Avaliação de Disponibilidade de uma Arquitetura de Computação de Borda com Redes de Petri Estocásticas

Carlos Brito[±], Laécio Rodrigues[±], Iure Fé[*], Guto Leoni Santos[⊗], Francisco Airton Silva[±]

± Universidade Federal do Piauí (UFPI) - Picos, PI, Brasil
∗ 3º BEC - Exército Brasileiro - Picos, PI, Brasil
⊗ Universidade Federal de Pernambuco (UFPE) - Recife, PE, Brasil

{carlosvictor,laecio8andrade,faps}@ufpi.edu.br, gls4@cin.ufpe.br

Abstract. *Mobile Edge Computing (MEC) emerged as an alternative to reduce network latency by bringing the data processing close to the users. This proximity requires that the service is available most of the time. Trying to assess the availability of such systems in real contexts requires high costs. This paper uses Stochastic Petri Nets (SPNs) to assess the availability of an MEC architecture, seeking to avoid premature investment in real equipment. This work has the potential to assist administrators in redefining the most optimized MEC architectures by decreasing the risk of failure.*

Resumo. *Mobile Edge Computing (MEC) surgiu como uma alternativa para reduzir a latência da rede que levou o processamento do fluxo de dados para mais perto dos usuários. Essa proximidade do usuário exige que o serviço esteja disponível o maior período de tempo possível. Avaliar a disponibilidade de tais sistemas em contextos reais demanda altos custos. Este artigo utiliza Redes de Petri Estocásticas (SPNs) para avaliar a disponibilidade de uma arquitetura MEC, buscando evitar o investimento prematuro em equipamentos reais. Este trabalho tem o potencial de auxiliar os administradores de rede a planejar arquiteturas MEC mais otimizadas diminuindo o risco de falhas.*

1. Introdução

Smartphones tornaram-se ao longo do tempo uma parte essencial na vida das pessoas. Esses aparelhos evoluíram a ponto de realizar tarefas que há uma década eram impensáveis. Atualmente existem várias aplicações móveis que permitem facilidade de comunicação e entretenimento. Existem também periféricos como pulseiras eletrônicas que permitem monitorar a saúde do usuário. Porém, muitas aplicações exigem alto poder de processamento que apenas servidores podem oferecer, tais como: jogos eletrônicos [Zafari et al. 2020] ou healthcare [Wan et al. 2020].

A *Mobile Cloud Computing* (MCC) é uma arquitetura computacional que surgiu para estender a capacidade dos dispositivos móveis. A MCC foi formada pela junção da computação móvel com a computação em nuvem tradicional. A junção dessas tecnologias proporciona uma infinidade de recursos e serviços, porém o tempo de resposta pode ser alto dependendo da localização dos recursos da nuvem. A *Mobile Edge Computing* (MEC) [Guo et al. 2020] surgiu posteriormente para mitigar alguns dos problemas

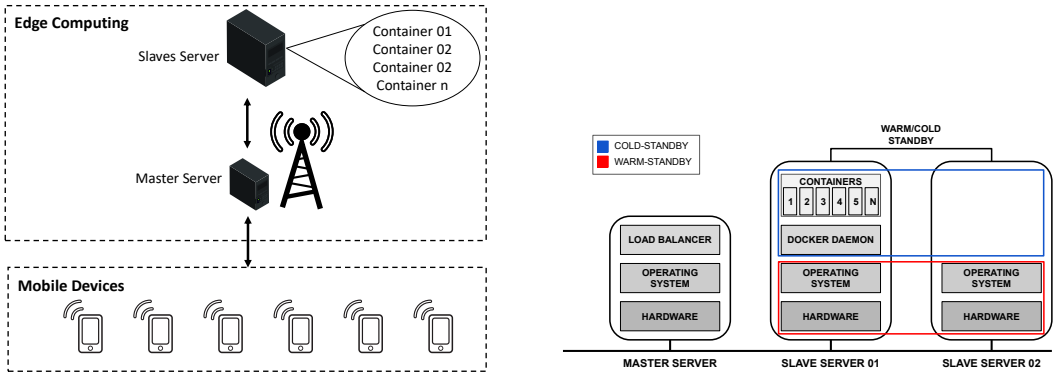
da computação em nuvem. A MEC foi idealizada para prover serviços em tempo real, e portanto, robustez e disponibilidade da arquitetura são requisitos essenciais. Todos os componentes de um sistema são propensos a falhas, e falhas devem ser reparadas o mais rápido possível a fim de evitar grandes perdas para os usuários. Avaliar a disponibilidade de uma arquitetura MEC pode ter alto custo considerando experimentos práticos. Assim, se faz necessário avaliar arquiteturas MEC antes mesmo de uma implantação real.

