Planejamento de Arquiteturas Resilientes em Kubernetes: Uma abordagem baseada em Tempo de Recuperação e Consumo Elétrico

Iure Fé[1], Luis Guilherme Silva[1], André Soares[1], Paulo Rego[2] e Francisco Airton Silva[1]

[1]Universidade Federal de Piauí (UFPI)

[2]Universidade Federal do Ceará (UFC)

{iure.fe,luis.e,andre.soares,faps}@ufpi.edu.br, paulo@dc.ufc.br

Resumo. *A arquitetura de microsserviços tem sido cada vez mais utilizada para implantar sistemas na nuvem. No entanto, essas arquiteturas continuam sujeitas aos efeitos de desastres. Devido à quantidade de cenários e elementos de configuração possíveis, preparar sistemas de microsserviços para recuperação em um tempo restrito e sem grandes impactos no consumo elétrico é um desafio. Este artigo propõe um modelo capaz de quantificar o tempo de recuperação e o consumo elétrico deste tipo de sistema para auxiliar no planejamento de sistemas resilientes e ecologicamente conscientes. Os resultados do modelo identificaram os elementos mais significativos da arquitetura e delimitaram intervalos com maiores melhorias relativas.*

Abstract. *Microservices architecture has been increasingly used to deploy systems in the cloud. However, these architectures are still prone to the effects of disasters. Due to the number of possible scenarios and configuration elements, preparing microservices systems for recovery in a restricted time without major impacts on electrical consumption is challenging. This article proposes a model capable of quantifying the recovery time and electrical consumption of this type of system to assist in planning resilient and ecologically aware systems. The model results identified the most significant elements of the architecture and delimited intervals with greater relative improvements.*

1. Introdução

As aplicações nativas da nuvem estruturadas em torno de microsserviços oferecem múltiplas vantagens, principalmente devido à sua arquitetura modular. Essa modularidade não apenas simplifica os processos de desenvolvimento, mas também facilita a distribuição desses serviços entre nós em vários *data centers* (DCs), o que além de trazer melhorias no desempenho também oferecem melhoria na disponibilidade dos sistemas [Bhavsar et al. 2023]. Outro benefício das arquiteturas baseadas em microsserviços é o seu potencial para reduzir o consumo de energia. O consumo global de energia atribuído às atividades de tecnologia da informação e comunicação é de aproximadamente 2%, com projeções indicando um aumento para 14% até 2040 [Wang 2022]. A utilização da conteinerização oferece um caminho para redução de energia por meio da capacidade de escalar a arquitetura rapidamente.

No entanto, as aplicações de microsserviços continuam sujeitas a desastres, que podem se originar de eventos induzidos pelo homem, como ataques cibernéticos, terrorismo ou sabotagem, bem como de ocorrências naturais, como inundações, furacões ou incêndios [Trivedi and Xia 2015]. As soluções de recuperação de desastres (DR) são fundamentais para garantir a continuidade das operações organizacionais em meio a tais eventos. O Tempo de Recuperação (RT) é definido como a duração necessária para que o sistema retome os níveis mínimos de serviço após uma interrupção. A análise deste período é crucial para avaliar a eficácia dos mecanismos de sobrevivência de um sistema [Trivedi and Xia 2015, Longo et al. 2017].

Deve ser levado em conta que as estratégias de DR muitas vezes utilizam de métodos que dependem redundância de recursos [Welsh and Benkhelifa 2020, Andrade and Nogueira 2019]. A adição de recursos adicionais tem um impacto direto no consumo elétrico da arquitetura [Welsh and Benkhelifa 2020]. Duplicar ou triplicar todos os componentes da arquitetura irá aumentar o consumo elétrico em todo o tempo de utilização da aplicação. No entanto, o aumento do consumo pode ser maior que o necessário para o RT desejado, ou mesmo ao escalar componentes não fundamentais, resultar no RT inferior ao mínimo esperado. Alcançar um equilíbrio entre recuperação de desastres e consumo elétrico exige o gerenciamento de uma ampla gama de parâmetros das aplicações e infraestrutura [Longo et al. 2017]. Observar o comportamento de recuperação de desastres e os custos elétricos em aplicações de produção muitas vezes é inviável devido ao potencial de replicação dos impactos de desastres na aplicação. A realização de testes de sobrevivência em um ambiente real pode ser proibitivamente cara em termos de tempo de implementação e consumo elétrico.

Embora a literatura existente ofereça diversas estratégias e ferramentas de planejamento para lidar com vários tipos de desastres e considerações relacionadas à confiabilidade e ao consumo de energia em DCs, há uma lacuna na pesquisa que aborda especificamente DR focada em ambientes de microsserviços na nuvem em relação aos custos elétricos. Este trabalho visa preencher esta lacuna, apresentando uma abordagem de planejamento baseada em modelos, que não só auxilia na obtenção de níveis de RT esperados em cenários de desastres repentinos e previsíveis, mas também auxilia na redução do consumo global de energia do sistema. Este artigo oferece duas principais contribuições: (i) modelo capaz de quantificar RT e consumo elétrico para sistemas baseados em microsserviços orquestrados por Kubernetes; (ii) estudo por meio de caso de uso com a identificação dos componentes mais significativos para as métricas de RT e consumo elétrico, bem como a relação dessas métricas. O modelo proposto possibilita aos projetistas de sistemas orquestrar sua infraestrutura para alcançar o nível de sobrevivência desejado de forma consciente do consumo energético de cada opção.

