Electronic Supplementary Material

Cloud-integrated cyber–physical systems: Reliability, performance and power consumption with shared-servers and parallelized services

Shuyi MA

School of Management, Xi'an Jiaotong University, Xi'an 710049, China; Department of Advanced Design and Systems Engineering, City University of Hong Kong, Hong Kong, China

Jin LI (✉)

School of Management, Xi'an Jiaotong University, Xi'an 710049, China

E-mail: jinlimis@xjtu.edu.cn

Jianping LI

School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China

Min XIE

Department of Advanced Design and Systems Engineering, City University of Hong Kong, Hong Kong, China

Appendix A

Table A1 Key notations and descriptions

Notions	Description
N	Set of physical nodes with $\|N\|$ nodes
S_t	Set of unfished services at time t with a size of $\|S_t\|$
n_j	Jth physical node
s_{ij}	Subservice of service s_i on node n_j
λ_h, λ_s	Hardware and software failure rates
μ_h, μ_s	Hardware and software repair rates
−1, 0, 1, 2	Service states for hardware failure, software failure, uptime, completion, respectively
m_t	Number of activated nodes at time t
x_{jt}	Number of unfinished services on node n_j at time t
a_{ij}	Binary variable to indicate whether s_{ij} exists, 1 for true and 0 otherwise
b_{ij}	Binary variable to indicate whether s_{ij} is in software failure downtime, 1 for true and 0 otherwise
h_j	Binary variable to indicate whether n_j is in hardware failure downtime, 1 for true and 0 otherwise
k	Number of uptime nodes at time t
z	Number of uptime subservices at time t
Δt	Interval length of sojourn time for state transitions
$P^{k,k+1}$, $P^{k,k-1}$	Probability that a node is repaired or failed within Δt
$P_{z,z+1}$, $P_{z,z-1}$	Probability that a virtual facility is repaired or failed within Δt
P_{IC}	Probability that the system remains in the current state within Δt
c	Scalarized resource capacity of each node
r_{ijt}	Scalarized resource share of service s_{ij} at time t
w_i	Workload requirement of service s_i
w_{ij}, wp_{ijt}	Workload requirement of service s_{ij} and the service progress at time t
d_i	Deadline requirement of service s_i
v_{ijt}	Discounted processing speed of service s_{ij} at time t
T_{ij}^{s}	Total service time of s_{ij}
T_{ij}^{l}	Latest hardware failure repair time of s_{ij}
T_{ij}^{d}	Time delay in the last execution due to software failures of s_{ij}
T_{ij}^{w}	Workload required minimum uptime of s_{ij}
T_{jt}	Minimum service window for node j at time t
α	Coherency factor of colocated services, $0 \le \alpha \le 1$, higher value suggests higher resource contention
ρ	Scalability factor of services, $0 \le \rho \le 1$, higher value suggests higher speedup gain from parallelism
θ	Proportion of duration of autoretries over a service period
η	Proportion of software failure caused downtime over a service period without hardware failures
β	Proportion of hardware failure caused downtime over a service period
p_0, p_1	Static and peak power of each node
RE, PE, PC	System reliability, performance, and power consumption

Appendix B

We conducted a comparison among the four distinct assignment policies outlined in Section 5.1. The key differentiating factor between them lies in the interaction between resource sharing and service parallelism. Policy 1 involves traditional assignments devoid of any resource sharing or service parallelism strategies. This approach examines the remaining capacity of each physical node and assigns a task to a suitable node based on its capabilities. In contrast, Policy 2 integrates resource sharing, resulting in the consolidation of services to optimize the utilization of active physical nodes. Policy 3 introduces service parallelism, distributing services across physical nodes to evenly distribute the workload. Finally, Policy 4 adopts a combination of resource sharing and service parallelism. To illustrate these concepts, we present illustrative matrices showcasing eight physical nodes with eight processing cores and four services, each requiring four processing cores. This assignment is depicted in matrix format, where each row signifies a service, and each column represents a node. The numerical value within each cell signifies the assigned cores for the respective subservice.

$$\text{Policy 1}: \begin{vmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \end{vmatrix}, \quad \text{Policy 3}: \begin{vmatrix} 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \end{vmatrix}$$

$$\text{Policy 2}: \begin{vmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}, \quad \text{Policy 4}: \begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix}$$

Appendix C

In Section 5.2, we delve into the probability distributions of availability factors through the application of the multistage SPR algorithm. Figure 5 presents a comparative analysis of the outcomes across different values of random failure rates. While our results are rooted in the context of exponentially distributed random failures, it is noteworthy that our approach remains applicable to more general scenarios that encompass time-varying failure rates.

In a standard cloud-based CPS setup, physical nodes operate continuously throughout a week, encompassing a total of 24 hours. Over time, the failure rates may escalate due to wear and tear, a phenomenon that has been discussed by Polotski et al. (2019). Meanwhile, virtual facilities are spawned to cater to specialized service requirements and are subsequently released upon task completion. Given that software failures often stem from misconfiguration, virtual facilities may exhibit higher failure rates during initial stages, as outlined in the works of Bhardwaj et al. (2021).

