IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 23, 2021, accepted May 8, 2021, date of publication May 17, 2021, date of current version June 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3081559

Kubernetes-Container-Cluster-Based Architecture for an Energy Management System

ZONGSHENG LI, HUA WEI, ZHONGLIANG LYU, AND CHUNJIE LIAN
Guangxi Key Laboratory of Power System Optimization and Energy Technology, Guangxi University, Nanning 530004, China
Corresponding author: Hua Wei (weihuagxu@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 51967002.

ABSTRACT This paper proposes an energy management system (EMS) architecture based on the Kubernetes container cluster to solve the problem of traditional EMSs being unable to simultaneously achieve high reliability and high resource utilization. Container cluster technology is used to encapsulate, isolate and deploy applications, which solves the problem of low system reliability caused by interlocking failures. Discrete Markov theory is applied to propose a dynamic Pod fault-tolerant EMS model. The results of the solution model are used to adjust the Pod redundancy in real-time to achieve the highest reliability to satisfy physical resource constraints. The results of the performance analysis show that the reliability of the proposed architecture is 99.9999504%. Compared with the EMS of the service-oriented architecture (SOA), the annual failure time is reduced from 3.83 minutes to 0.26 minutes. The comprehensive utilization of hardware resources increases by approximately 20%, and performance indicators such as the peak access success rate improve significantly. The proposed architecture is implemented in a real-power system, with good operating results and broad application prospects.

INDEX TERMS Energy management system, Kubernetes, container, discrete Markov theory, reliability.

I. INTRODUCTION

An energy management system (EMS) is the main component of power dispatch and monitoring, and its reliability directly affects the safe operation of power grids. The existing EMSs adopted a near-regional preparation method during construction and deployment, which introduces challenges associated with resisting earthquakes, typhoons, and other regional extreme natural disasters. For example, the 5.12 Wenchuan earthquake incapacitated the power dispatch system in some areas, and the entire power grid lost its monitoring ability. In September 2016, Typhoon ''Moranti'' blew away the main and backup dispatching systems of an actual power grid in China, causing loss of control of the grid and power outages in large areas. In addition, EMS software failure and alarm system failure have resulted in power outages. Therefore, improving the reliability of EMSs has become the focus of current EMS research. In the context of today's rapid technological innovation, the monolithic architecture can no longer meet the low coupling and high-reliability requirements of EMSs. The service-oriented architecture (SOA) first proposed by Gartner in 1996 is the basic framework for reorganizing web services. This architecture offers easy replacement and high reliability and has quickly become the mainstream option for software development. The SOA is currently being applied to EMSs [1], [2].

However, the SOA-based EMS (S-EMS) still faces limitations. First, the degree of decoupling of the SOA is still coarse-grained, and service failures within the application will cause the application to fail. Coarse-grained decoupling will make it difficult to quickly locate the fault point when an application fails, requiring the system to be shut down for a long period and thus endangering the safety of the power system. Second, good isolation between applications is lacking, and there is a risk of cascading failure due to application failure. Third, when there is a sudden increase in system access, the enterprise service bus (ESB) can introduce performance degradation into the system. Bus failure or performance degradation will affect the robustness of the entire system.

In recent years, with the rapid development of cloud technology, commercial cloud computing platforms have introduced a series of advantages, such as low cost, high reliability, reliable disaster recovery, elastic computing, and storage. Software structure innovation combined with cloud technology is the main research direction for improving EMS

The associate editor coordinating the review of this manuscript and approving it for publication was Ying Xu.

reliability. At present, relevant research work has incorporated cloud technology into power systems. Feng *et al.* [3] developed a production-level power system simulation platform based on cloud computing technology connected to the New England power grid, demonstrating that the application of cloud computing technology to a power system is offers both economic advantages and reliability. Cao *et al.* [4] used cloud-based information and communication technology for power monitoring, which reduced the operating cost of the monitoring platform. Joonsang *et al.* [5] proposed a big data management platform combined with cloud computing technology based on the information processing and storage of a smart grid to improve the efficiency and reliability of power services. Chekired *et al.* [6] proposed an effective electric vehicle charging and discharging scheduling method based on the cloud computing infrastructure, which alleviated the capacity and storage limitations of smart grids. Yang *et al.* [1] improved the system reliability by extending system standby on the cloud based on the SOA structure and combining it with a virtual private cloud. Lyu *et al.* [7] proposed an EMS based on the microservice architecture (MS-EMS), which completely decouples system applications, reduces system resource costs and improves system reliability through service resource management optimization methods.

