, creating a new matrix that represents the survival times over this period ($\mathrm{MT}_{S'}$).

Then, a new matrix that represents only September of 2017 was generated ($\mathrm{MT}_{S''}$) and a new STM was calculated as the median of $\mathrm{MT}_{S'}$ and $\mathrm{MT}_{S''}$. The results of this calculation are presented in Table 2.

Table 2. A new STM created after recency strategy in $\hat{S}$ function.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	263	203	143	83	1251	1191	1176	1090	1042	970	891	831	460	723	670	610	576	694	636	576	514	456	395	315
2	280	569	349	256	404	363	313	255	195	135	92	276	215	155	96	71	585	524	465	402	344	284	233	173
3	114	338	348	309	253	193	157	159	119	80	888	820	664	700	649	794	756	674	614	588	516	480	420	353
4	343	304	224	184	215	175	117	167	158	100	192	390	333	295	238	245	187	130	74	62	394	334	297	223
5	163	184	247	195	500	472	425	434	374	314	182	117	188	133	126	742	687	627	572	512	447	387	599	539
6	461	293	223	323	238	206	172	252	410	350	317	415	355	276	235	192	135	284	424	364	300	252	188	128
7	109	619	558	498	446	391	333	289	229	159	113	800	494	685	852	795	735	680	612	555	500	438	378	318

Better results were achieved with this change and can be observed in Fig. 6. Giving greater weight to more recent results allowed a gain of 8%, reaching an 88% success rate under the same conditions of our experiment. The new results for the DOW and HID relationship can be observed in Fig. 6.

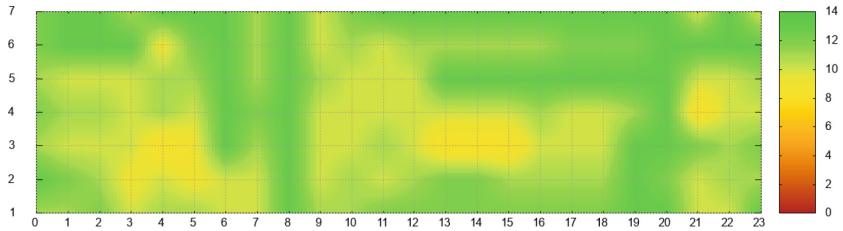

Fig. 6. Heat map of survival success after recency factor in $\hat{S}$.

Expanding our experiment to involve all 37 instances available in the US-WEST region, the success rate increases considerably, reaching 94% in our simulations, considering STM times with greater weight for more recent results. This occurs because when treating all instances, we include instances with advanced resources and with higher and more stable prices. The demand for these instances by cloud users is low and intermittent, allowing long TTR times.

Given the accuracy achieved in our experiments, we have shown that a survival curve can offer data to be used in Fault Tolerance (FT) parameters, e.g. the checkpointing interval. Given an application's required execution time (T_A) and T_S, estimated by STM, an interval can be given by Eq. 2.

$$interval = \begin{cases} \min(60, T_S) & T_A \geq T_S \\ 60 * \frac{T_S}{T_A} & T_A < T_S \end{cases} \tag{2}$$

Considering a data intensive application task that needs time $T_A = 500$ min, having checkpointing time $C_t = 10$ min, using intervals used in [19,22] (30 and 60), total execution varies between 580 and 620 min. A bigger interval means faster execution. With $T_S = 1000$ min, Eq. 2 results an interval of 120 min, leading to total execution time of 540 min. All collected data (30.3 GB) and source code are available at a public repository and can be used to reproduce our experiment and test other scenarios not contemplated in this paper.

5 Conclusions

In this paper, we present a heuristic model that uses price change traces of SIs as input in a machine learning and statistical model to predict TTR. To the best of our knowledge, there is not any available study that combines machine learning (CBR model) and SA to predict TTR in SIs. We demonstrate how our strategy can be used to define FT parameters, like the checkpointing interval or number of replications. Aspects related to the restoration of failed executions are beyond the scope of this paper. We consider that existing fault tolerance techniques can be used and their respective parameters can be defined using our approach. Furthermore, in order to achieve longer survival times, new bid strategies can be introduced, e.g. using bid values defined by a percent increase over actual instance prices.