2. Trabalhos Relacionados

Esta seção sintetiza esforços de pesquisa pertinentes nas áreas de redução do tempo de recuperação de desastres e minimização do consumo de energia no planejamento de sistemas em nuvem. Ramasamy et al. [Ramasamy et al. 2022] desenvolveu um método de aprimoramento de redundância para clusters Kubernetes para reforçar a alta disponibilidade em meio a interrupções de serviço. Este método emprega um cluster

de reserva para facilitar a troca de serviços com base na disponibilidade da aplicação, eliminando a necessidade de intervenção do usuário na restauração do cluster pós-falha. No entanto, o trabalho não abordou o consumo elétrico da abordagem nem considerou métodos de planejamento da infraestrutura.

Nong et al. [Nong et al. 2022] propuseram uma estrutura de sobrevivência, formulada em CloudSim, para alocação eficiente de máquinas virtuais (VMs). Esta estrutura abordou falhas de rack migrando VMs para racks operacionais, onde as VMs são convertidas em imagens e reiniciadas como contêineres. Essa abordagem teve sucesso com VMs, no entanto, não elaboradas considerações sobre microserviços ou consumo elétrico. Bhavsar et al. [Bhavsar et al. 2023] introduziu uma abordagem de recuperação de desastres do Kubernetes baseada na utilização de ferramentas de backup e restauração da Amazon Web Services (AWS), essa abordagem demonstrou a viabilidade do uso dos serviços da nuvem para automatizar as tarefas de DR. Diferente da abordagem apresentada nesse trabalho, não foram considerados o consumo elétrico, nem a possibilidade de variação do tamanho da infraestrutura.

Trivedi et al. [Trivedi and Xia 2015] apresentaram uma abordagem baseada em modelo utilizando Stochastic Reward Net (SRN) e Continuous Time Markov Chain (CTMC) para avaliar as capacidades de recuperação do sistema dentro de intervalos especificados pós-desastre. Essa abordagem capturou efetivamente o comportamento das métricas de desempenho em sistemas de grande escala durante e após cenários de desastre. No entanto, também não foram analisadas as consequências do consumo elétrico da arquitetura durante o desastre. No domínio da redução do uso de recursos, Hamadah [Hamadah and Aqel 2019] discutiu qualitativamente as possibilidades de recuperação rápida usando ferramentas fornecidas por serviços em nuvem, com foco no consumo elétrico. Isa et. al. [Isa et al. 2019] proporam uma infraestrutura de computação em neblina energeticamente eficiente e resiliente para aplicações de monitoramento de saúde, modelada usando Programação Linear Inteira Mista (MILP). Este modelo levou em consideração o consumo de energia dos componentes ociosos e ativos, demonstrando a influência da implantação de data centers distribuídos na pegada energética dos sistemas de monitoramento de saúde. No entanto, nenhum dos trabalhos com foco no consumo elétrico levou em conta o uso de contêineres, nem as especificidades dos orquestradores.

3. Conceitos Básicos

Grandes sistemas baseados em microsserviços, podem utilizar centenas de contêineres, para auxiliar no controle desse tipo de ambiente são utilizados orquestradores. A abordagem desse trabalho foi baseada no Kubernetes, que é um dos principais orquestradores desse tipo de sistema [Bhavsar et al. 2023]. A arquitetura do Kubernetes é apresentada na Figura 1. Nessa arquitetura a aplicação se estende por vários DCs em um ambiente de nuvem. A infraestrutura básica do Kubernetes é composta por dois tipos de Nós principais: o Plano de Controle (CP) e os Nós de Trabalho (WN).

Neste ambiente, os containers com aplicações estão alocadas em Pods. Um Pod é uma agregação de um ou mais contêineres que encapsulam aplicações que compartilham recursos comuns de rede e armazenamento. No contexto de uma arquitetura de microsserviços, existem vários Pods, cada Pod pode abrigar um microsserviço único ou

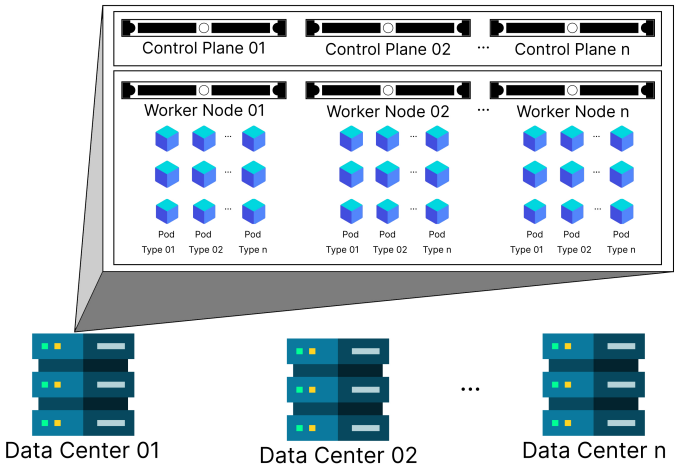


Figura 1. Arquitetura de um cluster Kubernetes.

pode fazer parte de um conjunto de réplicas de um microsserviço. O conjunto de réplicas de um tipo de Pod é chamado de *deployment*. O CP orquestra e sincroniza as operações de WNs e Pods em uma arquitetura Kubernetes. O segmento WN compreende nós que também podem ser hardwares dedicados ou VMs, os WN executam os microsserviços.

