GLOBECOM 2024 - 2024 IEEE Global Communications Conference | 979-8-3503-5125-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/GLOBECOM52923.2024.10901098

Resilient and Efficient Microservices: Stochastic Modeling and Quantification of Energy Consumption and Recovery Times

Iure Fé, Luis Guilherme Silva, André Soares, and Francisco Airton Silva
Federal University of Piauí (UFPI)
Piauí, Brazil
{iure.fe,luis.e,andre.soares,faps}@ufpi.edu.br

Alessandro Mei,
Sapienza University of Rome
Roma, Italy
mei@di.uniroma1.it

Paulo A. L. Rego,
Federal University of Ceará (UFC)
Ceará, Brazil
paulo@dc.ufc.br

Eunmi Choi
Kookmin University
Seoul, South Korea,
emchoi@kookmin.ac.kr

Tuan Anh Nguyen, *IEEE Member* ®,
Jae Woo Lee, Dugki Min®
Konkuk University
Seoul, South Korea,
{anhnt2407, jwlee, dkmin}@konkuk.ac.kr

***Abstract*—As the adoption of microservices architectures in cloud deployments grows, so does the challenge of ensuring rapid recovery and minimal energy consumption in the face of disasters. This paper introduces a Generalized Stochastic Petri Net (GSPN) model specifically designed to quantify recovery times and electrical consumption in such environments. The model supports the development of resilient and eco-conscious systems by allowing precise manipulation and planning based on various configuration scenarios. We identify critical architectural elements and define intervals that yield significant improvements. Our findings not only enhance the understanding of energy and recovery dynamics in microservices but also serve as a crucial tool for system designers aiming to optimize both performance and sustainability. The implications of this research facilitate a strategic approach to disaster recovery planning, contributing to the broader field of cloud computing resilience.**

***Index Terms*—Microservices, Disaster recovery, Power Consumption, Recovery Time, Generalized Stochastic Petri Net, Cloud Computing**

I. Introduction

Cloud-native applications utilizing microservices architecture offer benefits such as simplified development and enhanced service distribution across data centers (DCs), improving performance and system availability [1]. These architectures potentially reduce energy consumption, significant since global energy use by information and communications technology is expected to rise from 2% to 14% by 2040 [2, 3]. Containerization facilitates this by enabling scalable architectures that quickly adjust capacity, thus minimizing unnecessary energy use. However, microservices are vulnerable to disasters, including cyber-attacks and natural events [4], making disaster recovery (DR) essential for maintaining operations. DR strategies often involve resource redundancy, impacting electrical consumption [5]. Balancing recovery and consumption involves managing various parameters and testing different infrastructure scenarios [6]. Real-world testing of DR strategies can be costly and impractical. Existing research provides strategies for managing disasters and power in DCs, but lacks focus on microservices and their energy implications. This study proposes a model-based approach to optimize recovery times and reduce energy use in microservices environments, addressing this research gap.

GSPNs are essential for modeling systems with complex dynamics like concurrency and random timing. By extending basic Petri nets, GSPNs incorporate both timed and immediate transitions. This integration enables them to simulate the random delays and concurrent activities often seen in distributed systems effectively. The value of GSPNs stems from their dual capabilities: they provide a graphical representation, which simplifies the visualization and comprehension of system behaviors, and a strong mathematical basis, which ensures accurate modeling and analysis. These features make GSPNs indispensable tools in system performance evaluation and optimization [7].

