K8s-DT: Um Gêmeo Digital do Kubernetes Baseado em Modelos Estocásticos

Iúre de Sousa Fé[1], Luiz Fernando Bittencourt[2], Paulo Maciel[3] e Francisco Airton Silva[1]

[1]Universidade Federal do Piauí – PI – Brasil

[2]Universidade Estadual de Campinas – SP – Brasil

[3]Universidade Federal de Pernambuco – PE – Brasil

{iure.fe,faps}@ufpi.edu.br

bit@ic.unicamp.br

prmm@cin.ufpe.br

Resumo. *Aplicações em microsserviços recorrem ao autoscaling para equilibrar custo e qualidade de serviço, mas políticas reativas como o HPA do Kubernetes tendem a adicionar réplicas tardiamente e dependem de limiares ajustados por tentativa e erro. Esse atraso eleva o tempo de resposta durante picos de carga e desperdiça recursos quando a demanda diminui. Para enfrentar essa limitação, acoplamos um gêmeo digital baseado em Stochastic Petri Nets (SPN) ao cluster Kubernetes e conduzimos análises what-if guiadas por um SLA explícito de tempo de resposta. O objetivo é aplicar dinamicamente configurações de réplicas que atendam ao SLA com menor consumo de recursos. Inicialmente, validamos as previsões do gêmeo digital com 95% de confiança em relação a um cenário real em nuvem. Em seguida, avaliamos dois cenários com perfis de carga distintos, nos quais o gêmeo digital atendeu ao SLA e reduziu o número médio de réplicas em* $32{,}83\%$ *(carga de alta variância) e* $18{,}7\%$ *(carga aleatória), em comparação com os melhores baselines do HPA. Dessa forma, a abordagem reduziu o uso de recursos sem comprometer o cumprimento do SLA e evitou a calibração dependente da carga baseada em limiares fixos no HPA.*

Abstract. *Microservice-based applications rely on autoscaling to balance cost and quality of service; however, reactive policies such as Kubernetes HPA tend to add replicas too late and depend on thresholds that are often tuned through trial and error. This delay increases response time during workload peaks and wastes resources when the load subsides. To address this gap, we couple a Digital Twin based on Stochastic Petri Nets (SPN) with the Kubernetes cluster and conduct SLA-driven what-if analyses focused on explicit response-time constraints. Our goal is to dynamically apply replica configurations that satisfy the SLA while minimizing resource consumption. We first validate the predictions produced by the Digital Twin against a real cloud environment with 95% confidence. Then, we develop two experimental scenarios with distinct workload profiles, in which the Digital Twin meets the SLA while reducing the average number of replicas by* $32{,}83\%$ *(high-variance workload) and* $18{,}7\%$ *(random workload) when compared to the best HPA baselines. Overall, our approach reduces resource usage without compromising SLA compliance and eliminates the need for workload-dependent threshold calibration in HPA.*

1. Introdução

O autoscaling é um mecanismo essencial para aplicações modernas e ambientes de microsserviços em nuvem, pois permite ajustar dinamicamente a infraestrutura para atender requisitos de Qualidade de Serviço (QoS) com uso mais adequado de recursos [Veeck et al. 2025, Vasumathi et al. 2025a, Lakshan and Hussain 2025]. Nesse contexto, o Kubernetes (K8s) atua como a principal plataforma de orquestração, automatizando a implantação e o gerenciamento de cargas de trabalho. Entre seus mecanismos nativos, destaca-se o Horizontal Pod Autoscaler (HPA), que ajusta dinamicamente o número de réplicas com base em métricas como uso de CPU e memória [Vasumathi et al. 2025a].

No entanto, realizar um autoscaling adequado impõe desafios significativos. A principal dificuldade reside na natureza reativa dos mecanismos tradicionais como o HPA, que reagem apenas quando um limiar de métrica é atingido [Veeck et al. 2025]. Essa reatividade leva a atrasos nas respostas de escalonamento, resultando em ineficiência de recursos, como o subprovisionamento durante picos de tráfego (degradando o desempenho e violando o SLA) ou o superprovisionamento quando a demanda é baixa (elevando os custos operacionais) [Lakshan and Hussain 2025, Vasumathi et al. 2025a]. Além disso, o HPA tradicional, ao basear-se predominantemente em métricas de CPU e memória, falha em considerar outros fatores importantes de QoS, como vazão e tempo de resposta. Portanto, a escolha do limiar é sensível à carga de trabalho, pois um mesmo limiar pode atender aos critérios de QoS para uma carga e ser insuficiente para outra. Outra dificuldade do uso do HPA é a necessidade de identificar limiares adequados para cada um dos *deployments* do sistema. Neste trabalho, utilizamos variantes do HPA com limiares fixos de CPU como *baseline* experimental por representarem a política reativa nativa mais amplamente adotada no ecossistema Kubernetes, servindo como referência inicial para avaliar os ganhos obtidos com a recomendação orientada por SLA do gêmeo digital [Veeck et al. 2025].

Apesar das abordagens baseadas em *Machine Learning* e *Reinforcement Learning* (RL) serem propostas como soluções proativas para superar a natureza reativa do HPA [Lakshan and Hussain 2025], elas introduzem seu próprio conjunto de limitações e desafios operacionais. Soluções que utilizam RL e Deep Learning frequentemente exigem tempo considerável para aprender e se ajustar às mudanças na carga de trabalho, além de demandarem recursos computacionais substanciais para o treinamento, o que levanta preocupações de escalabilidade, além de necessitar de extensivos dados históricos que nem sempre estão disponíveis [Lakshan and Hussain 2025]. Além disso, modelos preditivos podem ter complexidade em lidar com padrões de carga altamente dinâmicos. O foco limitado em mecanismos de aprendizado contínuo para adaptação dinâmica a padrões de carga em evolução e a questão do trade-off entre a interpretabilidade do modelo e o poder preditivo permanecem como lacunas críticas a serem exploradas [Lakshan and Hussain 2025].