Adhering to the principles set forth by Guo et al. (2020), we can leverage the Weibull distribution to model systems that experience both aging and infant mortality, while the Erlang distribution is apt for capturing partially failing effects. Consider $\lambda(t)$ as the hardware or software failure rate at time t. In the case of the Weibull distribution, the random failure rate is a function of time denoted as $\lambda(t) = \lambda a(\lambda t)^{a-1}$, where $\lambda > 0$ signifies the scale parameter and $a > 0$ represents the shape parameter. For $a > 1$, the failure rate increases with time, portraying an aging system. Conversely, for $a < 1$, the failure rate diminishes with time, symbolizing an infant mortality system. On similar lines, the Erlang distribution encompasses a scale parameter $\lambda > 0$ and a positive integer shape parameter b. The failure rate is governed by the function $\lambda(t) = \frac{\lambda(\lambda t)^{b-1}}{(b-1)!\Phi(t)}$, where $\Phi(t) = 1 + \lambda t + \frac{(\lambda t)^2}{2!} + \ldots + \frac{(\lambda t)^{b-1}}{(b-1)!}$ represents the partial sum of the exponential series expansion $e^{\lambda t}$. When $b > 1$, it generates an increase, and the failure rates surge over time. Figure C1 graphically showcases the evolution of failure rates across different distribution scenarios. Panel (a) sheds light on an aging hardware system, where the hardware failure rate conforms to a Weibull distribution with $\lambda_h = 0.005$ and $a = 1.2$. Panel (b) unveils another aging hardware system where the hardware failure rate adheres to an Erlang distribution with $\lambda_h = 0.005$ and $b = 4$. Last, panel (c) offers insights into an infant

mortality software system, wherein the software failure rate diminishes over time, following a Weibull distribution with $\lambda_s = 0.04$ and $a = 0.8$.

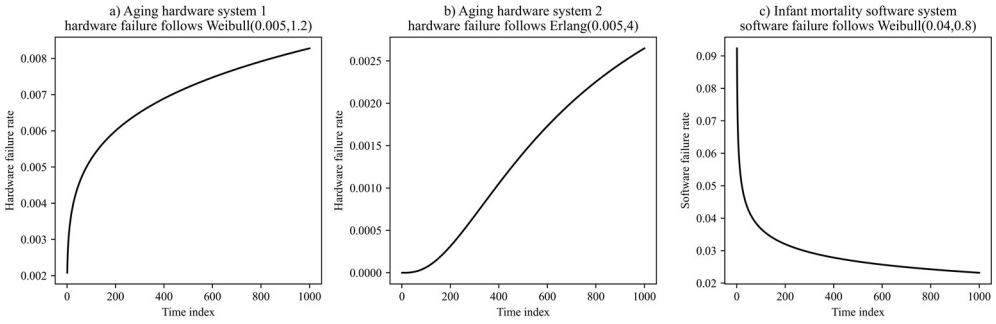

Fig. C1 Decreasing and increasing failure rates under Weibull and Erlang distributions.

Drawing from diverse failure models, in conjunction with exponentially distributed repairs featuring parameters $\mu_h = 2.5$ and $\mu_s = 0.5$, Fig. C2 engages in a comparison of availability factor distributions through the utilization of the multistage SPR algorithm. Within the context of relatively short service times (one hour), both the Weibull-distributed hardware and software failures have similar θ and β distributions to those stemming from the exponential failure model. However, it is worth noting that the hardware failure rate under the Erlang distribution tends to be lower than that under the exponential and Weibull distributions for the majority of time periods, a trend depicted in Fig. C1. Panels (a) and (c) of Fig. C2 elucidate a marked decrease in availability factors tied to hardware failures. Moreover, the software failure downtime proportion η showcases a tendency toward lower values in scenarios characterized by infant mortality software systems.

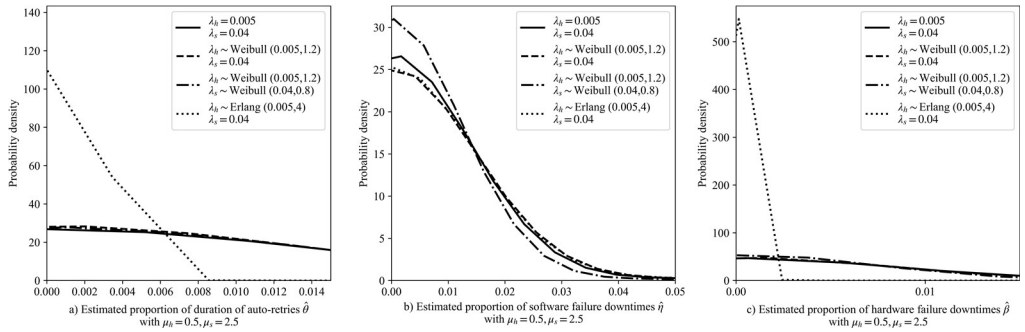

Fig. C2 Estimated distribution for availability factors under different failure models.

References

Bhardwaj A, Mangat V, Vig R, Halder S, Conti M (2021). Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions. Computer Science Review, 39: 100332 doi:10.1016/j.cosrev.2020.100332

Guo Z, Li J, Ramesh R (2020). Scalable, adaptable, and fast estimation of transient downtime in virtual infrastructures using convex decomposition and sample path randomization. INFORMS Journal on Computing, 32(2): 321–345 doi:10.1287/ijoc.2019.0888

Polotski V, Kenne J P, Gharbi A (2019). Joint production and maintenance optimization in flexible hybrid Manufacturing-Remanufacturing systems under age-dependent deterioration. International Journal of Production Economics, 216: 239–254 doi:10.1016/j.ijpe.2019.04.023