Key contributions of this study are as follows. *(i) Development of a GSPN Model for Predictive Simulation:* This contribution aims to establish a robust, predictive modeling framework using GSPN to assess recovery times and energy consumption post-disaster in Kubernetes-based microservices systems. The model's purpose is to provide a predictive simulation tool that allows for the optimization of resilience and energy efficiency. *(ii) Execution of a Detailed Sensitivity Analysis to Highlight System Vulnerabilities:* The paper introduces

Corresponding authors: {anhnt2407, dkmin}@konkuk.ac.kr;
This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. 2020R1A6A1A03046811).; This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2021R1A2C2094943)

a comprehensive sensitivity analysis that identifies the system components most influential to recovery time and energy metrics. The purpose here is to spotlight vulnerabilities and critical points within the system architecture, facilitating targeted improvements and better resource allocation. *(iii) Formulation of Strategic Guidelines for Balancing Resilience and Energy Efficiency:* Leveraging the outcomes from the GSPN model and sensitivity analysis, this contribution seeks to formulate strategic guidelines. These guidelines are intended to aid system designers in architecting microservice infrastructures that efficiently balance resilience with energy consumption, fostering sustainable system design. *(iv) Quantitative Assessment of Component Impact on System Performance:* By quantifying how individual components of the system affect performance metrics, the paper provides empirical evidence supporting specific architectural and operational adjustments. This assessment serves the purpose of validating design choices and operational strategies to optimize both recovery time and energy use. *(v) Innovative Approaches to System Recovery and Energy Consumption Evaluation:* The paper introduces novel methodological approaches that combine theoretical models with practical evaluations, focusing on recovery and energy consumption in cloud environments. The purpose is to push the boundaries of current methodologies, offering new ways to evaluate and improve the resilience and eco-friendliness of cloud services.

These contributions collectively enrich the academic and practical realms by offering advanced tools and methodologies for designing and managing more resilient and energy-efficient cloud systems.The proposed model allows system designers to orchestrate their infrastructure to achieve the desired level of survivability while being quantitatively aware of the architecture's energy consumption. The organization of this paper is as follows. Section II discussed previous related works. Section III details the microservices architecture. Section IV presents the developed stochastic model. Section V shows the analysis results. The paper is concluded in section VI.

II. Related Work

This section reviews efforts aimed at shortening disaster recovery times and cutting energy use in cloud system planning. Andrade et al. [5] utilized a Stochastic Petri Net (SPN) model to evaluate disaster recovery in cloud systems based on virtual machines, focusing on metrics like response time and availability but not covering energy use, containers, or orchestrators. Nong et al. [8] developed a survivability framework in CloudSim for relocating virtual machines to operational racks during failures, where they are rebooted as containers, though it lacks consideration for microservices and energy impacts.

Bhavsar et al. [1] explored a Kubernetes-based disaster recovery using Amazon Web Services to automate recovery processes, without considering energy consumption or infrastructure scalability. Trivedi et al. [4] used a Stochastic Reward Net (SRN) and Continuous Time Markov Chain (CTMC) to model recovery capabilities post-disaster, capturing system performance effectively but not examining electrical consumption during disasters.

Hamadah [9] discussed rapid recovery options using cloud services with an emphasis on electrical consumption. Isa et al. [10] proposed a Mixed Integer Linear Programming (MILP) based energy-efficient fog computing infrastructure for health monitoring, assessing the energy use of components but not incorporating container technologies.

These studies underscore the value of disaster recovery mechanisms, typically involving backups or redundancy across data centers, yet they do not address all aspects mentioned in our work.

III. Microservices Architecture

Large systems utilizing microservices often deploy hundreds of containers managed by orchestrators like Kubernetes, a primary system used in this study [1]. Figure 1 illustrates the Kubernetes architecture, spanning multiple data centers (DCs) in a cloud setting. This architecture comprises two main node types: the Control Plane (CP) and Worker Nodes (WN), with CP managing the synchronization of WNs and Pods, which are units containing one or more containers that share networking and storage resources. Each Pod may host individual microservices or a set of replicated services, known as a deployment [11].

The distributed nature of this system, as shown in Figure 1, enables Pods to be spread across various DCs, enhancing application resilience by providing spatial diversification [12]. In the event of a DC failure, the number of active Pods drops, prompting Kubernetes to initiate Pod instantiation in operational DCs to mitigate this loss. This process involves container and application initialization, which is time-sensitive and varies based on disaster scope, instantiation duration, node count, and the number of Pods [11]. Misconfigurations can lead to recovery times exceeding acceptable thresholds.