Este trabalho propõe uma solução baseada em Gêmeos Digitais (GD) para apoiar decisões de *autoscaling* em Kubernetes por meio da avaliação de cenários *what-if* orientados por SLA [Abdelrahman et al. 2025, Bittencourt et al. 2024]. Embora práticas consolidadas de observabilidade, benchmarking e teste de carga permitam caracterizar o comportamento de aplicações distribuídas, elas não fornecem diretamente, a cada instante, qual configuração de réplicas deve ser aplicada para atender ao SLA com menor uso de recursos. No K8s-DT, o modelo é continuamente sincronizado com o estado ob-

servado do cluster e utilizado para avaliar configurações candidatas e realizar a aplicação de melhores opções no ambiente real. Assim, a proposta não substitui ferramentas de monitoramento ou teste de carga, mas acrescenta um mecanismo de autoscaling baseado nas métricas monitoradas buscando melhores alternativas por meio da exploração de possíveis cenários.

A construção do GD exige equilibrar fidelidade e custo: modelos muito detalhados aumentam o esforço de modelagem, a necessidade de dados e o custo computacional, enquanto abstrações excessivas podem ocultar efeitos como concorrência entre réplicas, contenção de recursos e variações de carga. Para lidar com esse compromisso, representamos o sistema por meio de SPNs [Maciel 2023a], cujo nível de detalhe pode ser ajustado conforme o cenário analisado. SPNs já têm sido empregadas em sistemas distribuídos, incluindo ambientes orquestrados pelo Kubernetes [Maciel 2023a]. Com o modelo validado, o GD pode extrair métricas de desempenho e uso de recursos e executar análises *what-if* para comparar configurações candidatas antes de sua aplicação no sistema real, reduzindo a necessidade de avaliar diretamente cada alternativa no ambiente em execução [Fé et al. 2025].

O restante deste artigo é organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta a arquitetura da solução proposta, incluindo o funcionamento do Gêmeo Digital e o modelo SPN que o representa. A seção 4 descreve o procedimento e os resultados da validação do modelo. A seção 5 descreve os casos de uso com os resultados práticos da abordagem proposta. Finalmente, a Seção 6 apresenta as conclusões deste trabalho e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Os principais trabalhos relacionados identificados possuem temas que combinem desde gêmeos digitais para Kubernetes até ajustes adaptativos do HPA e técnicas de previsão de carga. O conjunto revela abordagens distintas para decidir *quando* e *quanto* escalar, bem como diferentes níveis de fidelidade para estimar o impacto dessas decisões nas principais métricas operacionais.

Borsatti et al. [Borsatti et al. 2024] apresentam o *KubeTwin*, um gêmeo digital capaz de reencenar cenários em escala e modelar comunicação considerando latências. O vínculo bidirecional permite testar configurações de réplicas no gêmeo e encaminhar "configurações" promissoras ao cluster real, favorecendo experimentos reprodutíveis e estudos de políticas de agendamento e escalonamento. Embora o gêmeo digital possibilite explorar configurações, o estudo não estrutura a busca guidada por metas explícitas de SLA de tempo de resposta, e sim apresenta um framework que pode ser usado para isso.

Em relação ao uso de HPA, Pozdniakova et al. [Pozdniakova et al. 2024] propõem ajustar dinamicamente o alvo de utilização com base em padrões de carga observados. No lugar de um limiar fixo, a meta de utilização passa a variar para resguardar o requisito de tempo de resposta. No entanto, o trabalho foca apenas em um serviço e um alvo de utilização, o que reduz a visibilidade de efeitos em sistemas compostos por múltiplos microsserviços; aqui, avaliamos cenários *multi-serviço* no gêmeo, com latências distribuídas e inspeção de vazão/CPU e réplicas por serviço e no agregado, antes da aplicação real.

Ahmad et al. [Ahmad et al. 2024] introduzem o *Smart HPA*, combinando autoscaling por serviço com um módulo centralizado que atua apenas em condições de restrição

de recursos. O arranjo transfere recursos de serviços superprovisionados para serviços com pouca capacidade, mitigando gargalos sem excesso de sinalização. No entanto, a coordenação é predominantemente heurística e não se apoia em mecanismos proativos, o que não prioriza explicitamente o cumprimento de níveis de serviço e mantém a reação dependente da variação de demanda de cada serviço; neste estudo, a recomendação de parâmetros ocorre pelo teste de cenários possíveis considerando hipóteses voltadas para o cumprimento de SLA.

Para predição de carga com mapeamento direto em réplicas, da Silva et al. [da Silva et al. 2025] propõem o ANN-HS: uma abordagem adaptativa de escalonamento horizontal em clusters Kubernetes que utiliza Redes Neurais Artificiais (ANNs) para previsão de carga, visando melhorar a eficiência no consumo de recursos e otimizar a alocação de réplicas em comparação com o HPA tradicional. Aqui, testamos previamente as decisões por meio do gêmeo e priorizamos configurações que atendem ao SLA sob distribuições de latência e padrões de carga variados, antes da aplicação no ambiente real.

Por fim, Vasumathi et al. [Vasumathi et al. 2025b] integram LSTM ao HPA para antecipar picos, e resolver as ineficiências e os atrasos de escalonamento inerentes aos métodos tradicionais reativos, como o HPA do Kubernetes, propondo um framework preditivo impulsionado por IA para otimizar o uso de recursos e promover a sustentabilidade em ambientes de nuvem. No entanto, a predição permanece acoplada ao HPA reativo e depende de *tuning* e histórico, o que pode não ser preciso em casos de picos inesperados. Neste trabalho, testamos previamente as decisões por meio do gêmeo e possuímos foco direto no requisito do serviço, além disso nossa abordagem é previamente comparada com cenários de distribuições de latência e padrões de carga variados, antes da aplicação no ambiente real.