Redes de Petri Estocásticas (SPN) [Silva et al. 2015b] podem ser utilizadas para avaliar diversos tipos de sistemas complexos, incluindo uma arquitetura MEC. Alguns artigos relacionados abordaram modelos SPN para avaliar a disponibilidade de sistemas de computação móvel [Araujo et al. 2014], [Oliveira et al. 2013], [Santos et al. 2018]. No entanto, nosso trabalho explora a dependência dos componentes da arquitetura, adicionando a técnica de falha em cascata [Araujo et al. 2019], o que permite maior acuidade dos resultados. Também foi explorado a redundância de servidores almejando maximizar a disponibilidade do sistema.

As seções seguintes estão organizadas da seguinte forma: A Seção 2 apresenta a arquitetura utilizada neste artigo. A Seção 3 apresenta o modelo SPN gerado a partir da arquitetura, incluindo redundância. A Seção 4 apresenta os resultados da análise estacionaria do modelo, que são os índices de disponibilidade e *downtime* do sistema. Por fim, a Seção 5 traça algumas conclusões e futuros trabalhos.

2. Arquitetura

A Figura 1(a) apresenta a arquitetura base utilizada neste artigo que é uma arquitetura adotada em trabalhos anteriores com outros objetivos [Trinh and Yao 2017]. Na parte inferior temos os dispositivos móveis que irão gerar os dados por meio de aplicações. Os dados são enviados para uma camada MEC superior, responsável por processar e retornar informações aos dispositivos móveis. Neste trabalho damos foco à disponibilidade do lado dos servidores MEC. Consideramos uma arquitetura em *cluster* onde um módulo *master* reside em um primeiro servidor e nós *slaves* residem em um segundo servidor. Os nós *slaves* são microsserviços executados em contêineres. Neste trabalho cada contêiner é configurado para executar em um núcleo do servidor para garantir um paralelismo efetivo dos recursos.



(a) Arquitetura Base (b) Arquitetura em Camadas

Figura 1. Proposta de arquitetura base

Uma grande limitação da proposta de arquitetura base é que caso um dos dois servidores fiquem inativos todo o sistema deixará de funcionar. Por isso, a Figura 1(b) apresenta uma proposta de arquitetura com redundância.Nessa ideia têm-se o *slave server 01* e o *slave server 02*. Ambos servidores se mantêm sempre ligados, porém, o *docker* e os contêineres no *slave server 02* são instanciados apenas se o *slave server 01* cair. Sendo assim, tem-se um mecanismo de redundância de *warm standby* [Yuan and Meng 2011] no *hardware* e sistema operacional e *cold standby* [Briš 2013] no *daemon docker* e *containers*.

3. Modelo SPN

A Figura 2 apresenta um modelo SPN estendido para a arquitetura MEC, com 2 *slave servers* (em *warm-standby*) e um *master server*. Como falado anteriormente, esse novo modelo apresenta um mecanismo híbrido entre *warm-standby* (componentes HW e OS) e *cold-standby* (componentes DD e CTS). Para satisfazer as condições do *warm-standby* foram adicionadas duas transições, `HWSF_MTTF` e `OSSF_MTTF` (no bloco do *slave server 02*), que representam o tempo de falha do HW e OS quando não estão executando o Docker, ou seja, ociosos.