To enhance availability, a K-out-of-N (KooN) strategy, where spare Pods and Nodes are pre-deployed, can be employed to decrease recovery times (RT) at the cost of increased energy consumption [13]. This approach is essential in maintaining system performance during unplanned disruptions.

IV. Model

This section outlines the model applied to depict the infrastructure of microservices orchestrated by Kubernetes, employing a GSPN. GSPN is chosen for its capability to simulate parallel, concurrent, asynchronous, deterministic, and non-deterministic events within systems [14]. Such features allow GSPN to accurately model the dynamic responses of systems to disasters, a utility demonstrated in prior studies [4–6]. The model, reflecting the architecture shown in Figure 1, is detailed in Figure 2 and Table I, with transitions classified as single-server (SS) and infinite-server (IS) firing semantics.

The CP block of the model illustrates the Control Plane's operation, delineating functional and non-functional states at

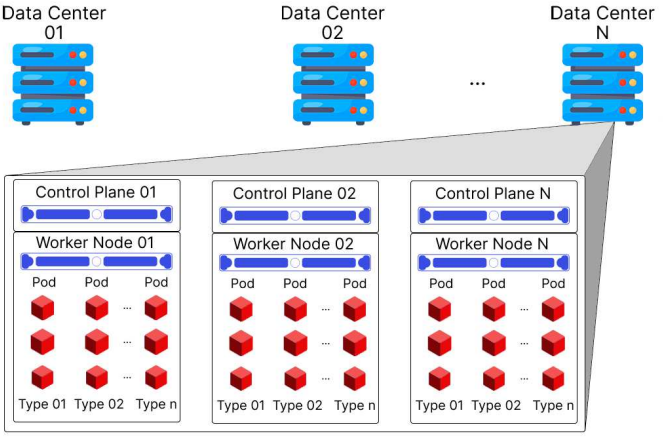


Fig. 1: Architecture of a Kubernetes cluster.

various nodes within data centers (DCs). It contains two primary places: CP_{UPn}, indicating operational Control Plane nodes, and CP_{DOWNn}, signifying non-operational elements in the n^{th} DC, with initial counts given by CP_{UPnVal}. Transitions CP_{FAILn} and CP_{REPn} represent the failure and subsequent repair of a Control Plane, respectively. An immediate transition, CP_{Rn}, is conditionally activated based on the guard expression [g1], which is used to manage the model's reaction to disasters. Specifically, if the model's aim is to analyze Recovery Time (RT) or post-disaster behavior, the expression $\#D_n\_Dis = 1$ triggers node removal in the event of a DC disaster, where $\#$ calculates token counts.

This disaster-specific sub-block, depicted in Figure 2, includes places D_n_ROP and D_n_Dis, with D_n_ROP indicating no disaster at DC n, and D_n_Dis activated upon disaster occurrence, initiating the $D_n\_Dis\_time$ transition at set times for detailed disaster impact analysis.

TABLE I: Description of model transitions.

Transition	Description	Firing	Weight	Priority
WN_{REPn}	Repair of a DC WN n	SS	-	1
WN_{FAILn}	Failure of a DC WN n	IS	-	1
WN_{Rn}	Removing a WN from DC n	-	1.0	3
CP_{FAILn}	Failure of a DC CP n	SS	-	1
CP_{REPn}	Repair of a DC CP n	-	1.0	3
CP_{Rn}	Removing a CP from DC n	SS	-	1
D_{nA}	DC capacity addition n	-	1.0	1
D_{nR}	Capacity removal on DC n	-	1.0	2
P_mD_{nCT}	Start creating a Pod m on DC n	-	1.0	1
P_mD_{nINST}	Time of a Pod m on DC n	IS	-	1
P_mD_{nFAIL}	Failure of a Pod m on DC n	IS	-	1
P_mD_{nREP}	Repair a pod type m in DC n	SS	-	1
P_mD_{nCTR}	Removal of Pods under creation m on DC n	-	1.0	2
P_mD_{nUPR}	Removal of a working Pod m on DC n	-	1.0	2
P_mD_{nFAILR}	Removing a failed Pod m on DC n	-	1.0	2
$D_n\_Dis\_time$	Sudden Disaster Event	SS	-	1

The WN block within the model details Worker Nodes, behaving like the Control Plane (CP) block. Transitions such as WN_{FAILn} and WN_{REn} depict the failure and subsequent recovery of Worker Nodes, while WN_{Rn} handles their removal in disaster scenarios, influenced by configuration [g2].

