DOI: xxxxxxxxxxxxxxxxxxxxxxxxx

FIFC: Uma Ferramenta de Injeção de Falhas em Ambientes de Nuvens

Cláurdio Márcio Filho
Universidade Federal Rural de Pernambuco
claudiomarciofilho@hotmail.com

Erica Teixeira Gomes de Sousa
Universidade Federal Rural de Pernambuco
erica.sousa@ufrpe.br

RESUMO

Garantir a disponibilidade de sistemas de nuvem é um ponto crucial para assegurar a qualidade dos serviços oferecidos por meio destes sistemas. A injeção de falhas pode ser uma ótima maneira de testar ambientes de nuvem e verificar onde existem potenciais riscos à sua dependabilidade, além disso pode fornecer dados cruciais para construir mecanismos para detecção e predição de falhas. Este trabalho apresenta uma ferramenta para injeção de falhas para computação em nuvem. Através desta ferramenta foi possível analisar o impacto da ocorrência de falhas num ambiente de nuvem configurado com a plataforma OpenStack.

Palavras-Chaves

Computação em Nuvem; Injeção de Falhas; Avaliação de Disponibilidade.

ABSTRACT

Ensuring the availability of cloud systems is a crucial point in ensuring the quality of services offered through these systems. Fault injection can be a great way to test cloud environments and check where there are potential risks to their dependability, and it can also provide crucial data to build mechanisms for detecting and predicting faults. This work presents a fault injection tool for cloud computing. Using this tool, it was possible to analyze the impact of failures in a cloud environment configured with the OpenStack platform.

Keywords

Cloud Computing; Fault Injection; Availability Evaluation.

CCS Concepts

• Computer systems organization→Dependable and fault-tolerant systems and networks

1 INTRODUÇÃO

A computação em nuvem permite disponibilizar serviços de maneira prática, rápida e de custo acessível para muitos usuários, contudo, sistemas de nuvem também estão sujeitos a sofrer falhas que podem trazer grandes impactos financeiros tanto para os provedores do serviço hospedado na nuvem quanto para os próprios provedores da plataforma de nuvem [1].

Estima-se que as ocorrências de falhas na computação em nuvem representam um prejuízo anual de 700 bilhões de dólares [5]. Uma falha em um componente da nuvem computacional, pode afetar muitos outros devido a interdependência entre eles. Desta forma, uma abordagem eficaz é detectar a falha em seus estágios iniciais, visto que medidas preventivas podem ser realizadas para evitar sua ocorrência [8].

A injeção de falhas é uma atividade essencial para possibilitar a análise da dependabilidade de sistemas de nuvem. Essa análise permite prover mecanismos de tolerância a falhas de forma a mitigar a ocorrência de falhas no ambiente de nuvem, oferecendo uma maior disponibilidade desse ambiente [3].

Diante destas informações, este trabalho apresenta uma ferramenta para a injeção de falhas em ambientes de nuvem. Essa ferramenta permite a análise do impacto da ocorrência de falhas em serviços na nuvem.

A Seção 2 deste artigo traz os principais conceitos utilizados neste trabalho. Já a Seção 3 apresenta mais detalhes sobre trabalhos relacionados à injeção de falhas em ambientes de nuvem. A Seção 4 apresenta a ferramenta proposta e a Seção 5 detalha os experimentos que foram executados e os resultados obtidos. E a Seção 6 conclui este trabalho com considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão definidos alguns conceitos importantes para este trabalho.

2.1 Computação em Nuvem

Computação em nuvem, segundo o *National Institute of* Standards and Technologies (NIST), é um forma de disponibilizar recursos computacionais que podem ser acessados de maneira universal. Esses recursos computacionais são computação, armazenamento, rede e devem ser provisionados de maneira prática quando necessários [2].

O OpenStack é uma plataforma de nuvem open source que permite que um usuário gerencie um sistema de nuvem. Esta plataforma é utilizada mundialmente por grandes empresas provedoras dos mais variados serviços como a Blizzard Entertainment, Daum Communication, LINE Corporation e Adobe [7].