A arquitetura distribuída, conforme ilustrada na Figura 1, permite que os Pods sejam distribuídos estrategicamente em vários DCs em quantidades diferentes. Essa diversificação espacial proporciona nas aplicações maior resiliência, permitindo-lhe resistir a possíveis desastres que impactam um ou vários DCs [Ramasamy et al. 2022]. No caso de falha do DC, há uma diminuição inevitável no número de Pods operacionais naquele momento específico. Posteriormente, o Kubernetes tenta compensar essa deficiência iniciando a instanciação dos Pods necessários nos DCs funcionais restantes. Este processo de ativação do Pod, que abrange a inicialização do contêiner e do aplicações, não é instantâneo e está sujeito a restrições temporais [Kubernetes 2023]. A duração necessária para esta recuperação depende da extensão do desastre, tempo de instanciação, quantidade de nós, quantidade de Pods da aplicação, entre outros. A má configuração pode exceder o RTO aceitável para aplicação. Para reduzir o tempo de recuperação podem ser incorporados Pods e Nós sobressalentes ao sistema. Esta estratégia K-out-of-N (KooN) é amplamente utilizada. Em cenários de desastre, a presença de Pods e Nós adicionais reduz o RT, embora incorra em custos elétricos adicionais.

4. Modelo

Essa seção apresenta o modelo utilizado para representar a infraestrutura de sistemas de microsserviços orquestrados por Kubernetes. A escolha pela modelagem em Generalized Stochastic Petri Net (GSPN) decorre da sua capacidade de representar sistemas que possuam eventos com concorrência, assincronicidade, distribuição, determinísticos, ou estocásticos [Maciel 2023]. Essa capacidade das GSPN permite a possibilidade de representar os eventos e comportamentos do sistema e do ambiente, como o esperado para sistemas que sejam alvo e reajam a desastres. Além disso, modelos GSPN tem sido utilizado com sucesso em outros trabalhos para representar desastres de sistemas em distribuídos [Andrade and Nogueira 2019, Longo et al. 2017, Trivedi and Xia 2015]. Outros

modelos, como CTMCs e Diagrama de Blocos de Confiabilidade, também poderiam ser utilizados, no entanto, devido ao grande número de possíveis estados tornaria o modelo muito difícil de utilizar e adaptar.

A Figura 2 exibe o modelo GSPN para a arquitetura da Figura 1. O modelo foi construído levando em conta os elementos dependentes de tempo e de configuração da infraestrutura do Kubernetes em cenários de recuperação. As transições deste modelo são detalhadas na Tabela 1. Para melhorar a compreensão, o bloco do DC foi subdividido em três sub-blocos. Dentro do bloco do CP, temos os locais CP_{UPn} e CP_{DOWNn}. Os tokens dentro de CP_{UPn} significam o número de Nós CPs funcionais no DC n^{th}. A quantidade inicial de planos de controle disponíveis no DC é indicada pela variável CP_{UPnVal}, enquanto os tokens em CP_{DOWNn} representam a contagem de elementos não funcionais dentro do plano de controle.