The Pods block represents Pods and the Data Center's (DC) capability to initialize them. Pod instantiation capacity per DC is shown by D_{nCAP} tokens, set initially by $D_{nCAPVal}$, calculated from the system's overall capacity minus the used capacity. This setup highlights the model's approach to managing resources and response to disruptions.

$$D_{nCAPVal} = dc\_n\_total\_cap - used\_cap\_dc\_n \tag{1}$$

In which,

$$dc\_n\_total\_cap = ppn\_dc\_n \times \#wndc\_n\_on \tag{2}$$

$$used\_cap\_dc\_n = \sum_{m \in U} (\#P_mD_{nCRT} + \#P_mD_{nUP} + \#P_mD_{nDOWN}), \tag{3}$$

The model delineates the capacity for Pod instantiation within each Data Center (DC) as the total capacity of the Worker Nodes (WNs) minus the already instantiated Pods, indicated by the variable set $U = \{m|m$ are the indices of Pods eligible for allocation in DC $n\}$. This setup ensures that the model comprehensively simulates the deployment of various Pod types across DCs. Each Pod type m within DC n is mapped to specific model locations—P_mD_{nUP}, P_mD_{nDOWN}, P_mD_{nCRT}—and their related transitions, which are tied to the DC capacity marker D_{nCAP}.

Active Pods are denoted by tokens at P_mD_{nUP}, with their count initialized by P_mD_{nUPVal}. Pod failures trigger the P_mD_{nFAIL} transition, moving a token to P_mD_{nDOWN}, while recovery is managed by P_mD_{nREP}.

Modifications in WN capacity within DCs affect Pod instantiation capabilities. Additional Pod instantiation capacity is modeled by the D_{nA} transition, activated under the condition $inst\_cap\_dc_n < dc\_ntotal\_cap$. Conversely, capacity reduction triggers the D_{nR} transition under $inst\_cap\_dc_n > dc\_ntotal\_cap$ conditions.

Post-disaster, Kubernetes facilitates Pod reallocation across operational nodes using the immediate transition P_mD_{nCT}, governed by the guard expression [g5], ensuring dynamic system resilience and capacity management in response to disruptions.

$$(allocated\_pod\_m < min\_pod_m) \wedge (crt\_pods\_dc_n < WN_{UPn}) \tag{4}$$

where:

$$allocated\_pod\_m = \sum_{n \in T} (\#P_mD_{nCRT} + \#P_mD_{nUP} + \#P_mD_{nDOWN}), \tag{5}$$