O OpenStack é dividido em vários módulos responsáveis por gerenciar diferentes aspectos da nuvem, como a criação de instâncias com o Nova, o gerenciamento de redes virtuais com o Neutron e gerenciamento de volumes com o Cinder e o Swift, além de outras funcionalidades como monitoramento

(Ceilometer), automação (Heat) e identificação (Keystone) [6].

2.2 Injeção de Falhas

A injeção de falhas é uma técnica importante para avaliar a confiabilidade dos sistemas de computador, seja por hardware ou software. O contraste entre os métodos de hardware e software reside principalmente nos pontos de injeção de falha que eles podem acessar, o custo e o nível de perturbação [4].
Os métodos de hardware podem injetar falhas nos pinos do chip e componentes internos, como circuitos combinacionais e registros que não são endereçáveis por software. Por outro lado, os métodos de software são convenientes para produzir mudanças diretamente no nível do estado do software [4].

3 TRABALHOS RELACIONADOS

A técnica de injeção de falhas vem sendo utilizada em muitos estudos sobre falhas em ambientes de nuvem. Essa seção apresenta a importância dessa técnica mostrando os diversos trabalhos relacionados que existem na área.

Os trabalhos como [13], [15], [16], [17] e [18] adotam a injeção de falhas para testar a confiabilidade da nuvem e a sua capacidade de recuperação após a ocorrência de falhas, ou seja, quais procedimentos a nuvem realiza para tentar se recuperar de uma falha e como isso de fato afeta a nuvem.

Alguns trabalhos apresentam ferramentas de técnica de injeção de falhas, como é o caso dos trabalhos de [9], [10], [11], [12] e [14]. Nesses trabalhos, ocorre uma descrição de tipos de falhas que se pode esperar na nuvem e também apresentam ferramentas que podem ser utilizadas para injetar ou simular falhas numa nuvem. Nesses trabalhos são apresentadas ferramentas como o ucXception e o iSPD, o primeiro sendo uma ferramenta de injeção de falhas e o segundo um simulador onde é possível simular uma nuvem e gerar erros nessa simulação.

Tabela 1. Trabalhos relacionados

Trabalho	Assunto	Tipos de Falhas	Ferramentas e Contribuições
[16]	Injeção de falhas para desenvolver mecanismos de recuperação de falhas.	Falhas transientes e “desastres”	Avaliação de disponibilidade com Redes de Petri Estocásticas, Design de um sistema de recuperação de desastres
[13]	Injeção de falhas para desenvolver mecanismos de recuperação de falhas.	-	Chaos Engineering e Digital Twin
[18]	Injeção de falhas para desenvolver mecanismos de recuperação de falhas.	Destruição de VMs	CloudSim
[15]	Injeção de falhas para desenvolver mecanismos de recuperação de falhas.	Falhas de software e falhas de envelhecimento (*aging*) do software	Estudo de caso dos mecanismos de tolerância falhas de micro serviços no Kubernetes
[17]	Injeção de falhas para desenvolver mecanismos de recuperação de falhas.	Falhas nas operações de input e output.	CrashFuzz
[10]	Proposição de nova ferramenta para injeção de falhas para ambientes de nuvem.	-	-
[12]	Tipos de falhas na nuvem, proposição de novas ferramentas para injeção de falhas para ambientes de nuvem.	Falha de rede, Falhas de hardware, Falha de mídia, Falha de máquina virtual, Falha por omissão, Falhas arbitrárias.	iSPD
[11]	Tipos de falhas na nuvem, proposição de novas ferramentas para injeção de falhas para ambientes de nuvem.	Falhas de software (exceções, delays, retornos incorretos, parâmetros inválidos)	Algoritmo de detecção de anomalia e algoritmo de classificação de falhas
[9]	Tipos de falhas na nuvem, seleção de alvos para injetar falhas.	Hardware, Consumo de Recurso, Rede, Configuração.	Syringe, Chaos Engineering,
[14]	Tipos de falhas na nuvem, proposição de novas ferramentas para injeção de falhas para ambientes de nuvem.	Hardware e software	ucXception

Na Tabela 1 é possível ver uma comparação entre os trabalhos relacionados, é possível notar que praticamente todos utilizam falhas de software como principal tipo de falha a ser injetada, observa-se também que os trabalhos se dividem em trabalhos que focam em utilizar a injeção de falhas para desenvolver novos algoritmos e em trabalhos que visam criar novas ferramentas para injeção de falhas a partir do entendimento dos tipos de falhas possíveis.