Os trabalhos revisados mostram abordagens reativas, adaptativas, preditivas e baseadas em gêmeos digitais para o problema de *autoscaling* em Kubernetes. Nesta versão, a avaliação experimental foi realizada com variantes do HPA baseadas em limiares fixos de CPU, por constituírem um *baseline* reativo amplamente utilizado. Além disso, soluções de elasticidade de infraestrutura no ecossistema Kubernetes atuam de forma complementar à proposta, que foca o escalonamento de *pods* no nível dos *deployments*, orientado por SLA e apoiado por análises *what-if*.

3. Arquitetura do sistema

Essa seção descreve a arquitetura utilizada na implantação do gêmeo digital. A arquitetura do gêmeo digital e da aplicação é apresentado na subseção 3.1. Já os modelos que representam o desempenho e utilização de recursos do sistema são apresentados na subseção 3.2.

3.1. Arquitetura de Gêmeo Digital

A Figura 1 apresenta a arquitetura da solução proposta, evidenciando os principais componentes, fluxos de informação e eventos envolvidos no funcionamento do GD. O cluster Kubernetes é responsável pela orquestração das aplicações, cujos microserviços são organizados em *deployments*. Cada *deployment* corresponde a um conjunto de réplicas de um microsserviço. A carga aplicada ao cluster é dinâmica e pode variar de acordo com a quantidade de usuários e com as características das requisições, resultando em períodos de maior ou menor demanda. O Kubernetes processa essas requisições e pro-

duz respostas conforme a capacidade computacional alocada aos *pods* de cada *deployment*. Variações nessa capacidade impactam diretamente métricas como tempo de resposta, vazão e utilização de recursos.

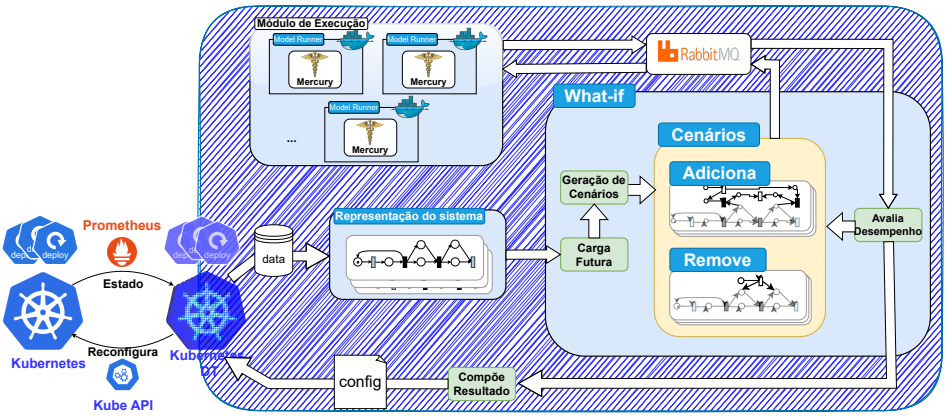

Figura 1. Arquitetura da solução

No K8s-DT são realizadas a atualização do estado dos modelos SPN, a avaliação de cenários *what-if* e a definição de configurações de *feedback* a serem aplicadas no cluster real. O monitoramento contínuo do sistema real é realizado pelo Prometheus, que coleta, armazena e disponibiliza métricas referentes à carga de trabalho, uso de recursos e desempenho do cluster, mantendo tanto os valores instantâneos quanto o histórico dessas medições. Com base nessas informações, o GD do Kubernetes (K8s-DT) é continuamente sincronizado com o estado do sistema real, garantindo que o modelo SPN represente, de forma consistente, o comportamento atual do ambiente monitorado.

A partir dessa representação, o GD pode ser utilizado para identificar configurações mais adequadas à variação de carga, de forma a atender aos SLAs de desempenho com o menor consumo possível de recursos computacionais. A identificação dessas configurações é realizada por meio da avaliação de cenários *what-if*, cujo objetivo é analisar configurações hipotéticas a serem aplicadas no Kubernetes que satisfaçam os SLAs com menor custo de recursos. No contexto dessa análise, a evolução *carga futura* é projetada com base nas últimas requisições. Para essa projeção, utiliza-se o *Holt's Linear Trend Method* [Hyndman and Athanasopoulos 2018], que permite estimar o comportamento da carga até a próxima atualização do Gêmeo Digital. Sempre que a carga projetada for superior à carga atual observada, a projeção é adotada como entrada para os modelos; caso contrário, utiliza-se a carga corrente. Essa estratégia considera sistematicamente o pior caso de carga, reduzindo o risco de violações de SLA durante períodos de crescimento inesperado da demanda.

3.2. Modelo SPN

Essa seção apresenta o modelo base utilizado para representar cada *deployment* da aplicação em um cluster Kubernetes. A Figura 2 ilustra um conjunto de blocos do modelo SPN base, em que cada bloco representa um dos *deployments* de um sistema baseado em Kubernetes. Os blocos SPN *Deploy Base* devem ser replicados de acordo com a quantidade de *deployments* do sistema. A modelagem separada de cada *deployment* permite execuções mais rápidas e resulta em uma solução mais genérica, capaz de ser aplicada a sistemas de diferentes tamanhos. Durante a sincronização, são atualizados em

cada *deployment* os seguintes parâmetros: taxa de chegada, tamanho da fila utilizada, quantidade de pods em processamento, quantidade de pods ociosos e tempo de serviço atual. Nesta versão, chamadas síncronas entre serviços não são modeladas explicitamente como dependências entre *deployments*. Seus efeitos podem aparecer de forma indireta nas métricas sincronizadas do sistema real, mas a representação detalhada dessas interações permanece fora do escopo atual. A abordagem considera serviços concorrentes no nível dos *deployments*, por meio de múltiplos blocos SPN e do paralelismo de processamento permitido pelas réplicas disponíveis em cada serviço. Assim, o modelo captura efeitos de concorrência sobre filas, ocupação e vazão no contexto da análise de *autoscaling*. Os valores estocásticos de tempo são representados pelas transições que são os retângulos cinzas no modelo. Eles representam a distribuição de tempo obtida do sistema. Já as capacidades são representadas pelos lugares (círculos), cujas marcas (quantidade de pontos nos Lugares) representam as quantidades individuais de capacidade e estado de cada *deployment*. A sequência de disparo segue os arcos, que removem os pontos dos lugares antes da transição e deposita no local apontado pela seta.