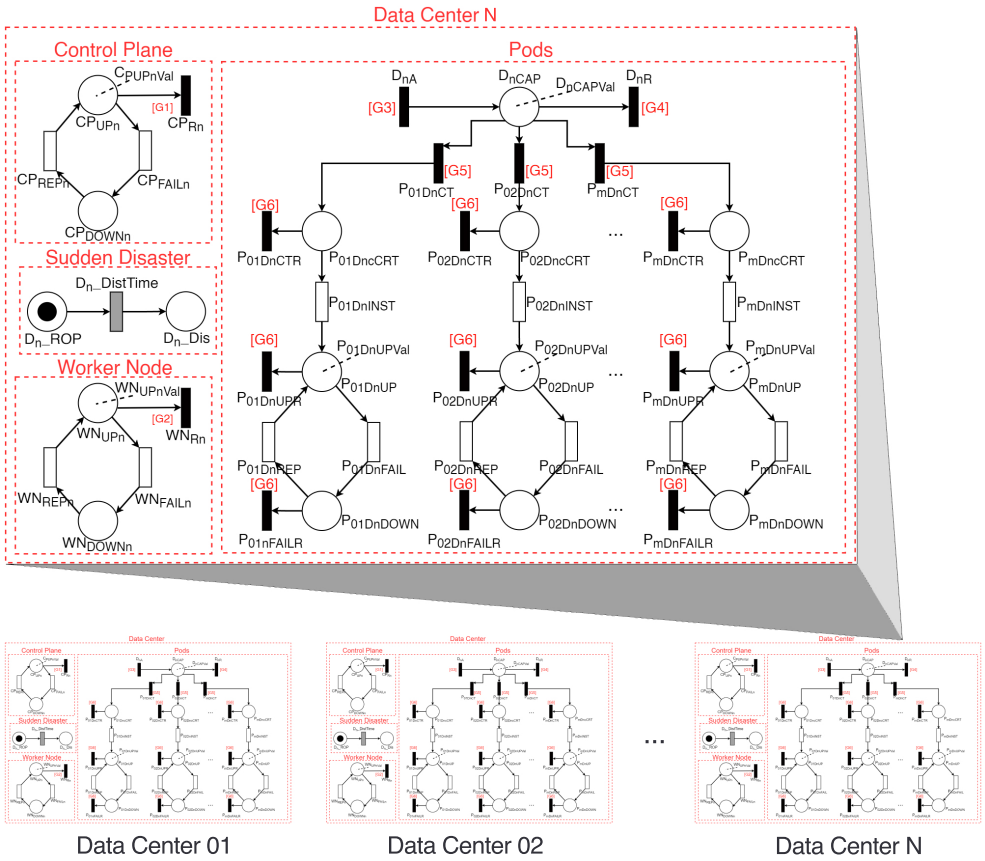


Figura 2. GSPN da infraestrutura Kubernates alocada em Data Center

A transição CP_{FAILn} dentro do modelo simboliza a ocorrência de falha de um CP. Por outro lado, a transição CP_{REPn} representa o reparo de um CP em falha. O bloco CP também incorpora a transição imediata CP_{Rn}, que é condicional à expressão de guarda [g1]. A condição de [g1] depende da métrica buscada no modelo. Em cenários que visam avaliar as métricas estacionárias do sistema, torna-se necessário desativar esta transição, o que ocorre quando se deseja observar o comportamento do sistema na ausência de desastres. Quando o objetivo da execução do modelo for analisar o RT, ou examinar o comportamento transitório do sistema após um desastre, a condição [g1] deve ser a ex-

Tabela 1. Descrição das transições do modelo.

Transição	Descrição	Semantica de Disparo	Peso	Prioridade
WN_{REPn}	Reparo de um WN do DC n	SS	-	1
WN_{FAILn}	Falha de um WN do DC n	IS	-	1
WN_{Rn}	Remoção de um WN do DC n	-	1.0	3
CP_{FAILn}	Falha de um CP do DC n	SS	-	1
CP_{REPn}	Reparo de um CP do DC n	-	1.0	3
CP_{Rn}	Remoção de um CP do DC n	SS	-	1
D_{nA}	Adição de capacidade no DC n	-	1.0	1
D_{nR}	Remoção de capacidade no DC n	-	1.0	2
P_mD_{nCT}	Início da criação de um Pod m no DC n	-	1.0	1
P_mD_{nINST}	Tempo de um Pod m no DC n	IS	-	1
P_mD_{nFAIL}	Falha de um Pod m no DC n	IS	-	1
P_mD_{nREP}	Repair a pod type m in DC n	SS	-	1
P_mD_{nCTR}	Remoção de Pods em criação m no DC n	-	1.0	2
P_mD_{nUPR}	Remoção de um Pod funcional m no DC n	-	1.0	2
P_mD_{nFAILR}	Remoção de um Pod falhado m no DC n	-	1.0	2
$D_n\_Dis\_time$	Evento de desastre rependino	SS	-	1

pressão $\#D_n\_Dis = 1$, o que resulta na remoção dos nós em caso de desastre no DC n. O caractere # representa o cálculo do número de tokens no lugar.

A ocorrência potencial de um desastre no DC n é configurado de um sub-bloco de desastre correspondente, conforme ilustrado na Figura 1. Este sub-bloco compreende dois locais: D_n_ROP e D_n_Dis. Inicialmente, o token em D_n_ROP indica a ausência de um desastre no DC n, enquanto a presença de um token em D_n_Dis significa uma ocorrência de desastre. A transição determinística $D_n\_Dis\_time$ é disparada em um momento predeterminado para análise de um desastre. O bloco WN dentro do modelo representa o agregado de entidades dos Worker Nodes. Ela apresenta um comportamento similar ao descrito para o bloco de CP, com as transições WN_{FAILn}, WN_{REn} representando os tempos de falha e recuperação de WNs respectivamente, enquanto WN_{Rn} representa a remoção de Nós decorrente de desastre, dependendo da configuração de [g2], para resultados estacionários ou de desastre.

O bloco Pods no modelo simboliza os Pods e a capacidade do DC para instanciação de Pods. A capacidade de instanciação disponível para o sistema em um DC específico é indicada pelos tokens em D_{nCAP}. A capacidade inicial é atribuída através da variável $D_{nCAPVal}$, cujo valor é dado pela equação: $D_{nCAPVal} = dc_n total\_cap - used\_cap\_dc_n$, onde: $dc_n total\_cap = ppn\_dc_n \times \#wndc_n\_on$ e

$$used\_cap\_dc_n = \sum_{m \in U} (\#P_mD_{nCRT} + \#P_mD_{nUP} + \#P_mD_{nDOWN}), \quad (1)$$

e U = $\{m \mid m$ são os índices dos Pods elegíveis para alocação no DC $n\}$, o que significa que a capacidade disponível para instanciação de Pods é a capacidade total oferecida pelos WNs subtraída da quantidade de Pods atualmente instanciados. A modelagem deste bloco deve abranger os diversos *deployments* que podem ser instanciados dentro do DC. Para cada tipo de pod m que pode ser alocado no DC n, um conjunto correspondente de locais – P_mD_{nUP}, P_mD_{nDOWN}, P_mD_{nCRT} – juntamente com as transições diretamente associadas, também deve ser vinculada à capacidade do DC (indicado pelo local D_{nCAP}). Os Pods ativos do tipo m no DC n são representados pelos tokens em P_mD_{nUP}. A contagem inicial desses tokens é determinada pela variável P_mD_{nUPVal}. No caso de falha do Pod, a transição P_mD_{nFAIL} é acionada, transferindo um token de P_mD_{nUP} para

P_mD_{nDOWN}. A recuperação do Pod com falha é dada pela transição P_mD_{nREP}.