Visto a importância dessa área, o presente trabalho é a primeira etapa para a introdução de uma nova ferramenta de injeção de falhas que seja capaz de executar essa tarefa de maneira menos determinística e mais randômica, além da possibilidade de utilizar essa ferramenta em diversos tipos de ambientes de nuvem.

Para tal, um primeiro script de injeção de falhas foi desenvolvido e testado num ambiente de nuvem OpenStack, esse script e os testes executados serão discutidos nas seções posteriores.

4 A FERRAMENTA DE INJEÇÃO DE FALHAS EM AMBIENTES DE NUVEM

Com o objetivo de realizar injeção de falhas na nuvem uma nova ferramenta foi projetada. O fluxograma da Figura 1 foi utilizado para mostrar como ocorreu a implementação da ferramenta de injeção de falhas em ambientes de nuvem FIFC, um script injetor de falhas de software desenvolvido para a nuvem OpenStack.

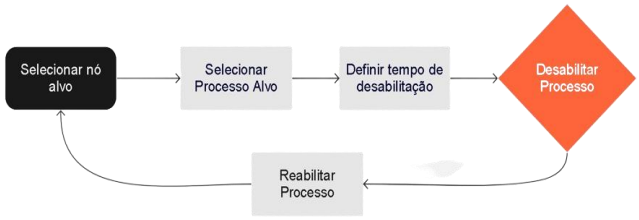

Figura 1: Fluxo do script desenvolvido para injeção de falhas.

O script desenvolvido funciona em 5 passos:

1. Selecionar um nó alvo: de maneira randômica, um dos nós da nuvem é selecionado para ser alvo da injeção de falhas
2. Escolher um processo alvo: o nó alvo possui diversos processos relacionados à nuvem e um desses processos é selecionado de maneira aleatória.
3. Definir intervalo de desabilitação: definir uma quantidade de tempo entre 3 a 10 minutos para desabilitar o processo alvo.
4. Desabilitar o processo: com os parâmetros definidos o processo alvo é encerrado.
5. Reabilitar o processo: após o tempo definido ter decorrido o processo é reativado.

5 ESTUDO DE CASO

O objetivo dessa seção é apresentar os experimentos de injeção de falhas realizados no ambiente de nuvem privada configurado com a plataforma de nuvem Openstack. O ambiente de nuvem é composto de 3 máquinas físicas onde foram configuradas com os principais módulos Openstack, constituindo uma nuvem com 3 nós, o Nó Controlador (Controller Node), o Nó de Computação (Compute Node) e o Nó de Rede (Network Node). As configurações das máquinas físicas adotadas para a configuração do nós da nuvem são apresentadas na Tabela 2.

Tabela 2. Configuração dos máquinas físicas da nuvem

Nó	Processador	Memória	Disco	Sistema Operacional	Módulos OpenStack	Monitoramento
Compute Node	Intel Core i5-	8 GB	1 TB	CentOS 7.9 Minimal	Nova, Neutron	Zabbix Agent
Controller Node	5200U				Nova, Neutron, Cinder, Swift, Glance, Keystone	Zabbix Agent Server
Network Node					Neutron	Zabbix Agent

O nó Controlador possui as funções principais de todos os grandes módulos do OpenStack é ele quem interpreta e aciona os comandos necessários para o funcionamento da nuvem, também é nele que a central de monitoramento foi instalada para coletar e concentrar os dados.

O nó de Computação fornece os recursos para instanciação de máquinas virtuais, por isso apenas os módulos Nova e Neutron estão presentes nele, para o monitoramento só é necessário o agente Zabbix para coletar e enviar os dados para o servidor Zabbix.

Por fim, o nó de Rede fica responsável por gerenciar as funcionalidades de rede como um serviço, e assim como o nó de Computação apenas o agente Zabbix está presente para coletar dados. A Figura 2 mostra a configuração do sistema Zabbix.