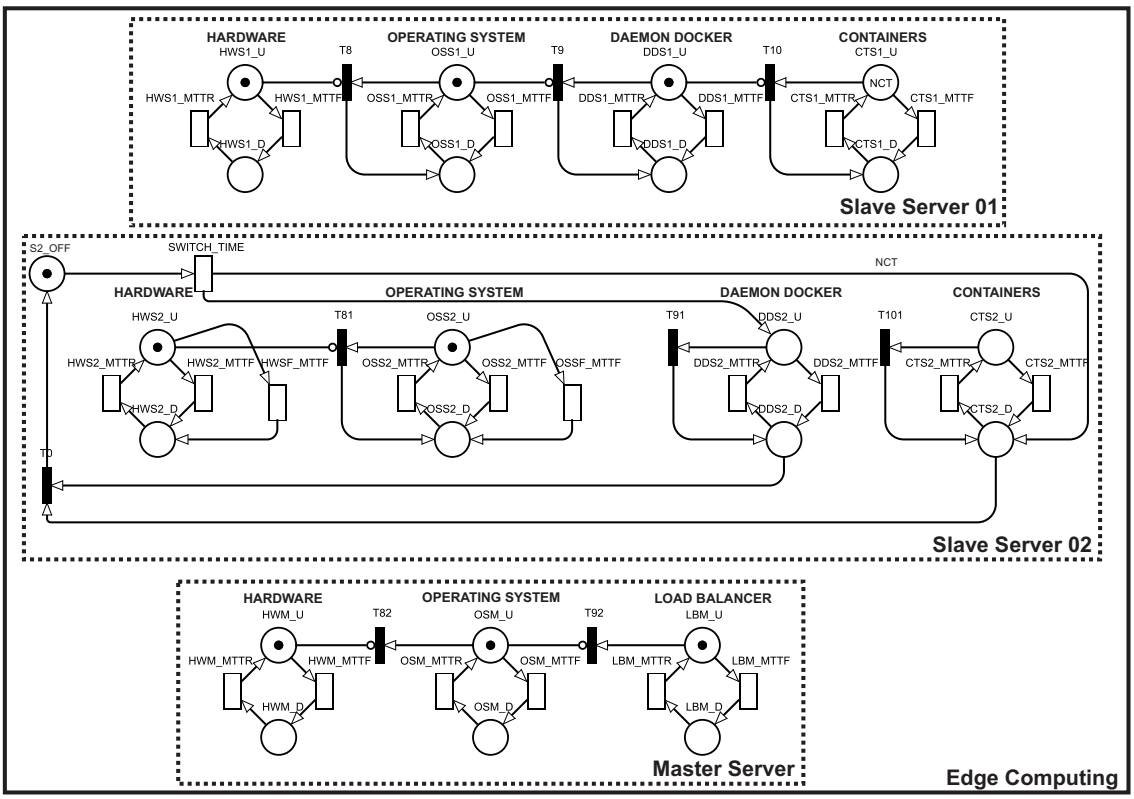


Figura 2. Modelo SPN estendido para uma arquitetura MEC

Considerando a dependência dos componentes o sistema estará funcionando simplesmente se todos os contêineres e o balanceador de carga estão funcionando ao mesmo tempo. O `NCT` é o parâmetro que representa o número máximo de contêineres que o sistema pode executar, somando o número de contêineres ativos nos dois servidores. O disparo do `SWITCH_TIME` ocorre quando o *slave server 01* cai e faz colocar o número de contêineres correspondente a `NCT` no lugar `CTS2_D`, ou seja, os

contêineres são criados com status inativos no *slave server 02* e levarão um tempo de CTS2_MTTR para serem instanciados. Portanto, a disponibilidade do modelo é dada por $A = P\{((\#CTS1\_U + \#CTS2\_U) = NCT)AND(\#LBM\_U > 0)\}$. A métrica de tempo de inatividade também foi aplicada para obtenção de resultados. O *downtime (D)* pode ser obtido por $D = (1 - A) \times 8760$, onde A representa a disponibilidade do sistema, e 8760 a quantidade de horas no ano. Vale ressaltar que as transições de MTTF são do tipo *infinite server* (paralelo) e as transições MTTR são do tipo *single server* (concorrente).

A Tabela 1 apresenta as condições de guarda utilizadas para o funcionamento do sistema no modelo.

