Uma Abordagem Baseada em Aprendizagem de Máquina para Predição de Falhas de Revogação em Instâncias Transientes

Jose Pergentino A. Neto[1], **Donald M. Pianto**[2], **Célia Ghedini Ralha**[3]

[1] Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Instituto de Educação Superior de Brasília (IESB) – Brasília, DF – Brasil

[2]Departamento de Estatística – Universidade de Brasília (UnB) – Brasília, DF – Brasil

[3]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Campus Darcy Ribeiro – Brasília, DF – Brasil

paraujo@aluno.unb.br, {ghedini, dpianto}@unb.br

Abstract. *Cloud computing providers has started offering their idle resources as transient servers. Spot instances are transient servers offered by Amazon, having their prices dynamically changed over time based on supply and demand. Using an appropriate strategic and fault tolerance mechanisms, users can effectively use spot instances to run applications in a lower price. This paper presents a strategy that combines both machine learning and a statistical model to predict time to instance revocation. Our experiments demonstrate that our heuristic is able to predict with high levels of accuracy. We evaluate our strategy and our results achieve 94% of success, which show that our model are effective and reaches high levels of accuracy under realistic working conditions.*

Resumo. *Computação em nuvem oferece seus recursos ociosos através de servidores transientes. Estes servidores são oferecidos pela Amazon como instâncias spots, que varia o seu preço de acordo com a oferta e procura. Utilizando mecanismos de tolerância a falhas e estratégias apropriadas, usuários podem executar aplicações nestas instâncias e diminuir o seu custo. Este artigo apresenta a combinação de aprendizagem de máquina e um modelo estatístico na predição do tempo até a sua revogação. Exaustivos experimentos demonstraram que a estratégia proposta atinge altos níveis de acurácia em sua predição. Com resultados que alcançam 94% de sucesso, nossa estratégia mostra a sua eficiência quando submetida a condições reais de uso dessas instâncias spots.*

1. Introdução

Computação em nuvem surgiu como uma proposta de ambientes com altos níveis de escalabilidade, flexibilidade a um preço accessível e atendendo a necessidades de computação emergente [Armbrust et al. 2010, Mell et al. 2011]. Para atender a estes elementos, provedores de nuvens computacionais necessitam de mecanismos que possam rapidamente planejar e provisionar recursos para usuários que necessitam de ambientes mais robustos na execução de aplicações. Nesse sentido, ambientes computacionais na nuvem vêm apresentando uma ótima alternativa para execução de aplicações que possuem carga de execução elevada em ambientes virtuais, seguros e escaláveis [Khan et al. 2012]. Segundo [Foster et al. 2008] e [Buyya et al. 2009], a nuvem computacional é um ambiente

onde os recursos estão conectados e disponíveis de uma maneira totalmente distribuída e disponibilizada sob demanda para usuários por meio da Internet, provendo serviços, meios de armazenamento e processamento em plataformas gerenciáveis, abstratas, virtualizadas e dinamicamente escaláveis.

Computação em nuvem tem sido bastante investigada pela sua crescente demanda, forçando provedores a desenvolver modelos variados de disponibilização dos seus serviços e recursos, principalmente na classe de *Infrastructure as a Service* (IaaS), que oferece, entre seus serviços, um conjunto de máquinas virtuais (VMs) com diferentes tipos e capacidades. Entre os provedores que se destacam ao oferecer uma coleção de serviços de computação em nuvem, estão a Amazon, Google e Microsoft, através das suas plataformas Amazon AWS *Elastic Compute Cloud* (EC2), *Google Compute Engine* (GCE) e Microsoft Azure, respectivamente. Em [Iosup et al. 2011], os autores mostram que o ambiente de nuvem computacional é visto como um promissor instrumento para execução de aplicações distribuídas do tipo *bag-of-tasks* (BoT). BoT é formado por um conjunto de aplicações (tarefas) independentes que podem ser executadas em um ambiente de execução paralela [Cirne et al. 2003]. Aplicações BoT vêm sendo utilizadas em uma variedade de cenários, incluindo simulações, processamento de imagens e bioinformática [Iosup et al. 2008, Oprescu and Kielmann 2010].

O crescimento na procura dos serviços de IaaS foi acompanhado também pela quantidade de VMs alocadas pelos usuários na categoria *on-demand*, que possui garantia de alta disponibilidade pelos seus recursos. Porém, devido a natureza da sua utilização, aplicações que estariam executando nestas máquinas virtuais podem não utilizar todos os recursos alocados. A utilização desses recursos pode variar de acordo com o tempo e, para o provedor, a imprevisibilidade na utilização destes recursos não permitem a garantia na relocação para outros usuários, porém permite a sua disponibilização sem garantia de disponibilidade provida na categoria *on-demand*. Em [Irwin et al. 2016], os autores apontam que a quantidade de recursos e a capacidade de ociosidade provida pelas alocações *on-demand* pode ser significante, devendo ser melhor explorada pelos provedores para que haja efetividade na utilização dos seus recursos e possa reduzir os seus custos operacionais.