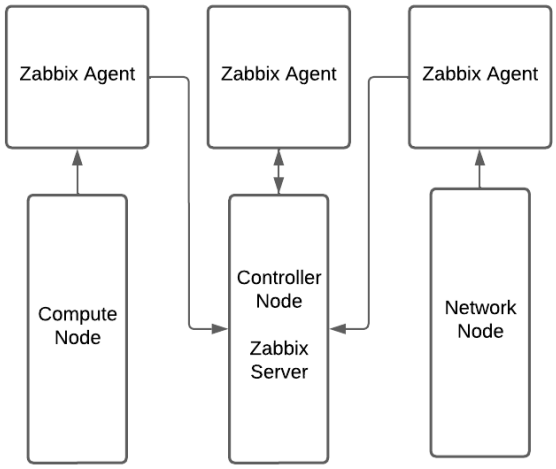

Figura 2: Composição da nuvem utilizada nos experimentos.

Os experimentos tem como objetivo injetar falhas na nuvem através da desabilitação de serviços que compõem a nuvem, isso significa que os serviços são desativados, forçando a nuvem a operar em estado de erro e podendo assim gerar falhas.

Nesses experimentos, a injeção de falhas ocorreu no Neutron, módulo responsável pelo gerenciamento da rede, onde podemos destacar o processo do *neutron-server* um importante componente que armazena e gerencia as informações da associação de endereços IP e das redes virtuais criadas para a nuvem. A injeção de falhas ocorreu também no Nova, módulo que gerencia a criação das máquinas virtuais da nuvem, destacando-se o *libvirtd* e o *httpd*, que são componentes responsáveis por gerenciar a

virtualização dos recursos e as requisições de operações da nuvem, respectivamente.

A Figura 3 mostra o cenário construído para os experimentos de injeção de falhas.

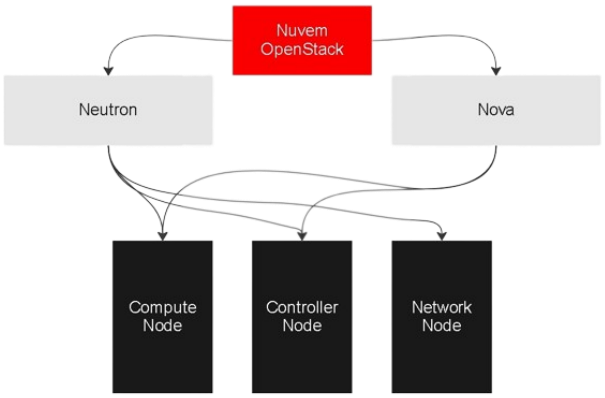

Figura 3: Composição da nuvem utilizada nos experimentos.

Cada módulo é testado separadamente utilizando o script personalizado apresentado na Seção 4 (Figura 1). Além do script, uma carga de trabalho foi usada baseada na criação/exclusão de máquinas virtuais da menor configuração possível (1 vCPU e 1 GB de RAM) e sem volume de armazenamento. Essas máquinas virtuais foram adotadas apenas para gerar trabalho de alocação de recursos de computação e rede. O recurso de armazenamento foi desconsiderado na instanciação da máquina virtual uma vez que esse recurso é de responsabilidade de outro módulo do OpenStack, o Cinder, que não foi selecionado para teste neste trabalho.

5.1 Experimento 1 com o Módulo Neutron

Neste experimento, a injeção de falhas ocorreu no módulo Neutron. O Neutron é o módulo que gerencia a rede tanto das máquinas físicas que representam os nós da nuvem quanto das máquinas virtuais. Ao desabilitar os principais processos deste módulo pode ocorrer problemas de conexão. Para este experimento a carga de trabalho era gerada manualmente. A carga de trabalho consistia da requisição de criação de 1 máquina virtual com 1 vCPU e 1 GB de RAM que após a criação era imediatamente excluída.

5.2 Experimento 2 com o Módulo Nova

O segundo experimento injetou falhas no módulo Nova. Nesse experimento, os nós de Controle e Computação foram alvos do teste, uma vez que o módulo de rede não executa tarefas de compartilhamento de recursos para a execução das instâncias virtuais. Esse experimento foi executado por aproximadamente 6 horas, pois tem-se assim uma boa quantidade de dados observados. A carga de trabalho para este segundo experimento foi automatizada. Três instâncias eram criadas e excluídas a cada 5 minutos durante todo o processo de injeção de falhas, assim o sistema da nuvem era utilizado em boa parte da sua capacidade simulando o uso normal.