The abovementioned studies have made apparent contributions to improving the reliability of EMSs, but they also have certain limitations. Some study only the application of cloud technology for backup or deployment of the EMS and do not fundamentally solve the problems of the EMS software itself. Yang *et al.* [1] not only used the advantages of cloud technology but also improved the EMS software architecture. However, the decoupling of the software is not sufficiently thorough, software isolation is lacking, and there is a risk of chain failure. Lyu *et al.* [7] containerized decoupled services to solve the problems of application isolation and deployment, and improved resource efficiency under the premise of ensuring reliability. However, they did not consider the constraints of physical resources, and the expansion strategy of the system was fixed, which makes it difficult to balance resource utilization and reliability.

In addition, none of the above studies considered the utilization of physical resources while improving reliability and the dynamic optimization of reliability under the condition of dynamic changes in physical resource loss.

The introduction of container technology provides a new direction for software isolation and reliability research. Docker containers use Namespace and C-Group to control the isolation of resources [8], [9]. Different containers and hosts share the underlying system image, and almost all the resources of the host can be used. Felter *et al.* [10] compared the performances of Docker and KVM by running multiple benchmarks, such as Fio, and their results showed that containers have the same or even better performance under the same conditions. The isolation between containers eliminates the risk of application failure chaining [11]. Kubernetes uses traffic redirection [12] to dynamically update [13], [14] system applications, which significantly reduces the downtime caused by application updates. The Kubernetes-based EMS (K-EMS) utilizes more fine-grained decoupling than other systems, and the decoupled service is the smallest unit of the system, which significantly improves the system fault tolerance. The cluster implements full lifecycle management of the containers [15], and can self-repair failed containers, reduce the recovery time of application failure [16], [17], and improve application reliability. In addition, the system responds to short-term performance bottlenecks such as high concurrency by elastic expansion [9] and adopts a lighter communication method to simplify the system structure and further improve the reliability of the system.

To improve EMS reliability and physical resource utilization, this paper combines Kubernetes container cluster technology with EMS, and proposes an EMS architecture based on Kubernetes container clusters. The contributions are as follows:

1) Container cluster technology is adopted to encapsulate, isolate and deploy applications, which solves the problem of low system reliability caused by failure chains.

2) An EMS reliability model based on discrete Markov theory [18] is proposed. With the flexible dynamic expansion capability of Pods, adjustment of the service fault tolerance strategy is realized, and a foundation for the dynamic optimization of resource and system reliability is laid.

3) An optimization model that aims at the highest reliability of an EMS under limited resources is proposed. An analysis shows that the reliability of EMSs with a fault-tolerant strategy using this model is 99.9999504%, and that the theoretical failure time per year is 0.26 minutes, which is 50% that of the MS-EMS and 1/15 that of the S-EMS.

The remaining chapters are organized as follows:

The second chapter describes the existing problems of the current EMS. Chapter 3 proposes the K-EMS architecture, and details its advantages. Chapter 4 introduces Markov theory into EMS reliability modeling. Chapter 5 proposes a dynamic Pod optimization model that considers the upper limit of resources and takes the highest reliability as its goal. Chapter 6 analyzes the actual operating performance of the K-EMS and compares it with that of the MS-EMS. Chapter 7 gives the conclusions.

II. SOA-BASED EMS ARCHITECTURE

Compared with the traditional monolithic architecture, the SOA structure has lower system coupling and higher system openness and flexibility. The SOA structure is the current mainstream architecture for EMS development. It connects the functional units of the application through well-defined interfaces and communicates between them. The work units are published in the form of services, and the services communicate through the ESB.

In the EMS, most applications need to call or inherit the running results of one or more applications for normal operation. Communication between applications needs to pass through the ESB.

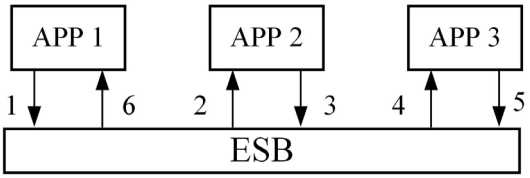

FIGURE 1. SOA-based EMS communication diagram.