Recentemente, provedores de nuvem computacional vêm adotando um modelo de negócio que oferece a capacidade ociosa dos recursos das máquinas virtuais em forma de servidores transientes [Irwin et al. 2016]. Os autores apresentam os servidores transientes como um conceito relativamente novo em nuvem computacional. Por este motivo, não há um padrão elaborado para definição de preços praticados e termos de uso. A AWS disponibiliza seus serviços de servidores transientes através de VMs chamadas de *Spot Instances* (SIs), que possui um mecanismo de mercado para medição de cobrança dos seus recursos. Neste modelo, semelhante a um leilão, o usuário define um valor máximo que está disposto a pagar pelo uso do recurso e adquire a SI se o valor de leilão for inferior ou igual ao seu lance [Gong et al. 2015]. A Figura 1 apresenta um exemplo do histórico na variação do preço de uma SI otimizada para memória (*m3.medium*) entre Dezembro de 2017 e Março de 2018.

Pela natureza de servidores virtuais transientes, o acesso aos seus recursos pode ser revogado do usuário a qualquer momento, sem permitir uma intervenção por parte do usuário. Uma Falha de Revogação (FR) ocorre quando o valor atual de uma SI é superior

ao lance previamente estabelecido pelo usuário. Portanto, o uso de uma SI pode comprometer a execução de aplicações devido ao seu ambiente volátil, mesmo se o usuário estiver disposto a aumentar o seu lance. Um novo lance deve ser realizado para aquisição de uma nova SI. Em [Irwin et al. 2017], os autores apresentam que, para utilizar uma SI de maneira eficiente, é necessário escolher o mecanismo apropriado para tratamento de falhas e seus parâmetros.

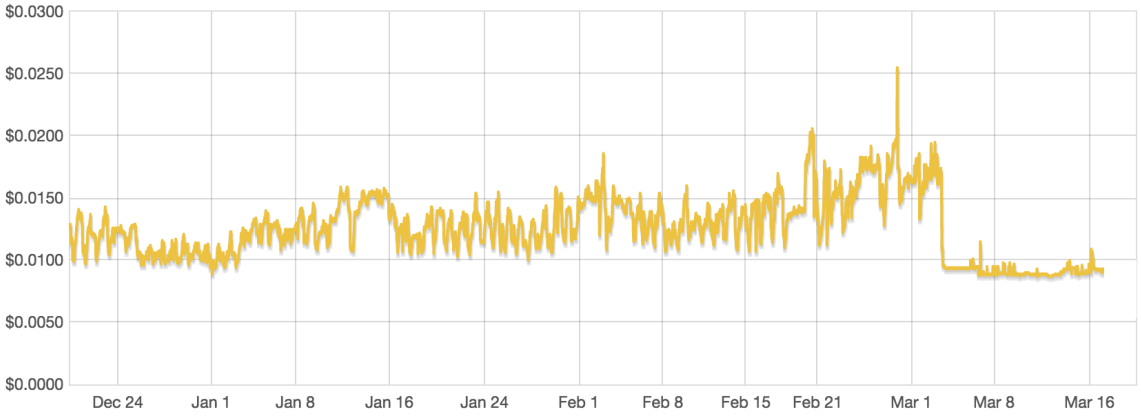

Figura 1. Histórico de preço nas instâncias spots em servidores *m3.medium*.

Neste artigo, é apresentada uma estratégia para utilização eficiente de SIs, provendo uma nova estratégia que combina aprendizagem de máquina e um modelo estatístico para predição do Tempo de Revogação (TR), baseando-se em padrões na mudança de preços combinada a estratégias de lance do usuário. Além de outras aplicabilidades, esta abordagem permite a definição do mecanismo de Tolerância a Falhas (TF) e seus parâmetros, possibilitando uma maior redução no custo final na utilização destas instâncias. A definição desses parâmetros são definidos com base nas características explicadas na Seção 3 deste artigo. O restante deste artigo está estruturado como segue. Na Seção 2 discutimos brevemente os trabalhos relacionados a este domínio. Na Seção 3 descrevemos a estratégia proposta, que oferece meios para serem utilizados na definição dos parâmetros na técnica de TF *Checkpoint/Restore* baseado em um modelo de aprendizagem de máquina que analisa o histórico da variação de preços em SI. Na Seção 4 apresentamos os resultados dos exaustivos experimentos realizados, corroborando com a abordagem proposta. E por fim, na Seção 5, sumarizamos este trabalho e discutimos elementos para futuras pesquisas.