O Nova é o módulo responsável pela computação, ou seja, gerenciamento de recursos e criação de instâncias. Desabilitar os serviços deste módulo causa impacto direto na administração das máquinas virtuais.

O serviço *httpd* é responsável por disponibilizar um *endpoint* para que os nós da nuvem façam suas requisições e a nuvem opere corretamente, ele foi considerado como um possível serviço de computação dada a criticidade de uma possível falha na criação de máquinas virtuais.

5.3 Resultados dos Experimentos de Injeção de Falhas

As Tabelas 3 e 4 resumem os resultados obtidos no experimento com o Neutron. A Tabela 3 apresenta os processos que ao serem desabilitados geraram falhas. A Coluna 1 mostra quais os nós referentes a cada módulo e a Coluna 2 mostra os serviços que podem ser desabilitados em determinado nó. A Coluna 3 exibe os tipos de falhas que foram observadas ao longo do experimento ao desativar cada processo. Já as Colunas 4 e 5 mostram o consumo de recursos do módulo no nó alvo, por fim, as Colunas 6 e 7 detalham o consumo médio de cada processo em seu respectivo nó.

Tabela 3: Resultados do experimento Neutron destacando processos que geraram falhas

Nó Alvo	Processo Desabilitado	Tipos de Falhas	Utilização de CPU (Nó Alvo)	Utilização de Memória (Nó Alvo)	Utilização de CPU (Processo)	Utilização de Memória (Processo)
Network	ovs-vswitchd	Falha de conexão; Máquina Inacessível.	5%	17%	0.2	36MB
	ovsdb-server	Falha de conexão; Máquina Inacessível.			<0.01%	4MB
Compute	ovs-vswitchd	Falha de conexão; Máquina Inacessível.	5%	30%	0.2%	37MB
	ovsdb-server	Falha de conexão; Máquina Inacessí			<0.01%	5MB

		vel.				
Controller	neutron-server	Falha na criação da VM, não é possível recuperar as informações da VLAN para anexar a VM.	40%	98%	20%	1.5GB
	ovs-vswitchd	Falha de conexão; Máquina Inacessível.			0.2%	38MB
	ovsdb-server	Falha de conexão; Máquina Inacessível.			<0.01%	3MB

A Tabela 4 apresenta os processos que ao serem desabilitados não geraram falhas nos experimentos com o Neutron. A Coluna 1 mostra quais os nós referentes a cada módulo e a Coluna 2 mostra os serviços que foram desabilitados. Já as Colunas 3 e 4 mostram o consumo de recursos do processo.

Tabela 4: Resultados do experimento Neutron destacando processos que não geraram falhas

Nó Alvo	Processo Desabilitado	Utilização de CPU (Processo)	Utilização de Memória (Processo)
Compute	neutron-openvswitch-agent	2.2%	130MB
Controller	neutron-l3-agent	2%	110MB
	neutron-openvswitch-agent	2.2%	120MB
	neutron-dhcp-agent	3%	110MB
	neutron-metadata-agent	10%	200MB
	neutron-metering-agent	2%	110MB
Network	neutron-l3-agent	1.9%	125MB
	neutron-openvswitch-agent	2.1%	150MB
	neutron-dhcp-agent	1.9%	120MB
	neutron-metadata-agent	1.9%	500MB
	neutron-metering-agent	1.9%	146MB

Como resultado do experimento com o Neutron foram observadas falhas de conexão uma vez que os nós da nuvem ficaram inalcançáveis e com isso a carga de trabalho não foi executada corretamente. Por isso, o tempo de execução deste experimento foi bastante reduzido, não passando de 60 minutos. Em relação aos recursos vale destacar o alto consumo de memória apresentado pelo nó controlador chegando a um consumo total aproximado de 98%, isso se deve principalmente ao fato da reinicialização dos processos causar picos no consumo de memória e de processador.

As Tabelas 5 e 6 apresentam os resultados dos experimentos com o Nova. Na Tabela 5 estão os processos que ao serem desabilitados causaram falhas. A Coluna 1 mostra quais os nós referentes a cada módulo e a Coluna 2 mostra os serviços que foram desabilitados. Já a Coluna 3 exibe o processo desabilitado e as Colunas 4 e 5 mostram o consumo de recursos das respectivas máquinas.