Tabela 1. Condições de guarda utilizadas

Transição	Expressão	Descrição
OSS2_MTTR	#HWS2_U >0	Ativado quando HW está ativo.
DDS2_MTTR	#OSS2_U >0	Ativado quando OS está ativo.
CTS2_MTTR	(#DDS2_U >0) AND (#CTS2_U <#CTS1_D)	Ativado quando Docker do *slave server 02* está up e *slave server 01* possui mais contêineres desativados do que contêineres ativados no *slave server 02*.
HWSF_MTTF	#S2_OFF=1	Ativado quando o *slave server 02* está ocioso.
OSSF_MTTF	#S2_OFF=1	Ativado quando o *slave server 02* está ocioso.
HWS2_MTTF	#S2_OFF=0	Ativado quando o *slave server 02* está ativo.
OSS2_MTTF	#S2_OFF=0	Ativado quando o *slave server 02* está ativo.
T91	((#CTS1_U = NCT) OR (#OSS2_U = 0))	Ativado quando o *slave server 01* está ativo ou quando componente dependente (OS) está inativo.
T101	((#CTS1_U = NCT) OR (#DDS2_U = 0) OR (#CTS2_U >#CTS1_D))	Ativado quando o *slave server 01* está ativo ou o Docker caiu, ou o número de contêineres ativo do server 01 se tornou maior do que no server 02.
T0	#CTS1_U = NCT	Ativado quando o slave server 01 estiver ativo.
SWITCH_TIME	#CTS1_D >0	Ativado quando o slave server 01 cair.

4. Estudo de Caso

Esta seção apresenta um estudo de caso com uma avaliação de disponibilidade considerando o modelo apresentado. Os valores do MTTF e MTTR de cada componente foram extraídas de [Araujo et al. 2019] e [da Silva Lisboa et al. 2018]. A Tabela 2 apresenta os valores de entrada dos componentes dos modelos. A nível de simplificação, optamos por utilizar a mesma configuração entre os três servidores. O tempo para acionar o servidor redundante na transição SWITCH_TIME foi de 0,0833333 horas, extraído de [Araujo et al. 2019].

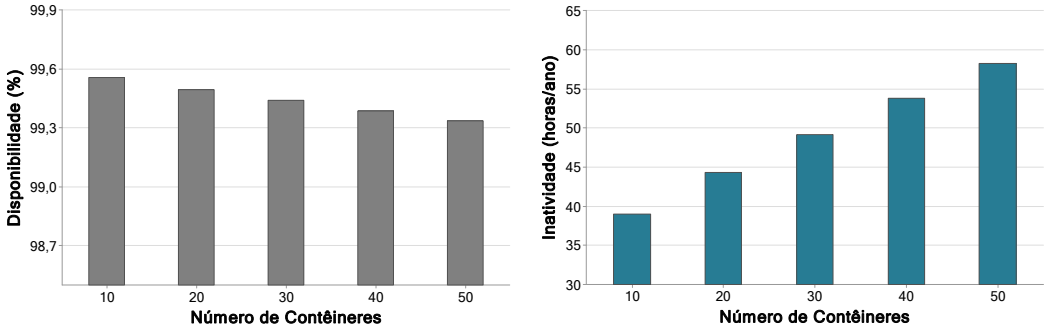
4.1. Análise de Disponibilidade

Essa seção apresenta a análise de disponibilidade. Foram definidos cinco cenários, variando o número de contêineres disponíveis (10, 20, 30, 40 e 50 contêineres). Estes cenários foram gerados a fim de observar a variação da disponibilidades do modelo bem como o impacto que o número de contêineres gera na arquitetura.

Tabela 2. Valores de entrada do modelo

Componente	MTTF (horas)	MTTR (horas)
Hardware	8760	8
Operating System	2800	1
Docker Daemon	2516	0,255
Container	1258	0,238
Load Balancer	700	1
Hardware (Ocioso)	17520	8
Operating System (Ocioso)	5600	1