2. Trabalhos Relacionados

Pesquisadores vêm realizando trabalhos significantes na área de computação em nuvem. A utilização de serviços computacionais no auxílio de soluções é definido como entrega sob demanda de infraestrutura, serviços e aplicações em um ambiente amplamente seguro, compartilhado e escalável através da Internet sob pagamento de um taxa [Rappa 2004]. Este modelo favorece usuários e provedores de nuvem computacional, especialmente porque usuários podem escolher os recursos e serviços de acordo com os requisitos da aplicação. Isso reduz os custos financeiros e recursos ociosos, possibilitando o provimento destes recursos para outros consumidores. Nos últimos anos, pesquisas em ambientes de nuvem computacional vêm explorando o uso de mecanismos de TF das mais variadas formas.

Em uma perspectiva que envolve instâncias transientes em nuvem computacional, pesquisas se concentram na exploração eficiente de SIs, seja utilizando técnicas para definição da melhor estratégia de lance para evitar FR [Voorsluys and Buyya 2012, Jangjaimon and Tzeng 2015, Irwin et al. 2017] ou provendo um ambiente resiliente através de mecanismos de TF para garantir a execução de aplicações sem perda de dados [Yi et al. 2012, Sharma et al. 2017b, Zhou et al. 2017]. Pesquisas que envolvem TF em SI concentram esforços em prover um ambiente transparente e resiliente, buscando a garantia do processamento sem perda de dados. Os estudos que envolvem técnicas de TF em ambientes de nuvem computacional são mais significativos em instâncias transientes comparado a servidores convencionais, que possuem disponibilidade assegurada através do SLA. Mesmo adicionando *overhead* na execução de um processo, devido a necessidade de salvar os seus estados em intervalos contínuos, como apresentada na Figura 2, *checkpoint/restore* é considerada uma das técnicas de TF mais utilizadas [Elnozahy et al. 2002] e é explorada em [Voorsluys and Buyya 2012, Jangjaimon and Tzeng 2015, He et al. 2015, Zhou et al. 2017, Lee and Son 2017, Sharma et al. 2017a].

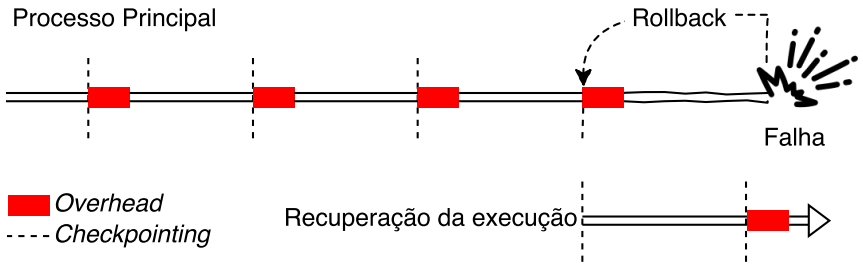

Figura 2. Funcionamento da técnica de TF *checkpoint/restore*.

Na utilização da técnica de *checkpoint/restore*, um importante elemento é a definição do intervalo de *checkpoint*, onde o estado da execução será armazenado, porque ele afeta o tempo total de execução. Um processo que executa em 30 horas, com 10 minutos de *overhead* quando cada estado é armazenado em intervalos de 1 hora, incrementa 290 minutos no tempo total do processo. Intervalos bem definidos possibilitam menores tempos nas execuções e diminuição de custos monetários. Intervalos maiores implicam em execuções mais rápidas.

Em [Voorsluys et al. 2011, Voorsluys and Buyya 2012], os autores definem um intervalo estático de 60 minutos, argumentando que a janela de cobrança aplicada pelo provedor é de 1 hora, ignorando o fato de que aplicações com grande fluxo de dados ou utilização expressiva de memória necessitam de mais tempo no processo de *checkpointing*. Como alternativa, outros autores apresentam uma estratégia na qual o tempo do intervalo é definido através do monitoramento de mudança dos preços [Yi et al. 2012]. Quando um novo preço é registrado, um novo estado é salvo. Esta estratégia compromete o tempo de execução em SIs que possuem mudanças consideráveis de preço no decorrer do tempo. Utilizando um conjunto de tempos previamente definidos, [Zhou et al. 2017] utiliza intervalos contínuos entre 10, 30 e 60 minutos, tendo opções que aumentam ainda mais o tempo total de execução comparado as outras estratégias citadas.