O bloco de Pods também reflete as mudanças nos WNs dos datacenters associados. A introdução de capacidade adicional para instanciação de Pod no DC é modelada pela ativação da transição D_{nA} por meio da expressão de guarda [g3]. Esta relação é expressa matematicamente na Condição: $inst\_cap\_dc_n < dc_ntotal\_cap$, onde $inst\_cap\_dc_n = used\_cap\_dc_n + \#D_{nCAP}$. Portanto, se a capacidade total, instanciada e disponível, for menor que a oferecida pelos WN, a capacidade disponível deve ser incrementada. Já a retirada da capacidade disponível é dada pelo disparo da transição D_{nR}, habilitada pela expressão de guarda [g4], que possui a condição: $inst\_cap\_dc_n > dc_ntotal\_cap$. Após a ocorrência de um desastre que reduza a quantidade mínima de Pods min_pod_m, o Kubernetes orquestra a instanciação de Pods em nós alternativos. Esta resposta operacional é representada pela ativação da transição imediata P_mD_{nCT}, controlada pela expressão do guarda [g5]. Essa expressão é regida pela condição $(alocated\_pod_{\_m} < min\_pod_m) \wedge (crt\_pods\_dc_n < WN_{UPn})$, onde:

$$alocated\_pod_{\_m} = \sum_{n \in T} (\#P_mD_{nCRT} + \#P_mD_{nUP} + \#P_mD_{nDOWN}), \tag{2}$$

e, $crt\_pods\_dc_n = \sum_{m \in U} \#P_mD_{nCTR}$, onde T = $\{n \mid n$ são os índices dos DCs elegíveis para alocação de Pods $m\}$. A expressão de guarda [g5] é, portanto, habilitada quando a quantidade de Pods de um determinado *deployment* m é menor que a quantidade mínima configurada e o datacenter n tem capacidade disponível de instanciação. O disparo da transição P_mD_{nCT} resulta na remoção de um token de D_{nCAP} e criação de um em P_mD_{nCTR}, iniciando da criação de um Pod. A duração da instanciação do Pod é determinada pela transição P_mD_{nINST}. Após o disparo de P_mD_{nINST}, um token é transferido de P_mD_{nCTR} para P_mD_{nUP}, recuperando a quantidade configurada de Pods.

A remoção de Pods é acionada devido a falhas de nós individuais ou ao colapso de um DC inteiro. Esses eventos são modelados pela ativação das transições P_mD_{nCTR}, P_mD_{nUPR} e P_mD_{nFAILR}, governadas pela expressão de guarda [g6]. Esta expressão, integrante da lógica operacional do modelo, é formulada conforme a condição: $(alocated\_pod_{\_m} > min\_pod_m) \vee (inst\_cap\_dc_n > dc_ntotal\_cap)$, que verifica se há mais Pods alocados que o especificado, ou se a quantidade de Pods é maior que a suportada pela infraestrutura atual.

Para avaliar a eficácia de uma configuração, um conjunto de métricas foram desenvolvidas para o modelo. Essas métricas abrangem: a contagem de Pods ativos, o número de WNs ativos, RT e consumo de energia. A quantidade de Pods ativos é obtido pela Equação (3):

$$total\_exp\_pod = \sum_{n \in N} \sum_{m \in U} E(\#P_mD_{nUP}), \tag{3}$$

onde N = $\{n \mid n$ São os índices dos DCs$\}$. A expressão $E(\#P_mD_{nUP})$ define o número esperado de tokens no lugar $\#P_mD_{nUP}$, esta expressão é apresentada na Equação: $E(\#P_mD_{nUP}) = \sum_{i=1}^{n} P(\#P_mD_{nUP} = i) * i$, onde $P(\#p_mdc_n\_on = i)$ representa a probabilidade de haver i tokens no lugar $p_mdc_n\_on$ [Maciel 2023].

A Equação (4) apresenta o consumo elétrico do sistema, ele leva em conta a potência média usada pelos nós CP (ρ_n) e WN (σ_n), bem como a potência usada de cada Pod ($\alpha_{n,m}$), e a potência média para criação de cada Pod ($\beta_{n,m}$), juntamente com o valor

do tempo necessário para criação ($\tau_{n,m}$):

$$P = \sum_{n \in N} \Big(E(\#cpdc_n\_on) * \rho_n) + E(\#wndc_n\_on) * \sigma_n \Big) + \sum_{n \in N} \sum_{m \in U} \Big(E(\#p_m dc_n\_on) * \alpha_{n,m} + \frac{E(\#p_m dc_n\_crt) * \beta_{n,m}}{\tau_{n,m}} \Big), \quad (4)$$

ela leva em consideração o consumo elétrico de nós e Pods ativos no sistema, bem como o consumo de instanciação de novos Pods na arquitetura. O consumo elétrico individual dos componentes foi baseada em modelos de consumo elétrico de componentes de arquitetura existente na literatura [Lin et al. 2020]. Já o RT pode ser obtido por meio do tempo médio de absorção do modelo adicionando a condição de guarda $(D_k Dis = 0) \wedge activePods_m \leq slaPod_m$ nas transições CP_{FAIL_n}, WN_{FAIL_n}, e $P_m D_{nFAIL}$, onde: $activePods\_m = \sum_{n \in T}(\#P_m D_{nUP})$, no qual, T = $\{n \mid n$ São os índices dos DCs capazes de executar o pod $m\}$. Essa adição faz com que essas transições estejam habilitadas enquanto o sistema não alcançar o valor mínimo de SLA para cada *deployment* m, no caso de um desastre no DC k. Após alcançar o mínimo de Pods o modelo desabilita as transições levando o modelo a absorção, sendo o tempo para a absorção o RT.