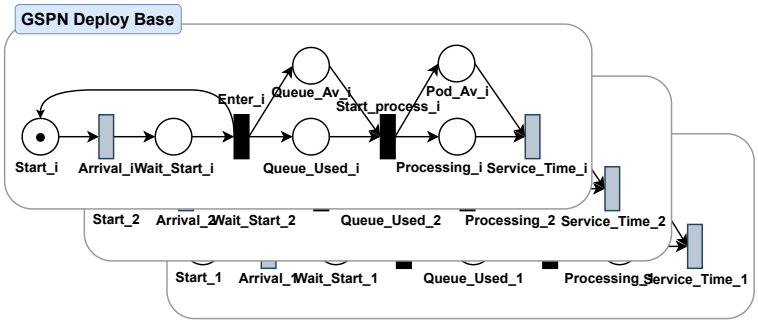

Figura 2. SPNs de deployment

O valor e distribuição do tempo médio entre chegadas do deployment *i* deve ser inserido na transição *Arrival_i*, que é uma distribuição *Single Server* (SS), na qual não há disparos concorrentes. Os lugares *Start_i* e *Wait_Start_i*, juntamente com a transição temporizada *Arrival_i*, representam a entrada de requisições em um *deployment*. O enfileiramento é iniciado com o disparo da transição *Enter_i*, que é uma transição imediata (retângulos pretos), que representa uma ação sem consumo de tempo, sendo utilizada para modelar decisões lógicas, escolhas de política ou sincronizações internas do sistema [Maciel 2023b]. A capacidade de enfileiramento do *deployment* é dada pela quantidade de tokens em *Queue_Av_i*, que também deve ser atualizada com a capacidade ainda disponível nos pods. Já a ocupação da fila no sistema é dada pela quantidade de tokens em *Queue_Used_i*. O início do processamento é representado pelo disparo da transição *Start_process_i*. A quantidade de pods ocupados em processamento é dada pela marcação do lugar *Processing_i*, enquanto a capacidade ainda disponível é representada pela quantidade de tokens em *Pod_Av_i*. O tempo médio de processamento de cada Pod é modelado pela transição temporizada *Service_time_i*, que é uma distribuição *Infinite Server*, que permite processamento simultâneo máximo igual à quantidade de Pods criados.

O modelo SPN base deve manter uma representação aderente ao sistema real, de modo que as variações na carga de trabalho, nas quantidades de pods em execução e nos estados das filas sejam refletidas no modelo. Devido às flutuações de carga decorrentes do aumento ou diminuição da demanda, os tempos de resposta de cada *deployment* e do

sistema como um todo podem variar em função dessa carga. Os valores de tempo de resposta, vazão e utilização dos pods no modelo e no sistema são obtidas de diferentes formas: no sistema real, essas grandezas são obtidas com o Prometheus, enquanto no modelo são derivadas das métricas resultantes da execução das SPNs com a configuração da última atualização. Essa abordagem no DT permite avaliar os valores das métricas nos cenários *What-if*, a partir de variações da atual configuração. Seja $I = \{d_1, d_2, \ldots, d_n\}$ o conjunto de deployments com gêmeo digital no cluster. A vazão (TP_i) de um deployment $i \in I$ no modelo é dada pela métrica apresentada na Equação 1:

$$TP_i = \frac{\mathbb{E}\{\#Processing_i\}}{Service\_Time_i}, \tag{1}$$

onde, $\mathbb{E}$ representa a esperança estatística de tokens no lugar $Processing_i$, definido pela Equação (2):

$$\mathbb{E}\{\#Processing_i\} = \sum_{j=1}^{m} P\{\#Processing_i = j\} * j, \tag{2}$$

onde, $P\{\#Processing_i = j\}$ representa a probabilidade de existirem j tokens no lugar $Processing_i$, e m é o número máximo de tokens no lugar $Processing_i$ [Maciel 2023b]. A vazão total é dada por:

$$TP_{total} = \sum_{i \in I} TP_i, \tag{3}$$

A quantidade de trabalhos no deployment pode ser medida por meio da métrica $Njobs_i$, apresentado na Equação 4:

$$Njobs_i = \mathbb{E}\{\#Wait\_Start_i\} + \mathbb{E}\{\#Queue\_Used_i\} + \mathbb{E}\{\#Processing_i\} \tag{4}$$

O tempo de resposta (RT) é obtido a partir da Lei de Little, $L = \lambda W$ [DeGlopper 1992], em que L representa o número médio de trabalhos no sistema, λ a taxa de chegada efetiva e W o tempo médio de permanência no sistema. Neste trabalho, não utilizamos a taxa de chegada nominal de entrada, mas a taxa efetivamente atendida na configuração avaliada. Assim, adotamos a vazão estimada pelo modelo como aproximação para a taxa de chegada efetiva no cálculo do tempo de resposta. Como o modelo é sincronizado a cada atualização com os parâmetros observados do sistema, o cálculo do RT é realizado sobre a configuração avaliada naquele instante, considerando a vazão correspondente ao fluxo efetivamente processado. Em nosso modelo, o RT de um deployment é dado pela Equação (5):

$$RT_i = Njobs_i \times TP_i^{-1} \tag{5}$$

E para todos os deploys do sistema temos a métrica:

$$RT_{total} = \big(\sum_{i \in I} Njobs_i\big) \times TP_{total}^{-1}, \tag{6}$$

então para o sistema completo é levado em conta o total de trabalhos dentro de todos os deployments e o total da vazão.