Os trabalhos relacionados utilizam estratégias distintas, porém com algo em co-

mum: seus intervalos não são dinâmicos e comprometem o tempo total de execução. Um importante ponto a ser observado é o entendimento do impacto deste intervalo em SIs. A definição do intervalo reflete diretamente na redução de custos monetários e no tempo total de execução da aplicação. Este é o foco deste trabalho.

3. Proposta

Nesta Seção, apresentaremos uma visão geral de nossa estratégia, que combina aprendizagem de máquina e um modelo estatístico na análise de dados históricos e atuais de SIs, definindo intervalos apropriados de *checkpoint*. Comparado a estudos similares, nossa estratégia identifica e leva em consideração os padrões nas alterações de preços considerando o dia da semana (DDS) e hora no dia (HDD), como proposto em [Javadi et al. 2013]. A performance e disponibilidade de VMs em ambientes de nuvem variam de acordo com o tipo de instância, zona, região e diferentes DDS e HDD. Observe que existem padrões na Figura 3, os quais refletem as quantidades acumuladas das alterações nos preços agrupados pelo DDS (a) e HDD (b), durante o período de Abril a Dezembro de 2017 registrados em uma SI otimizada para uso excessivo de memória (*m1.large*), agrupados pelas zonas da região *US-WEST* na AWS. A quantidade de alterações diminuem consideravelmente durante o fim de semana e possui picos após às 17 horas. Diante disso, a probabilidade de FR aumenta em momentos em que há maiores quantidades de alterações, consequentemente, poucas alterações implicam em menos FR. Além disso, a estratégia de lance dada pelo usuário é um fator que influencia fortemente a probabilidade de FR.

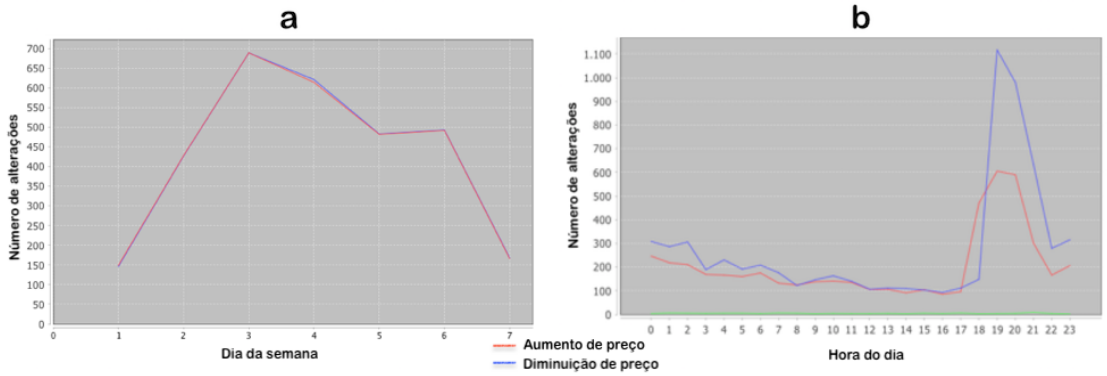

Figura 3. Padrões observados em alterações de preços em DDS e HDD.

Comparado a estudos similares, nosso esquema usa o Raciocínio Baseado em Casos (CBR- *Case Based Reasoning*) [Aamodt and Plaza 1994], que classifica alterações de preços em concordância aos padrões DDS e HDD observados. Um sistema inteligente deve ser capaz de se adaptar a situações novas e imprevisíveis, permitindo raciocínio e entendimento entre os fatos para reconhecer situações reais e aprender com suas experiências. Aplicações que utilizam CBR são capazes de aprender através de casos conhecidos previamente, adaptando a sua base de conhecimento na avaliação de novas experiências.

Utilizando os registros históricos com as alterações de preços das SIs (H_{SI}), é possível realizar simulações e criar um conjunto de situações que refletem casos reais (Δ).

Para isso, é necessário a criação de algoritmos que recuperem os preços de SIs, simulando um ambiente de leilão para identificar o tempo em que uma SI permanece com o usuário até a sua revogação. Usando uma função $Est_{bid}(H_{SI}\ , n)$, que calcula o preço mediano dos últimos n dias, um conjunto de simulações são capazes de reproduzir cenários em que o lance do usuário U_{bid} assume o valor da função Est_{bid} através de variações em seus parâmetros.

Com essas premissas, um conjunto de casos que representam casos reais pode ser definido como $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$, sendo δ formado por uma estrutura com os seguintes atributos: δ*.tipo-instância* representa o tipo de instância de uma SI; δ*.dia-da-semana* um número inteiro de 1 a 7; a δ*.hora-do-dia* um valor entre 0 e 23; δ*.preço* com o valor aplicado no momento de aquisição da SI; e δ*.tempo-ate-revogação* com o tempo (em minutos) em que uma instância permaneceu ativa com o usuário desde sua aquisição até o TR.