5. Estudo de Caso

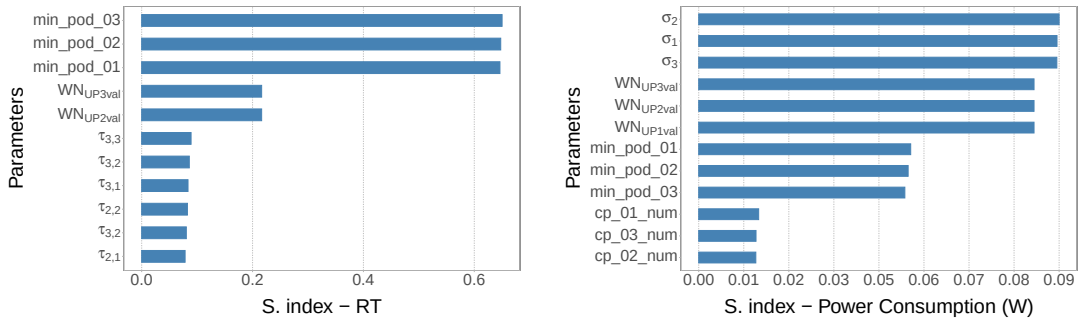
Esta seção delineia os resultados derivados da análise dos modelos propostos na Seção 4. O foco dessa análise é apresentar um uso do modelo para ajustar a relação entre RT e consumo elétrico. Para isso, serão identificados os elementos mais relevantes da infraestrutura para então variar esses elementos comparando o impacto no consumo elétrico e RT obtido em diferentes configurações. Além disso, também será apresentada a utilização do modelo para identificação de cenários mais exigentes em relação ao máximo tempo de RT. Para efeitos desta análise, a configuração do sistema segue os parâmetros descritos na Tabela 2, derivada de uma série de fontes acadêmicas e técnicas [Gomes et al. 2022, Lin et al. 2020]. O sistema avaliado é composto por três DCs, cada um equipado com quatorze nós dedicados ao sistema. Desses nós, três são alocados como CP, com os restantes servindo como WN. O sistema opera com três *deployments* que devem possuir ao menos 50 Pods ativos para o adequado funcionamento do sistema. Não há restrições para a criação dos Pods dos *deployments* em qualquer um dos nós. Os resultados foram calculados usando a ferramenta de modelagem Mercury [Maciel et al. 2017], que permite o desenho e obtenção dos resultados das métricas de GSPNs.

Tabela 2. Parâmetros do Sistema.

Parâmetro	Valor
cp_1_mttf, $wn\_d_1\_mttf$, cp_2_mttf, $wn\_d_2\_mttf$, cp_3_mttf, $wn\_d_3\_mttf$	1236 h
pod_{01}_mttf, $pod_{\_03}\_mttf$	1258,0 h
pod_{02}_mttf	768 h
cp_{01}_mttr, $wn\_d_{01}\_mttr$	1 h
pod_{01}_mttr,pod_{02}_mttr, $pod_{\_03}\_mttr$	0,238 h
$\tau_{1,1}$, $\tau_{1,2}$, $\tau_{1,3}$, $\tau_{2,1}$, $\tau_{2,2}$, $\tau_{2,3}$, $\tau_{3,1}$, $\tau_{3,2}$, $\tau_{3,3}$	0,1 h
$\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{1,3}$, $\alpha_{2,1}$, $\alpha_{2,2}$, $\alpha_{2,3}$, $\alpha_{3,1}$, $\alpha_{3,2}$, $\alpha_{3,3}$	23,09 W
ρ_1, σ_1,ρ_2, σ_2,ρ_3, σ_3	170 W
$\beta_{1,1}$, $\beta_{1,3}$, $\beta_{2,1}$, $\beta_{2,3}$,$\beta_{3,1}$, $\beta_{3,3}$	0,01 W
$\beta_{1,2}$, $\beta_{2,2}$, $\beta_{2,3}$	0,02 W
ppn_dc_1, ppn_dc_2, ppn_dc_3	10

Este estudo de caso investiga os elementos de configuração e de arquitetura mais impactantes para redução do RT e consumo elétrico por meio de análise de sensibilidade. A identificação dos elementos mais impactantes auxilia no levantamento de quais componentes devem ser priorizados para alteração das métricas de interesse. A obtenção do RT foi realizado por meio da configuração de desastre para ocorrer 0,75 horas após o início da execução. O cenário de desastre representa um evento catastrófico hipotético que torna um DC totalmente inoperante, reduzindo consequentemente a quantidade total de Pods do sistema.

A métrica de RT é computada por meio do tempo médio de absorção, que acontece quando o sistema alcança a configuração mínima desejada de Pods. Já a métrica de consumo elétrico foi obtida por meio da computação do estado estacionário do modelo, o que representa o valor esperado de consumo na maior parte do tempo da utilização do sistema, visto que, diferentes das falhas e recuperações comuns da operação diária de sistemas (presentes no modelo), os desastres são eventos excepcionais e de maior impacto [Longo et al. 2017]. Os parâmetros específicos utilizados nos modelos estão delineados na Tabela 2. Os Acordos de Nível de Serviço (SLAs) do sistema foram definidos para garantir 150 Pods ativos, sendo 50 de cada *deployment*.



(a) Tempo de Recuperação (b) Potência utilizada

Figura 3. Análise de Sensibilidade.