The performed simulations have confirmed the efficiency and the accuracy of the proposed model when used in a real environment with real data. Better results are achieved when strategies which place more weight on recent data are explored. Future work can explore other strategies to exploit more recent data, such as comparing recent instance price changes with previously established change patterns to update FT parameters, like checkpoint intervals. Use of the proposed strategy to definite the checkpoint interval decreases the total time of execution, allowing more effective use of SIs and reducing monetary costs for cloud users.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–59 (1994)
2. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
3. Elnozahy, E.N.(M.), Alvisi, L., Wang, Y.-M., Johnson, D.B.: A survey of rollback-recovery protocols in message-passing systems. ACM Comput. Surv. **34**(3), 375–408 (2002). https://doi.org/10.1145/568522.568525. ISSN 0360-0300
4. He, X., Shenoy, P., Sitaraman, R., Irwin, D.: Cutting the cost of hosting online services using cloud spot markets. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing - HPDC 2015, pp. 207–218. ACN Press, New York (2015)
5. Huang, K., Gao, X., Zhang, F., Xiao, J.: COMS: customer oriented migration service. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp. 692–695. IEEE, June 2017
6. Jangjaimon, I., Tzeng, N.-F.: Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing. IEEE Trans. Comput. **64**(2), 396–409 (2015)
7. Javadi, B., Thulasiram, R.K., Buyya, R.: Characterizing spot price dynamics in public cloud environments. Future Gener. Comput. Syst. **29**(4), 988–999 (2013)
8. Javadi, D., Thulasiramy, R.K., Buyya, R.: Statistical modeling of spot instance prices in public cloud environments. In: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, pp. 219–228. IEEE, December 2011
9. Khandelwal, V., Chaturvedi, A., Gupta, C.P.: Amazon EC2 spot price prediction using regression random forests. IEEE Trans. Cloud Comput. **7161**(c), 1 (2017)
10. Lee, K., Son, M.: DeepSpotCloud: leveraging cross-region GPU spot instances for deep learning. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp. 98–105. IEEE, June 2017
11. Meeker, W.: Statistical Methods for Reliability Data. Wiley, New York (1998)
12. Miller Jr., R.G.: Survival Analysis, vol. 66. Wiley, Hoboken (2011)
13. Rappa, M.A.: The utility business model and the future of computing services. IBM Syst. J. **43**(1), 32–42 (2004)
14. Amazon Web Services: Amazon EC2 spot instances. https://aws.amazon.com/ec2/spot. Accessed Jan 2018
15. Sharma, P., Irwin, D., Shenoy, P.: Portfolio-driven resource management for transient cloud servers. Proc. ACM Meas. Anal. Comput. Syst. **1**(1), 5:1–5:23 (2017)

16. Sharma, P., Lee, S., Guo, T., Irwin, D., Shenoy, P.: SpotCheck: designing a derivative IaaS cloud on the spot market. In: Proceedings of the Tenth European Conference on Computer Systems - EuroSys 2015, pp. 1–15 (2015)
17. Sharma, P., Lee, S., Guo, T., Irwin, D., Shenoy, P.: Managing risk in a derivative IaaS cloud. IEEE Trans. Parallel Distrib. Syst. **9219**(c), 1 (2017)
18. Shastri, S., Rizk, A., Irwin, D.: Transient guarantees: maximizing the value of idle cloud capacity. In: SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 992–1002. IEEE, November 2016
19. Voorsluys, W., Buyya, R.: Reliable provisioning of spot instances for compute-intensive applications. In: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, pp. 542–549. IEEE, March 2012
20. Voorsluys, W., Garg, S.K., Buyya, R.: Provisioning spot market cloud resources to create cost-effective virtual clusters. In: Xiang, Y., Cuzzocrea, A., Hobbs, M., Zhou, W. (eds.) ICA3PP 2011 Part I. LNCS, vol. 7016, pp. 395–408. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24650-0_34
21. Yi, S., Andrzejak, A., Kondo, D.: Monetary cost-aware checkpointing and migration on amazon cloud spot instances. IEEE Trans. Serv. Comput. **5**(4), 512–524 (2012)
22. Zhou, J., Zhang, Y., Wong, W.-F.: Fault tolerant stencil computation on cloud-based GPU spot instances. IEEE Trans. Cloud Comput. **7161**(c), 1 (2017)