and

$$crt\_pods\_dc_n = \sum_{m \in U} \#P_mD_{nCTR} \tag{6}$$

and

$$crt\_pods\_dc_n = \sum_{m \in U} \#P_mD_{nCTR}, \tag{7}$$

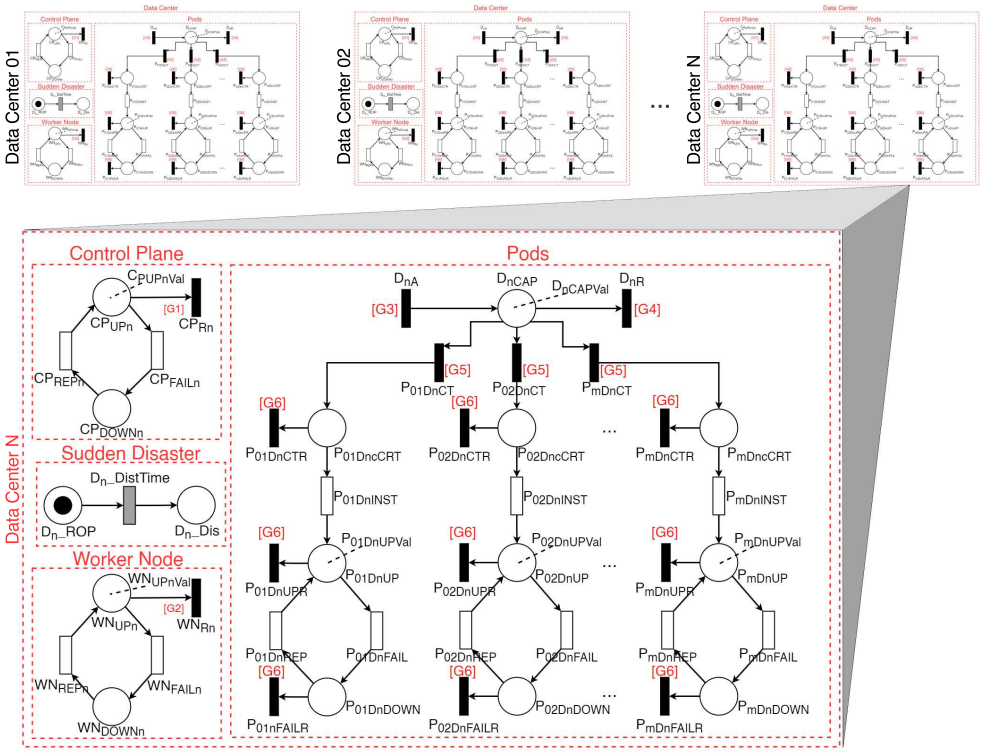


Fig. 2: GSPN of the Kubernetes infrastructure allocated in Data Center.

T = $\{n \mid n$ are the indices of DCs eligible for Pod m allocation$\}$. The guard expression [g5] activates when Pod deployment m in datacenter n falls below the minimum required amount and capacity for instantiation is available. Initiating the P_mD_{nCT} transition begins Pod creation, with the instantiation duration managed by the P_mD_{nINST} transition. Upon executing P_mD_{nINST}, a token moves from P_mD_{nCTR} to P_mD_{nUP}, restoring the designated number of Pods. Removal of Worker Nodes results from node failures or DC collapse, triggered by P_mD_{nCTR}, P_mD_{nUPR}, and P_mD_{nFAILR} transitions, dictated by guard expression [g6], as detailed in Eq. 8.

$$(allocated\_pod_m > min\_pod_m) \vee (inst\_cap\_dc_n > dc_n total\_cap) \tag{8}$$

A set of metrics was designed to assess the effectiveness of various configurations within the model. These metrics include the count of active Pods, the number of active Worker Nodes (WNs), Recovery Time (RT), and power consumption. The count of active Pods is calculated using Equation (9):

$$total\_exp\_pod = \sum_{n \in N} \sum_{m \in U} E(\#P_mD_{nUP}), \tag{9}$$

where N = $\{n \mid n$ These are the indices of the DCs$\}$. The expression $E(\#P_mD_{nUP})$ defines the expected number of tokens in place $\#P_mD_{nUP}$, this expression is represented in Eq.; (10):

$$E(\#P_mD_{nUP}) = \sum_{i=1}^{n} P(\#P_mD_{nUP} = i) * i, \tag{10}$$

where, $P(\#p_mdc_n\_on = i)$ represents the probability of there being i tokens in place $p_mdc_n\_on$ [14]. Equation (11) presents the electrical consumption of the system; it takes into account the average power used by the CP (ρ_n) and WN (σ_n) nodes, as well as the power used by each Pod ($\alpha_{n,m}$), and the average power to create each Pod ($\beta_{n,m}$), along with the value of the time required for creation ($\tau_{n,m}$):