Além disso, nossa estratégia utiliza a Análise de Sobrevivência AS como modelo estatístico [Miller Jr 2011], que é uma classe de métodos estatísticos utilizada para trabalhar com análise de dados que representam a observação do tempo até um determinado evento. Neste artigo, nós utilizamos uma técnica não-paramétrica que incorpora, em sua análise, dados censurados para evitar estimativas viciadas. Conhecidos também como dados truncados, esses dados censurados são compostos por registros que ultrapassam o limite de observação, normalmente ignorados em outros modelos estatísticos. Uma característica decorrente destas censuras está na observação incompleta ou parcial de um evento, que pode ocorrer por uma variedade de razões, dentre elas, a perda de acompanhamento de uma observação no decorrer do estudo e a não ocorrência do evento de interesse até o término do experimento [Giolo and Colosimo 2006]. No escopo deste artigo, um evento representa uma FR e o tempo decorrido até a presença deste evento é considerado o TR.

A ideia desta estratégia está na estimativa do maior Tempo de Sobrevivência (T_S) de uma SI não ser revogada, através de um nível de confiança de 98%. Este tempo é definido a partir de $T_S = \arg\max_t\{t \in \mathbb{R} \mid P(TTR > t) \geq 0.98\}$, que pode ser calculado como o 98° percentil do TR (TR_{98}), recuperado através da Curva de Sobrevivência (CS). A função ($\hat{S}$), detalhada na equação 1, que calcula a CS, é gerada através do estimador de Kaplan-Meier [Meeker 1998], onde t_i é o limite superior de um tempo finito de intervalos, D_i representa o número de falhas dentro do intervalo e S_i retrata o número de sobrevivência desde o início do mesmo intervalo de tempo. A ausência de falhas no intervalo causa uma estabilidade e a CS não decrementa.

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

A Figura 4 apresenta um agrupamento de CSs de algumas SI inicializadas à zero hora do sábado. Como exemplo, o TR_{98} para a instância com característica de poder de processamento (*c3.large*) mostra um tempo aproximado de 80 horas, enquanto que o TR_{98} da instância com alta capacidade de memória (*m3.large*) é consideravelmente inferior.

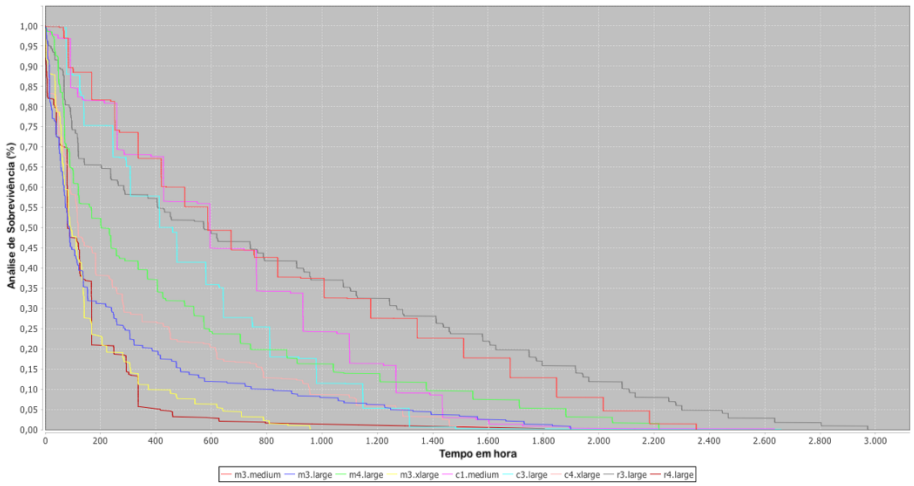

Figura 4. SIs e suas respectivas CSs.

O principal objetivo da nossa estratégia está na busca do intervalo ótimo de *checkpoint* que permita minimizar o tempo total de execução de uma aplicação e reduzir o custo total. A combinação de CBR e AS permite a criação de CSs, permitindo usar os quantis de TR na busca de um provável T_S e usá-lo como parte da estratégia, evitando a presença de FR.