3.3. Geração de Cenários

A etapa de *geração de cenários* é acionada a cada atualização do GD. Nessa etapa, as métricas extraídas das SPNs são utilizadas para avaliar mudanças de configuração, como

o ajuste do número de *pods* em um ou mais *deployments*. Para reduzir a quantidade de execuções necessárias, as configurações candidatas são avaliadas por uma busca baseada em *Parallel n-ary Search* [Schlegel et al. 2009], explorando o espaço entre os limites mínimo e máximo de *pods* permitidos por *deployment*.

Cada cenário é avaliado pela execução do modelo SPN correspondente no Módulo de Execução. A comunicação entre o GD e esse módulo é realizada via RabbitMQ, que encaminha as requisições para *containers* responsáveis pela execução da ferramenta Mercury [Pinheiro et al. 2021]. O custo computacional do GD está concentrado nessa etapa e é limitado pela separação do Módulo de Execução, pelo ajuste do número de *containers* e pelo uso de uma busca que reduz o número de configurações avaliadas. Os modelos de adição e remoção de *pods*, ilustrados na Figura 3, estendem o modelo base para avaliar, respectivamente, situações de escalonamento e redução de recursos. As modificações nos modelos para representar o tempo de criação e de remoção foram destacadas na Figura 3. Ambos utilizam as mesmas métricas para *avaliar o desempenho* em relação aos SLAs. Após a avaliação dos cenários, os resultados são consolidados e a configuração mais adequada é traduzida em parâmetros aplicáveis ao cluster Kubernetes real. Esse ciclo é executado continuamente, permitindo a adaptação dinâmica do sistema às flutuações de carga.

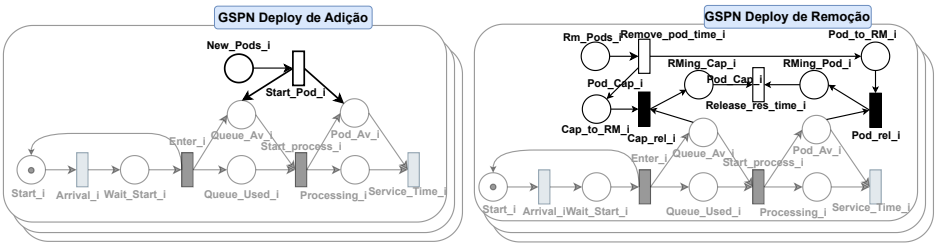

Figura 3. SPNs de deployment

4. Validação do Modelo SPN Comparando com Experimentos Reais

Esta seção apresenta a validação do modelo SPN por meio da comparação entre as métricas de desempenho estimadas pelo modelo e aquelas observadas em um ambiente real. O objetivo da validação é verificar a capacidade do modelo em representar adequadamente o comportamento do sistema diante de variações de carga, considerando especificamente as métricas de vazão e tempo de resposta. Os cenários avaliados foram definidos com foco em analisar o comportamento do *autoscaling* diante da variação de carga e em verificar a capacidade do gêmeo digital de apoiar a recomendação de réplicas orientada por SLA. Assim, os resultados devem ser interpretados de acordo com esse objetivo experimental. O ambiente experimental foi implantado na Amazon Web Services (AWS) utilizando o AWS Cloud Development Kit (CDK) [AWS 2025], o que permitiu automatizar o provisionamento da infraestrutura, padronizar a configuração dos experimentos e favorecer a reprodutibilidade dos resultados. Além disso, o processo de validação foi disponibilizado para permitir o uso e a reprodução dos experimentos que sustentam os resultados apresentados nesta seção. Foram criadas duas Virtual Private Clouds (VPCs) distintas: uma dedicada à geração de carga dos clientes e outra destinada à execução da infraestrutura Kubernetes que hospeda a aplicação avaliada. A infraestrutura do cluster Kubernetes foi composta por três *worker nodes* e um *control plane*, enquanto uma máquina

virtual adicional foi utilizada exclusivamente para a geração de carga. A aplicação analisada era formada por dois *deployments*, cada um inicialmente configurado com duas réplicas. O tempo médio de serviço dos *pods* foi definido como 0,2 segundos, seguindo uma distribuição exponencial.

A geração de carga foi realizada por meio de scripts desenvolvidos com o Locust [Locust 2015], configurados para produzir uma carga crescente e oscilante, variando de 0 até 120 requisições por segundo ao longo do experimento. A Figura 4 apresenta a comparação entre a vazão observada no sistema real e a vazão estimada pelo modelo. Observa-se uma forte sobreposição entre os valores obtidos experimentalmente e aqueles produzidos pelo modelo, indicando uma boa aderência entre ambos. Essa observação foi confirmada estatisticamente por meio de um teste *t* pareado com nível de confiança de 95%, que resultou em um valor de $p = 0{,}760061$. Esse resultado indica a ausência de evidência estatística de diferença entre as médias de vazão do modelo e do sistema real.

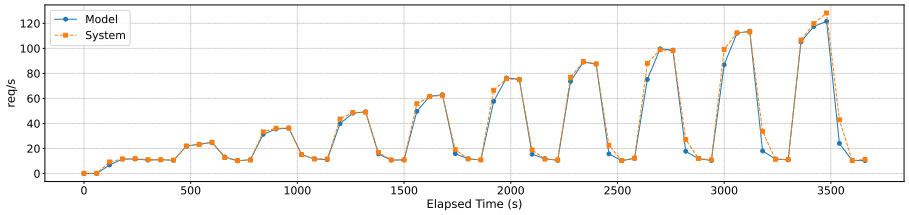

Figura 4. Validação da vazão (TP)

A validação do tempo de resposta apresenta comportamento consistente com os resultados observados para a vazão, conforme ilustrado na Figura 5. Durante a maior parte do experimento, especificamente até aproximadamente 2900 segundos, o tempo de resposta manteve-se estável, indicando que a capacidade dos *pods* foi suficiente para atender às requisições sem formação significativa de filas. A partir desse ponto, o aumento da carga resultou no início do enfileiramento e, consequentemente, no crescimento do tempo de resposta no sistema real. Esse comportamento também foi adequadamente capturado pelo modelo SPN, que reproduziu a transição entre o regime sem contenção e o regime com enfileiramento. O teste *t* aplicado aos valores de tempo de resposta, considerando novamente um nível de confiança de 95%, produziu um valor de $p = 0{,}942061$, reforçando a equivalência estatística entre os resultados do modelo e do sistema real.