$$P = \sum_{n \in N} \Big(E(\#cpdc_n\_on) * \rho_n) + E(\#wndc_n\_on) * \sigma_n \Big) + \sum_{n \ in N} \sum_{m \in U} \Big(E(\#p_mdc_n\_on) * \alpha_{n,m} + \frac{E(\#p_mdc_n\_crt) * \beta_{n,m}}{\tau_{n,m}} \Big), \tag{11}$$

it considers the electrical consumption of active nodes and Pods in the system, as well as the consumption of instantiating new Pods in the architecture. The RT can be obtained through the model's Mean Time to Absorption (MTTA) by adding the guard condition $(D_kDis = 0) \wedge activePods_m \leq slaPod_m$ in the transitions CP_{FAIL_n}, WN_{FAIL_n}, and P_mD_{nFAIL}, where:

$$activePods\_m = \sum_{n \in T} (\#P_mD_{nUP}), \tag{12}$$

where T = $\{n \mid n$ Are the indices of the DCs capable of running pod $m\}$. This addition causes these transitions to be

enabled as long as the system does not reach the minimum SLA value for each deployment m, in the event of a disaster on DC k. After reaching the minimum number of Pods, the model disables the transitions, leading the model to absorption, with the time for absorption being RT.

V. Case Study

This section presents the outcomes of the analyses performed on the models discussed in Section IV, adhering to the system parameters listed in Table II [15–17]. The system configuration under review includes three data centers (DCs), each comprising fourteen nodes. Of these, three nodes per DC function as Control Planes (CP), and the others operate as Worker Nodes (WN). The system is structured to support three deployments, which are capable of being distributed across any of the DCs. For computation and analysis, the Mercury modeling tool was employed [18], facilitating the design and evaluation of GSPN metrics.

TABLE II: System Parameters.

Parameter	Value
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

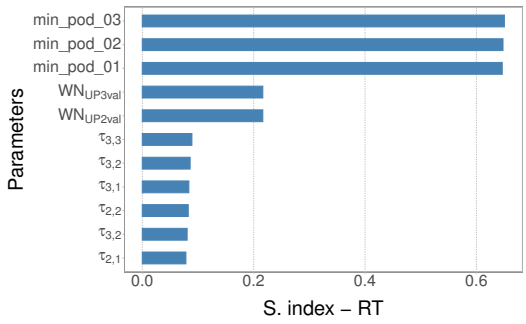
This case study explores key configurations and architectural elements that significantly affect Recovery Time (RT) and electrical consumption using sensitivity analysis. By identifying the most critical components, we prioritize changes to improve these metrics. A disaster scenario is modeled where a data center (DC) fails, drastically reducing the system's Pod count. RT is calculated using the Mean Time to Absorption (MTTA) when the system maintains a minimum set Pod configuration. Electrical consumption is measured in the steady state of the model, reflecting typical consumption outside of common operational failures. The model is calibrated to adhere to Service Level Agreements (SLAs) ensuring 150 active Pods, distributed as 50 per deployment [6].

The method applied for sensitivity analysis was the percentage difference [19], which entails altering a parameter across a set range and noting its extreme values as shown in Equation 13.

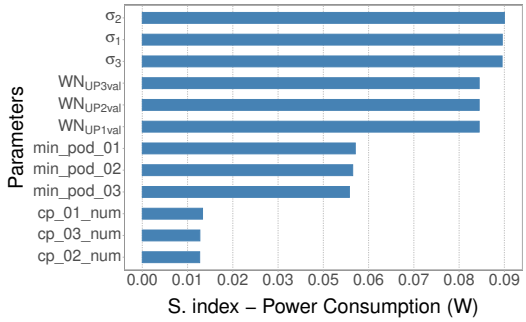
$$S_\theta\{Y\} = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}}, \tag{13}$$

For the analysis, max Y(θ) and min Y(θ) indicate the maximum and minimum values measured for the relevant metric as the parameter θ was adjusted. The alteration in parameter values was set at 25%.