4. Experimento

Na avaliação de nossa proposta, foram utilizadas 37 SIs distintas em todas as zonas da região *US-WEST*, utilizando dados reais de mudança de preços em um período de 6 meses, coletados entre Abril e Dezembro de 2017. Durante o experimento, 389.576 simulações foram realizadas, utilizando todas as combinações de dias e horas durante 13 semanas entre Outubro e Dezembro, tendo $U_{bid} = Est_{bid}(H_{SI}, n) \mid n \in \mathbb{N} = \{1 \ldots 7\}$. Através dos dados coletados, aproximadamente 1 milhão de δ foram gerados. A Figura 5 ilustra um exemplo de CS gerada através dos experimentos, indicando o valor do TR_{98}, que representa, aproximadamente, o tempo de 20 horas de sobrevivência.

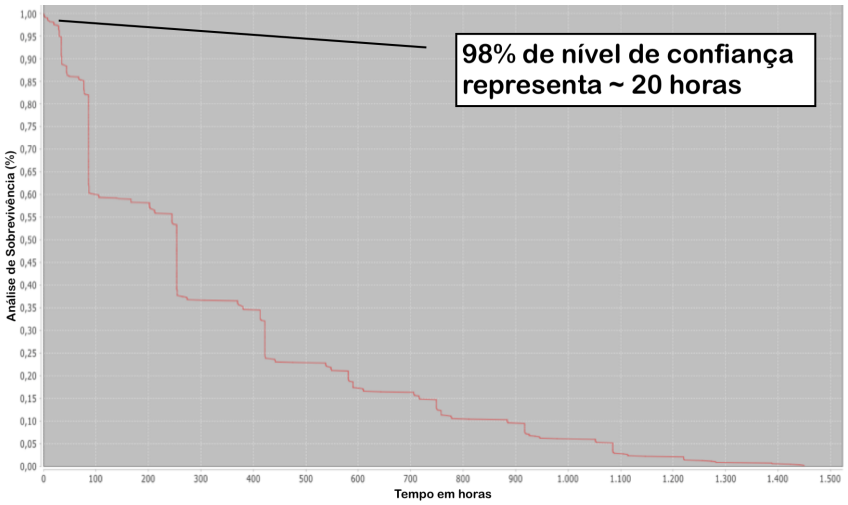

Figura 5. CS of *c1.medium* on Sunday at 5am.

O mesmo experimento foi repetido, com variações entre 1 e 7 dias no argumento n na função Est_{bid}. Nossos resultados mostram que $n = 7$ é a melhor estratégia, sendo superior aos outros elementos nesta variação. Para cada relação entre DDS e HDD, uma CS é gerada e o conjunto de T_S pode ser representada como uma Matriz de Tempo de Sobrevivência (MT_S), como ilustrada na Tabela 1, que mostra tempos, em minutos, obtidos através do 98° nível de confiança.

Tabela 1. Representação da MT_S de uma SI do tipo *c1.medium*.

	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

Para validar os tempos presentes na MT_S, um conjunto de experimentos foi realizado nas comparações entre cenários reais e os valores de tempo da MT_S. Utilizando o mesmo cenário do experimento, 80.808 simulações foram executadas e a taxa de sucesso na utilização de uma instância com poder computacional (*c1.medium*) é ilustrada na Figura 6. Podemos observar resultados positivos e significativos neste experimento, onde obteve-se taxas de sucesso de 80% com um desvio padrão de 1,36. Uma lacuna pode ser observada nas relações entre segunda e terça-feira entre 3 e 11 horas, com 8 falhas em 13 tentativas. Para mitigar este cenário, outra estratégia de lance deve ser considerada para o aumento do TR.

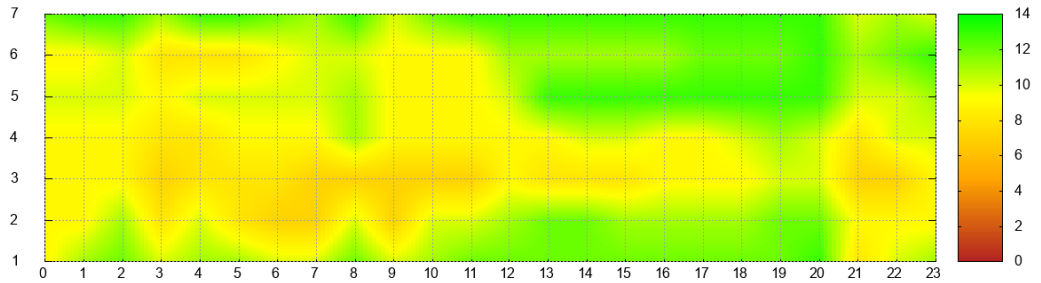

Figura 6. Mapa de calor de sobrevivências no TR em uma SI *c1.medium*.