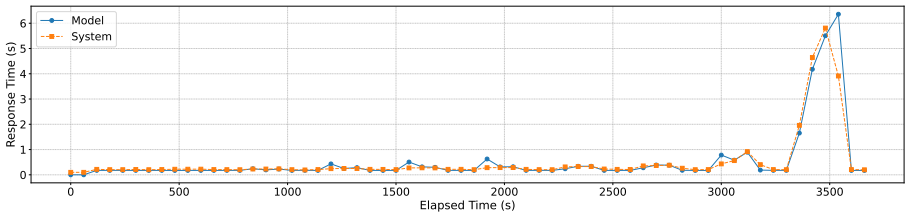

Figura 5. Validação do tempo de resposta (RT)

Os resultados obtidos demonstram que o modelo SPN é capaz de representar de forma consistente o impacto das variações de carga sobre as métricas de desempenho analisadas. Dessa forma, o modelo pode ser utilizado com confiança para realizar inferências sobre o comportamento do sistema sob diferentes condições operacionais, servindo como base para a avaliação de cenários *what-if* e para a análise de configurações alternativas com potencial de melhorar o atendimento aos SLAs.

5. Estudos de Caso

5.1. Estudo de Caso I

Este estudo de caso avalia o uso do Gêmeo Digital para controlar dinamicamente o aumento e a redução da capacidade dos *deployments* em um cenário caracterizado por variações acentuadas da carga de trabalho. O sistema foi submetido a intervalos entre chegadas variando de 0,2 até 0,018 segundos, representando um ambiente com flutuações intensas na demanda. A infraestrutura experimental foi implantada por meio do AWS CDK e composta por um *control plane*, sete *worker nodes* e dois *deployments*, cujos tempos médios de processamento são de 0,2 e 0,18 segundos. Inicialmente, cada *deployment* foi configurado com apenas uma réplica. O SLA considerado no experimento foi definido como um tempo máximo de resposta de 2 segundos, valor que contempla tanto resultados empíricos sobre tempos de espera aceitáveis [Nah 2004] quanto recomendações amplamente utilizadas na indústria [LLC 2025]. O Módulo de Execução foi configurado para utilizar um único contêiner, de forma a limitar o consumo de recursos destinados à execução dos modelos. Para fins de comparação, os resultados obtidos com o GD foram confrontados com o mecanismo nativo de escalonamento do Kubernetes, o Horizontal Pod Autoscaler (HPA). Foram avaliadas três configurações distintas do HPA, utilizando limiares de utilização de CPU de 50%, 70% e 90%. Nessas configurações, o escalonamento é acionado quando a utilização média de CPU ultrapassa o limiar definido, sendo os demais parâmetros mantidos com os valores padrão do cluster.

A Figura 6 apresenta os resultados de vazão para todas as configurações analisadas. Observa-se que a vazão acompanha diretamente a variação da carga imposta ao sistema, sem diferenças significativas entre as abordagens avaliadas, indicando que todas foram capazes de processar as requisições sem perdas. Em contraste, o comportamento do tempo de resposta apresentou diferenças expressivas entre as configurações, conforme ilustrado na Figura 7. Configurações com limiares mais elevados no HPA resultaram em tempos de resposta superiores, especialmente nos casos de 70% e 90%. Esse comportamento decorre do atraso na alocação de novas réplicas, uma vez que essas configurações requerem níveis mais altos de utilização para acionar o escalonamento, tornando o sistema menos responsivo a aumentos súbitos de carga. Nesse cenário, apenas a configuração do HPA com limiar de 50% e a abordagem baseada em Gêmeo Digital foram capazes de atender consistentemente ao SLA estabelecido. A Figura 8 apresenta a média temporal

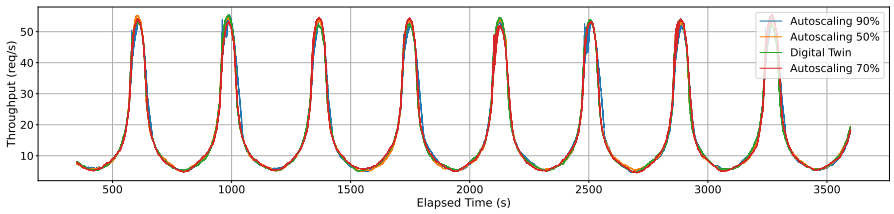

Figura 6. TP - Caso de Uso 01

do número total de réplicas ao longo do experimento, considerando todos os *deployments* do cluster. Assim, os valores fracionários reportados correspondem à média do total de réplicas observadas durante a execução, e não a quantidades instantâneas de *pods*. As configurações do HPA com limiares de 70% e 90% apresentaram os menores valores médios, com 5,93 e 8,97 réplicas, respectivamente; contudo, essas configurações não

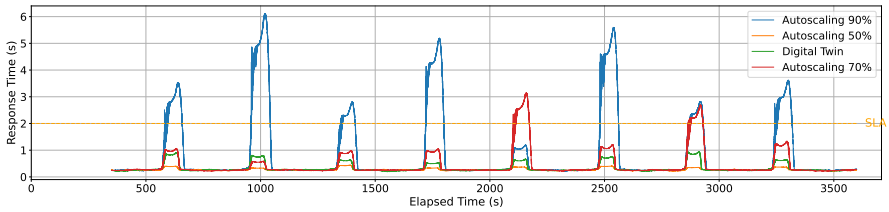

Figura 7. RT - Caso de Uso 01

atenderam ao SLA de tempo de resposta. A configuração com HPA de 50%, que cumpriu o SLA, utilizou em média 16,26 réplicas. A abordagem baseada em Gêmeo Digital utilizou, em média, 10,93 réplicas, considerando também a réplica destinada à execução do próprio Gêmeo Digital. Esse resultado representa uma redução de 32,83% no número médio de réplicas em relação ao HPA de 50%, o que também se refletiu em uma redução de 7,39% no consumo médio de CPU.