Figure 3a illustrates critical Recovery Time (RT) metrics, focusing on the most impactful variables for clarity. The graph



(a) Recovery Time



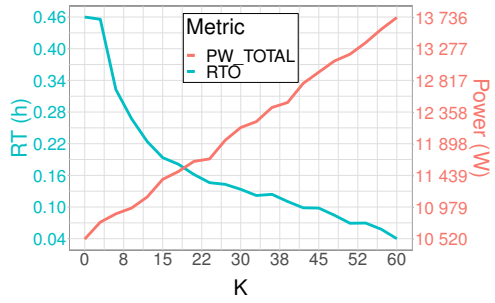
(b) Power Consumption

Fig. 3: Comparative Analysis of System Performance

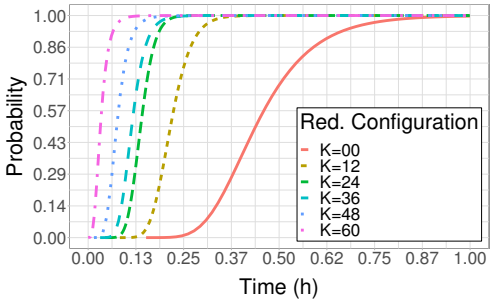
shows that the minimum number of Pods greatly influences RT, particularly in disaster scenarios; thus, increasing Pods in deployments significantly enhances system responsiveness. Additionally, Worker Node (WN) counts from Data Centers (DCs) 2 and 3 are included, highlighting their importance, as DC 1 experienced a disaster. Also, Pod instantiation times (τ_{mxn}) are crucial; however, reducing these times may be impractical due to dependencies on third-party components or the costs involved in modifying application architecture.

Figure 3b displays the sensitivity index for power consumption, with WN consumption (σ_n) identified as the most critical factor affecting electrical usage in large infrastructures. This analysis underscores the relevance of managing connected node numbers, a configurable element within the infrastructure. The graph also notes the significant role of the minimum number of Pods, adjustable via Kubernetes settings. Conversely, the impact of Control Plane (CP) numbers is less pronounced, reflected in their lower sensitivity values. These findings prioritize areas for potential efficiency improvements in infrastructure management.

Due to the impact of the minimum number of Pods on sensitivity analysis, we investigated the effects of increasing Pods beyond the Service Level Agreement (SLA) requirements (KooN) on Recovery Time (RT) and Electrical Consumption. Figure 4a displays these effects, showing that adding spare Pods reduces RT but increases electrical usage. The addition's efficacy varies: for up to 21 Pods, RT decreases by approximately 0.014 hours per Pod, whereas for more than 21 Pods, the decrease is about 0.002 hours per Pod, an



(a) RT x Power



(b) RT CDF

Fig. 4: Analyses of RT x Power and RT CDF

85% reduction efficiency, while additional power consumption stabilizes around 52.62 W per Pod.

This data aids administrators in balancing operational demands with environmental impacts, optimizing both RT reduction and energy efficiency. Figure 4b details the Cumulative Distribution Function (CDF) results, highlighting the probabilistic outcomes of varying redundancy levels. For instance, with no added redundancy (k=0), the average RT is 0.45 hours, but a 100% recovery certainty requires 1.09 hours. Increasing redundancy to k=12 lowers the average RT to 0.21 hours with complete recovery at 0.45 hours. These insights are crucial when strict recovery times are necessary, showcasing the need for precise redundancy planning to ensure compliance with stringent RT SLAs.

VI. Conclusion

This work employs GSPN to plan Recovery Time (RT) and electrical consumption in container-based microservices systems, focusing on Pod counts, RT, and energy metrics. Case studies reveal the significant impact of the minimum number of Pods on RT and how hardware configuration in Worker Nodes (WNs) influences electrical consumption. Variations in the number of spare Pods showed changing effects on RT and electrical usage, with RT decreasing by 0.85% per additional Pod while electrical consumption remained stable. Future research will explore optimization algorithms to enhance these outcomes by adjusting more variables. Additionally, we aim to incorporate scalable physical node configurations and develop strategies to further reduce RT and energy use.

References

[1] S. Bhavsar *et al.*, "Kubernetes cluster disaster recovery using aws," in *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, IEEE, 2023, pp. 1–6 (cit. on pp. 1–2).