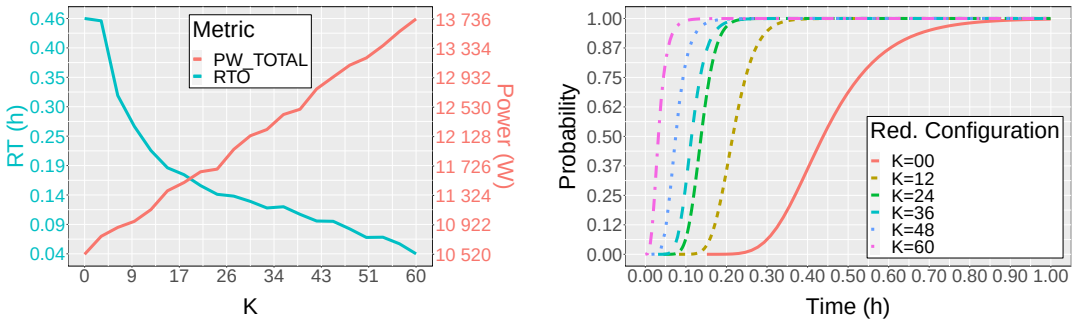
A técnica de análise de sensibilidade empregada foi a diferença percentual [Pinheiro et al. 2021], que envolve variar um parâmetro em uma faixa específica e depois identificar seus valores mais altos e mais baixos, conforme descrito na Equação 5:

$$S_\theta\{Y\} = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}}, \tag{5}$$

onde max Y(θ) e min Y(θ) representam os valores mais altos e mais baixos encontrados na métrica de interesse para o parâmetro dentro do intervalo. Foram variados os parâmetros com valores 25% a mais e a menos para cada elemento de configuração da arquitetura. A Figura 3(a) apresenta as métricas mais significativos de RT. Por conta de economia de espaço, foram removidas do gráfico as variáveis com resultados de índices de sensibilidade mais baixos. Podemos ver que a quantidade mínima de Pods obtiveram os maiores impactos, portanto a adição de Pods nos *deployments* terão o maior impacto no RT da aplicação no caso de desastre. Em sequência temos as quantidades de WNs dos DCS, observe que estão apenas presentes os referentes aos DCs 2 e 3, devido ao desastre ter ocorrido no DC 1. Por fim, temos os valores de tempo de instanciação dos Pods (τ_{mxn}),

esses valores podem ser também reduzidos para melhorar o RT do sistema, no entanto, a redução do tempo de instanciação nem sempre é possível, por conta de uso de componentes de terceiros, ou mesmo o custo de reimplementar parte da aplicação para reduzir τ_{mxn} pode ser proibitivo.

A Figura 3(b) apresenta o índice de sensibilidade da potência consumida. O consumo dos WNs (σ_n) são os elementos mais significativos dessa análise, demonstrando que manter a eficiência energética dos WNs é um fator bastante relevante para o consumo elétrico em grandes infraestruturas. Depois do consumo dos hardwares, temos suas quantidades como elementos mais relevantes. Esse fator apresenta o impacto do número de nós ligados, sendo esse um fator configurável e mais facilmente gerenciável da infraestrutura. Em seguida, as quantidades mínimas dos Pods também aparecem como significantes, sendo esse um elemento configurável no Kubernetes pelo administrador de sistemas. Por fim, o número de CPs aparece como elemento significativo, no entanto, com baixo valor quando comparado aos demais elementos. Da mesma forma que para o RT, os demais valores foram omitidos por conta de espaço. A análise comparada das Figuras 3(a) e 3(b) ajudam a direcionar quais componentes merecem atenção durante a busca pelo balanceamento das métricas de RT e Consumo elétrico, no caso dessa arquitetura estudada, o impacto pela adição de Pods acima do valor mínimo do SLA resulta em maiores impactos na redução do RT ao mesmo tempo que não tem um grande impacto no consumo elétrico.



(a) RT x Potência (b) CDF

Figura 4. Análise dos resultados.

Devido ao impacto do número mínimo de pods na análise de sensibilidade, comparamos o resultado de RT e consumo elétrico obtido da variação da quantidade superior à mínima necessária de Pods exigido pelo SLA, ou seja, adicionando K pods ao exigido pelo requisito. A Figura 4(a) apresenta os resultados da comparação. A adição de Pods sobressalentes acarreta redução do RT, no entanto, a mesma adição também apresenta um aumento do consumo elétrico. A análise do gráfico da Figura 4(a) auxilia na identificação do benefício relativo obtido pela adição desse tipo de redundância. É importante observar que a adição de Pods tem impactos diferentes dependendo da quantidade de Pods, até a quantidade de 21 Pods, o RT tem um decréscimo maior por pod ($\approx$ 0,014 horas por pod adicional). Valores de k superiores a 21 Pods temos a redução aproximada de $\approx$ 0,002 horas por Pod adicional, uma redução de $\approx$ 85% a cada Pod adicionado. Enquanto o consumo adicional por Pod se manteve em torno de $\approx$ 52,62 W em todo intervalo observado. Esse tipo de resultado quantitativo é uma ferramenta para o administrador identificar os impactos da redução do RT e menor consumo elétrico, auxiliando na tomada de decisão

da questão de negócio entre criticidade da aplicação e pegada de carbono do sistema.