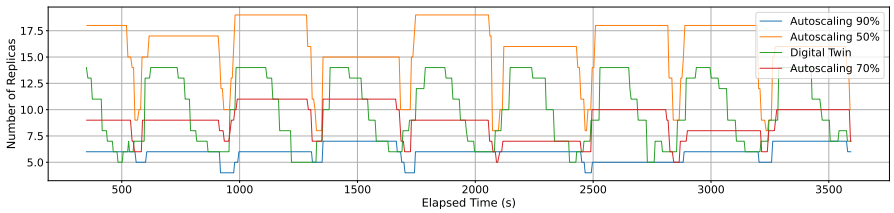

Figura 8. Número total de réplicas - Caso de Uso 01

5.2. Estudo de Caso II

O segundo estudo de caso avalia o comportamento do Gêmeo Digital em um cenário com carga aleatória, no qual os intervalos entre requisições variam de forma estocástica entre 0,1 e 0,006 segundos. A mesma sequência de carga foi aplicada em todas as configurações analisadas, garantindo condições equivalentes de comparação. Diferentemente do caso anterior, esse cenário apresenta menor amplitude de variação, uma vez que nem toda mudança de carga resulta em valores extremos. Esse tipo de carga enfatiza situações em que o principal desafio é manter o atendimento ao SLA com o menor consumo possível de recursos. Assim como no estudo anterior, a abordagem baseada em Gêmeo Digital foi comparada às configurações do HPA com limiares de 50%, 70% e 90%. A infraestrutura utilizada manteve sete *worker nodes*, um *control plane* e uma máquina virtual dedicada à geração de carga. A aplicação foi configurada com *deployments* cujos tempos médios de processamento foram definidos como 0,18 e 0,2 segundos.

A Figura 9 mostra que, com exceção da configuração do HPA com limiar de 90% durante um curto intervalo, as abordagens apresentam comportamentos semelhantes em termos de vazão. A pequena degradação observada nessa configuração é resultado da incapacidade de reagir rapidamente a variações abruptas de carga, ocasionando atrasos no processamento de requisições. O impacto dessas variações é mais evidente no tempo de resposta, apresentado na Figura 10. A configuração do HPA com limiar de 90% não foi capaz de atender ao SLA, com tempos de resposta atingindo valores próximos a 8 segundos. Para facilitar a análise das demais configurações, os resultados são apresentados com ampliação do eixo vertical, evidenciando diferenças sutis entre elas. A abordagem

baseada em Gêmeo Digital apresentou um aumento controlado do tempo de resposta durante os períodos de pico, atingindo aproximadamente 0,6 segundos, valor ainda inferior ao limite estabelecido pelo SLA. Esse comportamento reflete a estratégia de operação do Gêmeo Digital, que prioriza a economia de recursos mantendo apenas a capacidade necessária para atender aos requisitos de desempenho. A Figura 11 apresenta a média

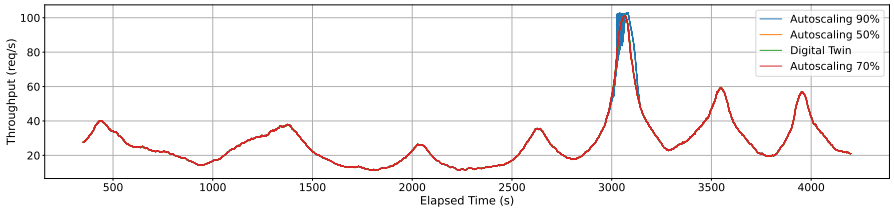

Figura 9. TP - Caso de Uso 02

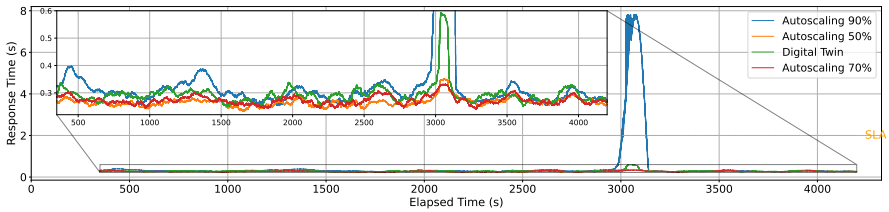

Figura 10. RT - Caso de Uso 02

temporal do número total de réplicas ao longo do experimento. As configurações do HPA com limiares de 90%, 70% e 50% utilizaram, em média, 5,3, 10,57 e 15,54 réplicas, respectivamente. A abordagem baseada em Gêmeo Digital utilizou, em média, 8,59 réplicas, considerando também o contêiner responsável pela execução do Módulo de Execução. Nesse cenário, o Gêmeo Digital apresentou uma redução de 18,7% no número médio de réplicas em relação à melhor configuração baseada em HPA que atendeu ao SLA, além de uma redução de 7,15% no consumo médio de CPU.