Tabela 5: Resultados do experimento Nova destacando processos que geraram falhas

Nó Alvo	Processo Desabilitado	Tipos de Falhas	Utilização de CPU (Nó Alvo)	Utilização de Memória (Nó Alvo)
Compute	libvirtd	Falha na comunicação de processos;	23.5%	39.6%
Controller	libvirtd	Falha na comunicação de processos;	58%	98%

Na Tabela 6 estão os processos que ao serem desabilitados não causaram falhas. A Coluna 1 mostra quais os nós referentes a cada módulo e a Coluna 2 mostra os serviços que foram desabilitados. Já as Colunas 3 e 4 mostram o consumo de recursos das respectivas máquinas.

Tabela 6: Resultados do experimento Nova destacando processos que não geraram falhas

Nó Alvo	Processo Desabilitado	Utilização de CPU (Processo)	Utilização de Memória (Processo)
Compute	openstack-nova-compute	2%	100MB
Controller	openstack-nova-conductor	12%	540MB
	openstack-nova-scheduler	5%	250MB
	openstack-nova-compute	2.4%	130MB

Em relação ao experimento com o Nova, ocorreram falhas de comunicação entre processos. Os processos de um mesmo módulo se comunicam internamente e quando algum deles é desabilitado alguns processos apresentam falhas de comunicação que podem impedir a criação das máquinas virtuais.

Quanto ao consumo de recursos é possível observar que a utilização de CPU chegou a um valor mínimo aproximado de 2% , podendo alcançar um total de 23% no nó de Computação e no nó de Controle há 20% de utilização mínima chegando em torno de 58% de utilização. Assim como no experimento com o Neutron, o consumo de memória também é bastante elevado, ficando em torno dos 98% de uso.

A respeito dos processos que não geraram falhas, isso se deve possivelmente a carga de trabalho utilizada não requisitar as funcionalidades desses processos, portanto a geração de falhas por conta desses processos seria improvável. Nota-se entretanto que o consumo de CPU dos processos que não geraram falhas é de aproximadamente metade do consumo total de CPU dos nós da nuvem, indicativo de que esses processos são importantes para o funcionamento da nuvem e que podem ser melhor explorados.

A Tabela 7 mostra a quantidade de falhas que ocorreram durante o teste nos módulos Neutron e Nova. A Coluna 3 exibe a quantidade total de falhas que ocorreram nos nós da nuvem levando em consideração cada módulo, já a Coluna 4 mostra o percentual de falhas observadas.

Tabela 7: Número de falhas por módulo e por nós.

Módulo Alvo	Nó Alvo	# de Falhas	% de Falhas
Neutron	Compute	0	0,00%
	Controller	613	9,32%
	Network	2468	37,52%
Nova	Compute	2124	32,29%
	Controller	1373	20,87%
	TOTAL	6578	100,00%

Nota-se que o Neutron apresentou mais falhas no nó de rede e o Nova apresentou mais falhas no nó de computação O Neutron apresentou 37,52% das falhas enquanto o Nova apresentou 32,29% das falhas .

Entretanto é importante notar que o nó de controle, apesar de não ser responsável por um percentual muito grande das falhas em cada experimento, ele é responsável por manter a API do OpenStack, esta por sua vez é a interface que permite a nuvem operar corretamente. Dessa forma, as falhas mais críticas, sejam elas devido ao módulo do Nova ou do Neutron, acontecem devido a falhas no nó de controle.

6 CONCLUSÃO

Este trabalho apresentou uma ferramenta de injeção de falhas para um ambiente de nuvem configurado com a plataforma OpenStack. A ferramenta proposta injeta falhas nos módulos Nova e Neutron do OpenStack.

Com a ferramenta apresentada foi possível gerar mais de 6000 falhas distribuídas ao longo dos módulos do OpenStack Neutron e Nova, essas falhas geraram dados sobre utilização de recursos da nuvem, dados esses que podem ser utilizados para modelos de detecção e previsão de falhas.