A Figura 3 mostra a disponibilidade e tempo de inatividade calculadas por análise estacionária com a ferramenta Mercury [Silva et al. 2015a]. A variação da disponibilidade apresentada na Figura 3(a) mostra que a disponibilidade cai conforme são adicionados contêineres ao sistema. A variação é entre 99,6 e 99,3 por cento, ou seja, se manteve acima de 99% em todos os cenários. A Figura 3(b) apresenta o tempo de inatividade em horas por ano. O modelo varia de 40 a 60 h/a. O modelo manteve-se abaixo de sessenta em todos os cenários. A taxa de aumento de inatividade cresce bem pouco e perceptivelmente a adição de novos contêineres não irá causar tanto impacto na inatividade. O mecanismo de redundância é o que justifica esse baixo aumento. Esses resultados são esperados uma vez que vários contêineres podem falhar ao mesmo tempo, porém o reparo é feito em um por vez. Logo, quanto mais contêineres houver, mais falhas são prováveis de ocorrer e mais tempo será necessário para reparar um a um. Porém, no modelo proposto acontece de maneira bem sutil, isso porque há o mecanismo de redundância no servidor.



(a) Disponibilidade (b) Tempo de Inatividade

Figura 3. Disponibilidade e tempo de inatividade da arquitetura variando número de contêineres

5. Conclusão

Este artigo propôs um modelo de SPN para representar e avaliar a disponibilidade de uma arquitetura básica do MEC. Cenários com diferentes quantidades de contêineres foram analisados, concluindo-se que a arquitetura apresentou bons resultados de disponibilidade mesmo com o aumento do número contêineres. Por fim, o modelo pode se tornar uma ferramenta para possiveis administradores da área no futuro. Como trabalho futuro pretende-se acrescentar redundância nos componentes do *master server* e também tentar explorar outros cenários para teste dos modelos.

Referências

Araujo, E., Dantas, J., Matos, R., Pereira, P., and Maciel, P. (2019). Dependability evaluation of an iot system: A hierarchical modelling approach. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2121–2126. IEEE.

Araujo, J., Silva, B., Oliveira, D., and Maciel, P. (2014). Dependability evaluation of a mhealth system using a mobile cloud infrastructure. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1348–1353. IEEE.

Briš, R. (2013). Evaluation of the production availability of an offshore installation by stochastic petri nets modeling. In *The International Conference on Digital Technologies 2013*, pages 147–155. IEEE.

da Silva Lisboa, M. F. F., Santos, G. L., Lynn, T., Sadok, D., Kelner, J., Endo, P. T., et al. (2018). Modeling the availability of an e-health system integrated with edge, fog and cloud infrastructures. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00416–00421. IEEE.

Guo, J., Luo, W., Song, B., Yu, F. R., and Du, X. (2020). Intelligence-sharing vehicular networks with mobile edge computing and spatiotemporal knowledge transfer. *IEEE Network*.

Oliveira, D., Araujo, J., Matos, R., and Maciel, P. (2013). Availability and energy consumption analysis of mobile cloud environments. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4086–4091. IEEE.

Santos, G. L., Endo, P. T., da Silva Lisboa, M. F. F., da Silva, L. G. F., Sadok, D., Kelner, J., Lynn, T., et al. (2018). Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. *Journal of Cloud Computing*, 7(1):16.

Silva, B., Matos, R., Callou, G., Figueiredo, J., Oliveira, D., Ferreira, J., Dantas, J., Lobo, A., Alves, V., and Maciel, P. (2015a). Mercury: An integrated environment for performance and dependability evaluation of general systems. In *Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN*.

Silva, F. A., Rodrigues, M., Maciel, P., Kosta, S., and Mei, A. (2015b). Planning mobile cloud infrastructures using stochastic petri nets and graphic processing units. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 471–474. IEEE.

Trinh, C. and Yao, L. (2017). Energy-aware mobile edge computing for low-latency visual data processing. pages 128–133.

Wan, S., Gu, Z., and Ni, Q. (2020). Cognitive computing and wireless communications on the edge for healthcare service robots. *Computer Communications*, 149:99–106.

Yuan, L. and Meng, X.-Y. (2011). Reliability analysis of a warm standby repairable system with priority in use. *Applied Mathematical Modelling*, 35(9):4295–4303.

Zafari, F., Leung, K. K., Towsley, D., Basu, P., Swami, A., and Li, J. (2020). Let's share: A game-theoretic framework for resource sharing in mobile edge clouds. *arXiv preprint arXiv:2001.00567*.