Na busca de resultados mais expressivos, uma heurística onde elementos que caracterizam a recência foram adicionados na função $\hat{S}$. Nessa heurística, o intervalo de tempo para o experimento a ser utilizado pela função foi alterado para o período de 5 meses, coletados entre Abril e Agosto de 2017. Com esta mudança, uma nova MT_S foi gerada ($\text{MT}_{S'}$), que representa os tempos de sobrevivência desse período. Além disso, outra matriz que representa apenas o mês de Setembro de 2017 foi gerada ($\text{MT}_{S''}$) e uma nova MT_S foi calculada através do cálculo da média entre $\text{MT}_{S'}$ e $\text{MT}_{S''}$, tendo os seus resultados apresentados na Tabela 2.

Podemos observar que resultados mais expressivos foram alcançados com esta mudança. O fator recência possibilitou um aumento de 8 pontos percentuais, chegando a 88% de sucesso para o cenário deste experimento. Um novo resultado é apresentado na Figura 7.

Tabela 2. Representação da nova MT$_S$ após a recência na $\hat{S}$.

	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

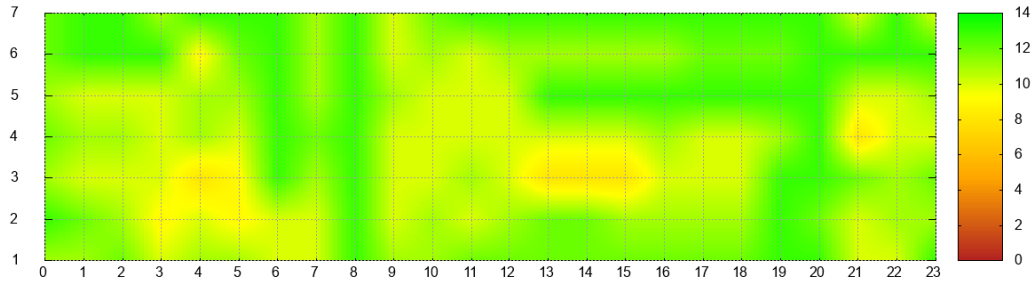

Figura 7. Mapa de calor de sobrevivências após modificação na $\hat{S}$.

Através dos altos índices alcançados em nossos experimentos, a construção de uma CS pode oferecer informações confiáveis para serem utilizadas em parâmetros de TFs, e.g., o intervalo em que os estados serão salvos na técnica de *checkpoint*. Dado um tempo necessário para a execução de uma aplicação (T_A) e o T_S recuperado a partir de uma MT$_S$, um intervalo de *checkpoint* pode ser dado através da Equação 2.

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

Considerando que uma aplicação com um fluxo elevado de dados ou consumo excessivo de memória necessita de um tempo considerável para a sua execução $T_A = 500$ minutos, com um tempo de $C_T = 10$ minutos no processo de salvar o estado da aplicação e utilizando os intervalos propostos em [Zhou et al. 2017, Voorsluys and Buyya 2012] (10, 30 e 60 minutos), o total de tempo de execução varia entre 580 e 1160 minutos. Como intervalos maiores significam execuções mais rápidas, tendo um tempo $T_S = 1000$, a Equação 2 produz como resultado um intervalo de 120 minutos. Utilizando este novo intervalo, o tempo total de execução é de 540 minutos, sendo inferior aos alcançados nas propostas citadas.

Incrementando o experimento ao envolver todas as 37 instâncias e computar os seus resultados durante as simulações, a taxa de sucesso aumenta consideravelmente, atingindo o valor aproximado de 94% de sucesso em simulações de sobrevivência. Isso ocorre devido a estabilidade na mudança de preços de algumas instâncias, que possuem valores mais elevados e não sofrem mudanças consideráveis em seus valores devido a baixos índices de procura pelos seus usuários, permitindo TR maiores. Todos os dados coletados e o código-fonte estão disponíveis em um repositório público[1].

[1]Disponível em https://github.com/jpergentino/spot-prediction

5. Conclusão

Neste artigo foi apresentada uma estratégia que utiliza o histórico das mudanças nos preços de SIs como dados de entrada em uma combinação dos modelos de aprendizagem de máquina e estatístico para fornecer valores preditivos de TR. Até onde foi possível pesquisar, não há trabalhos reportados na literatura que envolvem os temas relacionados neste artigo. Este é o primeiro estudo que combina aprendizagem de máquina através do modelo CBR e AS na predição de TR em SIs. Através dos experimentos, foi demonstrado que esta abordagem pode ser utilizada na definição de qual técnica de TF a ser adotada e os seus respectivos parâmetros, tais como o tempo do intervalo em *checkpoint* ou a quantidade de replicações para execução de uma aplicação.