Apesar do sucesso na injeção de falhas no ambiente do OpenStack, observa-se que vários processos da nuvem mesmo desabilitados não geraram falhas durante o período de experimentação, outras abordagens para injetar falhas podem se sair melhor nessa tarefa, como por exemplo, a mutação do código da nuvem, abordagem essa, que poderá ser explorada em trabalhos futuros.

Por fim, a primeira etapa deste projeto pode ser considerada bem-sucedida e os trabalhos futuros visam desenvolver ainda mais essa ferramenta para alcançar resultados ainda melhores.

7 REFERENCES

[1] Abderrahim, W. and Choukair, Z. PaaS dependability integration architecture based on cloud brokering. Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2016.

[2] Correia, F. Definição de computação em nuvem segundo o NIST. Plataforma Nuvem, 2011. Disponível em: https://plataformanuvem.wordpress.com/2011/11/21/definicao-de-computacao-em-nuvem-segundo-o-nist/. Acesso em: 05/06/2024.

[3] Crouzet, Y. and Kanoun, K. System Dependability: Characterization and Benchmarking. Elsevie, 2012. Disponível em: <https://hal.science/hal-00761042/document>. Acesso em: 24/02/2024.

[4] Cunha, J. M. Q. Fault injection for the evaluation of critical systems. Disponível em: <https://hdl.handle.net/1822/27841>. Acesso em: 04/03/2024.

[5] Li, Y., Jiang, Z. M., Li, H., Hassan, A. E., He, C., Huang, R., Zeng, Z., Wang, M. and Chen, P. Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform: An AIOps Solution. ACM Transactions on Software Engineering and Methodology. Volume 29. Issue 2. 2020.

[6] Open Source Cloud Computing Platform Software - OpenStack. Disponível em:

https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. Acesso em:12/02/2024

[7] Use Cases - Applications of the Technology - OpenStack. Disponível em: https://www.openstack.org/use-cases/. Acesso em: 12/06/2024.

[8] Watanabe, Y., Otsuka, H., Sonoda, M. Online Failure Prediction in Cloud Datacenters by Real-time Message Pattern Learning. IEEE 4th International Conference on Cloud Computing Technology and Science. 2012.

[9] Sharma, O., Verma, M., Bhadauria, S. and Jayachandran, P. A Guided Approach Towards Complex Chaos Selection, Prioritisation and Injection. 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), 2022.

[10] Cotroneo, D., Simone, L., Martino, A., Liguori, P. and Natella, R. Enhancing the Analysis of Error Propagation and Failure Modes in Cloud Systems. 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018.

[11] Cotroneo, D., Simone, L., Martino, A., Liguori, P. and Natella, R. Fault Injection Analytics: A Novel Approach to Discover Failure Modes in Cloud-Computing Systems. IEEE Transactions on Dependable and Secure Computing, v. 19, n. 3, p. 1476–1491, 2022.

[12] Lobato, R., Silva, F., Spolon, R., Manacero, A., Tamashiro, C. and Cavenaghi, M. Faults in Cloud Environment. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9476633>. Acesso em: 16/08/2023.

[13] Poltronieri, F., Tortonesi, M. and Stefanelli, C. ChaosTwin: A Chaos Engineering and Digital Twin Approach for the Design of Resilient IT Services. 17th International Conference on Network and Service Management (CNSM), 2021.

[14] Almeida, P., Cerveira, F., Barbosa, R. and Madeira, H. ucXception: A Framework for Evaluating Dependability of Software Systems. 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), 2022.

[15] Flora, J., Gonçalves, P., Teixeira, M. and Antunes, N. A Study on the Aging and Fault Tolerance of Microservices in Kubernetes. IEEE Access, v. 10, p. 132786–132799, 2022.

[16] Mendonça, J., Lima, R., Matos, R., Ferreira, J. and Andrade, E. Availability Analysis of a Disaster Recovery Solution Through Stochastic Models and Fault Injection Experiments. 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), 2018.

[17] Gao, Y., Dou, W., Wang, D., Feng, W., Wei, J., Zhong, H. and Huang, T. Coverage Guided Fault Injection for Cloud Systems. 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), 2023.

[18] Devi, R. and Muthukannan, M.. Self-healing Fault Tolerance Technique in Cloud Datacenter. 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021.