[2] J. C. Wang, "Understanding the energy consumption of information and communications equipment: A case study of schools in taiwan," *Energy*, vol. 249, p. 123 701, 2022 (cit. on p. 1).

[3] R. M. Canosa-Reyes *et al.*, "Dynamic performance–energy tradeoff consolidation with contention-aware resource provisioning in containerized clouds," *Plos one*, vol. 17, no. 1, e0261856, 2022 (cit. on p. 1).

[4] K. S. Trivedi and R. Xia, "Quantification of system survivability," *Telecommunication Systems*, vol. 60, pp. 451–470, 2015 (cit. on pp. 1–2).

[5] E. Andrade and B. Nogueira, "Performability evaluation of a cloud-based disaster recovery solution for it environments," *Journal of Grid computing*, vol. 17, pp. 603–621, 2019 (cit. on pp. 1–2).

[6] F. Longo, R. Ghosh, V. K. Naik, A. J. Rindos, and K. S. Trivedi, "An approach for resiliency quantification of large scale systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 4, pp. 37–48, 2017 (cit. on pp. 1–2, 5).

[7] X. Y. Li, Y. Liu, Y. H. Lin, L. H. Xiao, E. Zio, and R. Kang, "A generalized petri net-based modeling framework for service reliability evaluation and management of cloud data centers," *Reliability Engineering and System Safety*, vol. 207, p. 107 381, 2021, ISSN: 09518320. DOI: 10.1016/j.ress.2020.107381 (cit. on p. 1).

[8] M. Nong, L. Huang, and M. Liu, "Allocation of resources for cloud survivability in smart manufacturing," *ACM Transactions on Management Information Systems (TMIS)*, vol. 13, no. 4, pp. 1–11, 2022 (cit. on p. 2).

[9] S. Hamadah and D. Aqel, "A proposed virtual private cloud-based disaster recovery strategy," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, IEEE, 2019, pp. 469–473 (cit. on p. 2).

[10] I. S. M. Isa, M. O. Musa, T. E. El-Gorashi, and J. M. Elmirghani, "Energy efficient and resilient infrastructure for fog computing health monitoring applications," in *2019 21st International Conference on Transparent Optical Networks (ICTON)*, IEEE, 2019, pp. 1–5 (cit. on p. 2).

[11] Kubernetes, *Kubernetes production-grade container orchestration*, https://kubernetes.io/, Accessed: 2023-08-21, 2023 (cit. on p. 2).

[12] B. Ramasamy, Y. Na, W. Kim, K. Chea, and J. Kim, "Hacm: High availability control method in container-based microservice applications over multiple clusters," *IEEE Access*, vol. 11, pp. 3461–3471, 2022 (cit. on p. 2).

[13] M. Bendechache *et al.*, "Analysing dependability and performance of a real-world elastic search application," in *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, IEEE, 2019, pp. 1–8 (cit. on p. 2).

[14] P. R. M. Maciel, *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. CRC Press, 2023 (cit. on pp. 2, 4).

[15] C. Gomes, E. Tavares, M. N. d. O. Junior, and B. Nogueira, "Cloud storage availability and performance assessment: A study based on nosql dbms," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 2819–2839, 2022 (cit. on p. 5).

[16] V. Kharchenko, Y. Ponochovnyi, O. Ivanchenko, H. Fesenko, and O. Illiashenko, "Combining markov and semi-markov modelling for assessing availability and cybersecurity of cloud and iot systems," *Cryptography*, vol. 6, no. 3, p. 44, 2022 (cit. on p. 5).

[17] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, "A taxonomy and survey of power models and power modeling for cloud servers," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–41, 2020 (cit. on p. 5).

[18] P. Maciel *et al.*, "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)*, IEEE, 2017, pp. 50–57 (cit. on p. 5).

[19] T. PINHEIRO *et al.*, "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*, IOS Press, vol. 29, 2021, p. 16 (cit. on p. 5).