Os experimentos realizados confirmam a eficiência e a acurácia da estratégia proposta, alcançando elevadas taxas de sucesso durante as simulações que refletem ambientes e dados reais. Ao permitir que a estratégia apresentada possa definir o intervalo de *checkpoint*, os custos podem diminuir devido a redução do tempo total de execução, permitindo o uso de SIs de maneira mais eficiente. Como trabalhos futuros pretende-se investigar o uso desta abordagem em cenários de aplicações reais.

Referências

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616.

Cirne, W., Brasileiro, F., SauvÃ©, J., Andrade, N., Paranhos, D., Santos-neto, E., Medeiros, R., and Gr, F. C. (2003). Grid computing for bag of tasks applications. In *In Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*.

Elnozahy, E. N. M., Alvisi, L., Wang, Y.-M., and Johnson, D. B. (2002). A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE.

Giolo, S. R. and Colosimo, E. A. (2006). Análise de sobrevivência aplicada. *Edgard Blucher*.

Gong, Y., He, B., and Zhou, A. C. (2015). Monetary cost optimizations for MPI-based HPC applications on Amazon clouds. In *Proceedings of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis on - SC '15*, pages 1–12, New York, New York, USA. ACM Press.

He, X., Shenoy, P., Sitaraman, R., and Irwin, D. (2015). Cutting the Cost of Hosting Online Services Using Cloud Spot Markets. In *Proceedings of the 24th Int. Symposium*

on High-Performance Parallel and Distributed Computing - HPDC '15, pages 207–218, New York, New York, USA. ACM Press.

Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., and Epema, D. H. J. (2011). Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945.

Iosup, A., Sonmez, O., Anoep, S., and Epema, D. (2008). The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing - HPDC '08*, page 97, New York, New York, USA. ACM Press.

Irwin, D., Sharma, P., and Shenoy, P. (2017). Portfolio-driven Resource Management for Transient Cloud Servers. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1):5:1—-5:23.

Irwin, D., Shastri, S., and Rizk, A. (2016). Transient guarantees: maximizing the value of idle cloud capacity. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 85. IEEE Press.

Jangjaimon, I. and Tzeng, N.-F. (2015). Effective Cost Reduction for Elastic Clouds under Spot Instance Pricing Through Adaptive Checkpointing. *IEEE Transactions on Computers*, 64(2):396–409.

Javadi, B., Thulasiram, R. K., and Buyya, R. (2013). Characterizing spot price dynamics in public cloud environments. *Future Generation Computer Systems*, 29(4):988–999.

Khan, A., Xifeng Yan, Shu Tao, and Anerousis, N. (2012). Workload characterization and prediction in the cloud: A multiple time series approach. In *2012 IEEE Network Operations and Management Symposium*, pages 1287–1294. IEEE.

Lee, K. and Son, M. (2017). DeepSpotCloud: Leveraging Cross-Region GPU Spot Instances for Deep Learning. In *2017 IEEE 10th Int. Conf. on Cloud Computing (CLOUD)*, pages 98–105. IEEE.

Meeker, W. (1998). *Statistical Methods for Reliability Data*. Wiley, New York.

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing. *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg*.

Miller Jr, R. G. (2011). *Survival analysis*, volume 66. John Wiley & Sons.

Oprescu, A.-M. and Kielmann, T. (2010). Bag-of-Tasks Scheduling under Budget Constraints. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 351–359. IEEE.

Rappa, M. A. (2004). The utility business model and the future of computing services. *IBM Systems Journal*, 43(1):32–42.

Sharma, P., Irwin, D., and Shenoy, P. (2017a). Portfolio-driven Resource Management for Transient Cloud Servers. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1):5:1—-5:23.

Sharma, P., Lee, S., Guo, T., Irwin, D., and Shenoy, P. (2017b). Managing Risk in a Derivative IaaS Cloud. *IEEE Transactions on Parallel and Distributed Systems*, 9219(c):1–1.

Voorsluys, W. and Buyya, R. (2012). Reliable Provisioning of Spot Instances for Compute-intensive Applications. In *2012 IEEE 26th Int. Conf. on Advanced Information Networking and Applications*, pages 542–549. IEEE.

Voorsluys, W., Garg, S. K., and Buyya, R. (2011). *Provisioning Spot Market Cloud Resources to Create Cost-Effective Virtual Clusters*, pages 395–408. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yi, S., Andrzejak, A., and Kondo, D. (2012). Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances. *IEEE Transactions on Services Computing*, 5(4):512–524.

Zhou, J., Zhang, Y., and Wong, W.-F. (2017). Fault Tolerant Stencil Computation on Cloud-based GPU Spot Instances. *IEEE Transactions on Cloud Computing*, 7161(c):1–1.