Como mencionado anteriormente, os valores de RT foram obtidos pelo MTTA do modelo. Em casos que o tempo de recuperação tem pouca tolerância a variação, o modelo também oferece a alternativa de obtenção de resultados com maior certeza de recuperação. A Figura 4(b) apresenta os resultados da Função distribuição acumulada (CDF) para o RT com as configurações de K da Figura 4(a). A CDF resultante permite identificar a probabilidade de um RT em uma configuração de redundância k. Para $k = 0$, embora o RT médio seja 0.45h, a absorção com 100% de probabilidade ocorra apenas com 1,09h. Podemos observar que a medida que aumentamos o valor de K, também encontramos menores RT e menores tempos para 100% de absorção. Com $k = 12$, temos o valor médio de 0.21h, e a completa absorção em 0.45h. Portanto, em casos em que o SLA de RT seja um valor que não possa ser alcançado, o uso do MTTA não é suficiente, e a identificação de possuir 100% de absorção deve ser usado.

6. Conclusão

Neste trabalho foi proposto o uso de SPN para auxiliar no planejamento de RT e consumo elétrico de sistemas de microsserviços baseados em contêineres. Os modelos consideram métricas de quantidade de Pods, RT, e consumo elétrico. Os estudos de caso mostraram a aplicação do modelo para identificação dos elementos de maior impacto no RT e consumo elétrico. Foi possível identificar que a quantidade mínima de Pods é a mais impactante para o RT, por outro lado, o consumo elétrico está intimamente relacionado a configuração do hardware utilizado nos WNs e as suas quantidades. Também foi possível verificar os impactos relativos à variação da quantidade mínima de Pods sobressalentes no RT e Consumo elétrico. Esses impactos relativos demonstraram que a redução em RT pela adição de Pods varia conforme são adicionados novos Pods, chegando a uma redução de 0.85% do decremento de RT por Pod, enquanto a variação do consumo elétrico permaneceu constante. O estudo de caso evidencia uma possibilidade de uso do modelo para o planejamento de sistemas de microserviços. Os valores obtidos das métricas são capazes de guiar o administrador a aplicar redundância nos componentes mais relevantes, evitando consumo desnecessário ou não cumprimento de SLA de recuperação. Como trabalhos futuros pretendemos usar algoritmos de otimização para melhorar ainda mais os resultados variando mais parâmetros. Além disso, podemos apresentar outras aplicações do modelo em diferentes cenários. Outra possível extensão incluí adicionar configuração de Nós físicos escaláveis, bem como busca de estratégias para redução de RT e consumo elétrico.

Referências

Andrade, E. and Nogueira, B. (2019). Performability evaluation of a cloud-based disaster recovery solution for it environments. *Journal of Grid computing*, 17:603–621.

Bhavsar, S., Agrawal, A., Ropalkar, T., Kamdi, P., Hajare, A., Deshpande, S., Rathi, R., and Garg, D. (2023). Kubernetes cluster disaster recovery using aws. In *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–6. IEEE.

Gomes, C., Tavares, E., Junior, M. N. d. O., and Nogueira, B. (2022). Cloud storage availability and performance assessment: a study based on nosql dbms. *The Journal of Supercomputing*, 78(2):2819–2839.

Hamadah, S. and Aqel, D. (2019). A proposed virtual private cloud-based disaster recovery strategy. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 469–473. IEEE.

Isa, I. S. M., Musa, M. O., El-Gorashi, T. E., and Elmirghani, J. M. (2019). Energy efficient and resilient infrastructure for fog computing health monitoring applications. In *2019 21st International Conference on Transparent Optical Networks (ICTON)*, pages 1–5. IEEE.

Kubernetes (2023). Kubernetes production-grade container orchestration. `https://kubernetes.io/`. Accessed: 2023-08-21.

Lin, W., Shi, F., Wu, W., Li, K., Wu, G., and Mohammed, A.-A. (2020). A taxonomy and survey of power models and power modeling for cloud servers. *ACM Computing Surveys (CSUR)*, 53(5):1–41.

Longo, F., Ghosh, R., Naik, V. K., Rindos, A. J., and Trivedi, K. S. (2017). An approach for resiliency quantification of large scale systems. *ACM SIGMETRICS Performance Evaluation Review*, 44(4):37–48.

Maciel, P., Matos, R., Silva, B., Figueiredo, J., Oliveira, D., Fé, I., Maciel, R., and Dantas, J. (2017). Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*, pages 50–57. IEEE.

Maciel, P. R. M. (2023). *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. CRC Press.

Nong, M., Huang, L., and Liu, M. (2022). Allocation of resources for cloud survivability in smart manufacturing. *ACM Transactions on Management Information Systems (TMIS)*, 13(4):1–11.

Pinheiro, T., Oliveira, D., Matos, R., Silva, B., Pereira, P., Melo, C., Oliveira, F., Tavares, E., Dantas, J., and Maciel, P. (2021). The mercury environment: a modeling tool for performance and dependability evaluation. In *Intelligent Environments 2021: Workshop Proceedings of the 17th International Conference on Intelligent Environments*, volume 29, page 16. IOS Press.

Ramasamy, B., Na, Y., Kim, W., Chea, K., and Kim, J. (2022). Hacm: High availability control method in container-based microservice applications over multiple clusters. *IEEE Access*, 11:3461–3471.

Trivedi, K. S. and Xia, R. (2015). Quantification of system survivability. *Telecommunication Systems*, 60:451–470.

Wang, J. C. (2022). Understanding the energy consumption of information and communications equipment: A case study of schools in taiwan. *Energy*, 249:123701.

Welsh, T. and Benkhelifa, E. (2020). On resilience in cloud computing: A survey of techniques across the cloud domain. *ACM Computing Surveys (CSUR)*, 53(3):1–36.