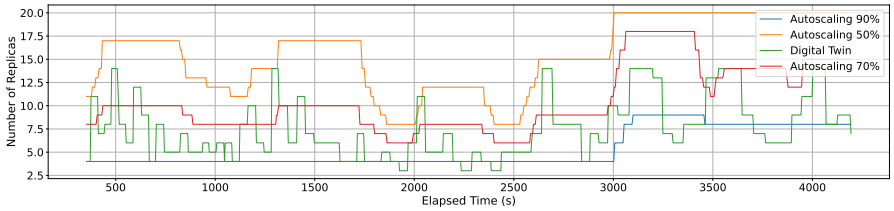

Figura 11. Número total de réplicas - Caso de Uso 02

6. Conclusão

Este trabalho apresentou um Gêmeo Digital para Kubernetes voltado à recomendação de configurações de réplicas orientadas por SLA de tempo de resposta, usando modelos SPN para avaliar cenários *what-if*. A solução combina sincronização do estado a partir de métricas observadas, projeção conservadora da carga no horizonte de decisão e uma busca que reduz o número de execuções necessárias para explorar o espaço de configurações

antes da aplicação no ambiente real. A fidelidade do modelo foi verificada experimentalmente em um ambiente na nuvem, com equivalência estatística entre valores observados e estimados para vazão e tempo de resposta, sustentando seu uso para inferência e comparação entre alternativas de escalonamento. Nos estudos de caso, a abordagem manteve o SLA e reduziu o número médio de réplicas em dois perfis de carga quando comparada às melhores configurações do HPA: 32,83% no cenário de alta variação e 18,7% no cenário com carga aleatória, com redução adicional no consumo médio de CPU em ambos. A comparação com o HPA mostrou que políticas baseadas em limiares fixos são sensíveis ao perfil de carga: um mesmo limiar pode manter o SLA em determinado padrão de demanda e falhar quando a carga se torna mais variável ou sofre picos, o que impõe ajuste manual por *deployment*. Na abordagem proposta, a recomendação é obtida pela avaliação de cenários *what-if* sob a carga considerada em cada ciclo, selecionando a configuração que atende ao SLA e reduz recursos, sem depender de percentuais de utilização que nem sempre refletem diretamente tempo de resposta e vazão. Por ser baseada em SPN, a abordagem é extensível: o mesmo fluxo de atualização parametrizada, execução *what-if* e seleção por restrições pode ser adaptado a outras métricas e objetivos (por exemplo, consumo elétrico). Como continuidade, pretendemos ampliar os cenários considerados e expandir a avaliação comparativa com abordagens adaptativas e preditivas de *autoscaling*, incluindo mecanismos como VPA e KEDA, de modo a aprofundar o posicionamento experimental da proposta frente ao estado da arte.

Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq, por meio do INCT ICoNIoT, processo nº 383355/2025-7.

Referências

Abdelrahman, M., Macatulad, E., Lei, B., Quintana, M., Miller, C., and Biljecki, F. (2025). What is a digital twin anyway? deriving the definition for the built environment from over 15,000 scientific publications. *Building and Environment*, 274:112748.

Ahmad, H., Treude, C., Wagner, M., and Szabo, C. (2024). Smart hpa: A resource-efficient horizontal pod auto-scaler for microservice architectures. In *2024 IEEE 21st International Conference on Software Architecture (ICSA)*, pages 1–12.

AWS, t. (2025). AWS instance types. `https://aws.amazon.com/pt/ec2/instance-types/`. Accessed: 2025-05-02.

Bittencourt, L. F., Braghetto, K. R., Cordeiro, D., and Sakellariou, R. (2024). On digital twins for cloud continuum applications. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 286–293. Springer.

Borsatti, D., Zaccarini, M., Matteucci, M., et al. (2024). Kubetwin: A digital twin framework for kubernetes deployments at scale. *IEEE Transactions on Network and Service Management*, 21(4):3889–3903.

da Silva, L. M. D., Alves, P. V. A., Silva, S. N., and Fernandes, M. A. C. (2025). Adaptive horizontal scaling in kubernetes clusters with ann-based load forecasting. *Cluster Computing*.

DeGlopper, D. R. (1992). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modeling. by raj jain. new

york: John wiley and sons, 1991. pp. 720. (hardcover). *International Journal of Legal Information*, 20(1):63–64.

Fé, I., Nguyen, T. A., Choi, E., Min, D., Lee, J.-W., Barbosa, V., Soares, A., Rego, P. A., Mei, A., and Silva, F. A. (2025). Energy-efficient performance optimization in kubernetes microservices using generalized stochastic petri net. *Journal of Network and Computer Applications*, page 104287.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Lakshan, S. and Hussain, S. (2025). A review of ai-driven techniques for cost optimization in kubernetes environments. In *2025 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pages 1–5. IEEE.

LLC, G. (2025). Como entender o core web vitals e os resultados da pesquisa google. Última atualização: 4 de agosto de 2025.

Locust, t. (2015). Locust an open source load testing tool. `https://locust.io/`. Accessed: 2025-05-02.

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

Maciel, P. R. M. (2023b). *Performance, reliability, and availability evaluation of computational systems, volume I: performance and background*. Chapman and Hall/CRC.

Nah, F. F.-H. (2004). A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163.

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*, pages 16–25. IOS Press.

Pozdniakova, O., Mažeika, D., and Cholomskis, A. (2024). Sla-adaptive threshold adjustment for a kubernetes horizontal pod autoscaler. *Electronics*, 13(7):1242.

Schlegel, B., Gemulla, R., and Lehner, W. (2009). K-ary search on modern processors. In *Proceedings of the Fifth International Workshop on Data Management on New Hardware*, pages 52–60.

Vasumathi, M., Sadasivan, M., Kumar, V. V. N. P., Kumar, B. K., et al. (2025a). Ai-driven predictive auto-scaling for efficient microservices deployment in cloud data centers using lstm and kubernetes hpa. In *2025 5th International Conference on Expert Clouds and Applications (ICOECA)*, pages 859–864. IEEE.

Vasumathi, M. T., Sadasivan, M., Asha, V., Phanindra Kumar, V. V. N., Kumar, B. K., and Veeresh (2025b). Ai-driven predictive auto-scaling for efficient microservices deployment in cloud data centers using lstm and kubernetes hpa. In *2025 5th International Conference on Expert Clouds and Applications (ICOECA)*, pages 1–6.

Veeck, C. H., Barbosa, M., and Dias, K. L. (2025). Reagir ou antecipar? uma comparação entre hpa e ml para balanceamento de carga. In *XLIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*.