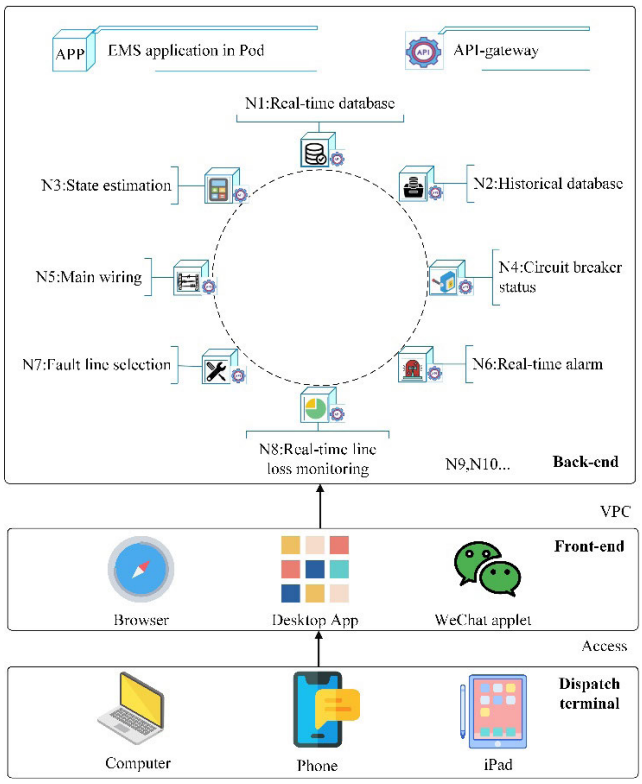

FIGURE 2. K-EMS architecture diagram.

Fig 1 is a simplified diagram of the communication process between applications in the S-EMS.

To run APP1, APP2 and APP3 need to be called successively, which requires message conversion and routing through the ESB. The ESB core functions include message conversion and message mechanism content routing.

However, there is a lack of good isolation between SOA architecture applications and a risk of cascading failures. In addition, each application of the system relies on the ESB to communicate; thus, a performance bottleneck of the ESB will affect the normal operation of an EMS when the access has high concurrency. Therefore, compared to existing EMSs, an EMS architecture with good isolation, finer-grained decoupling, more advanced communication methods, and higher reliability is needed.

III. KUBERNETES CONTAINER CLUSTER-BASED ARCHITECTURE FOR AN EMS

This article proposes an EMS architecture based on Kubernetes container clusters to improve the reliability and scalability of EMSs. As shown in Fig 2, each Pod node runs

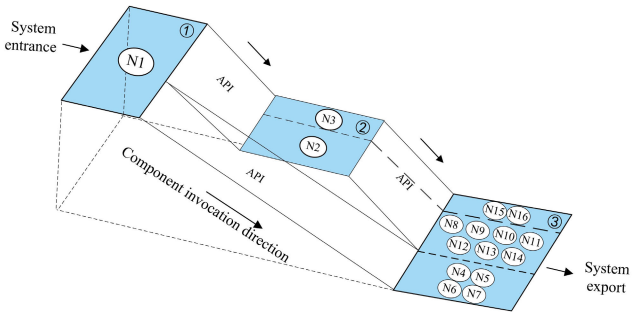

FIGURE 3. K-EMS structural transition diagram.

an EMS application and is composed of multiple containers. The containers of the system cluster communicate through the API gateway.

The system architecture adopts a decentralized distributed structure. Compared with the star architecture of the S-EMS, the ESB is abandoned, and a lighter RESTful and API gateway are used for service communication. The available architecture transfer diagram presents the calling and communication relationship of the application, as shown in Fig 3.

N1-N16 represent each EMS application, and the component calling direction indicates that a next-level component calls an upper-level component. For example, N2 calls N1, and N7 calls N2. The figure shows that all the components directly or indirectly inherit the results of the N1 (real-time database) application. The system has two levels of application nesting. The latter component communicates and inherits the result by accessing the API provided by the previous component. Compared with the S-EMS structure, the K-EMS abandons the ESB without changing the functional structure of the EMS, reduces the number of communication transfer layers of the system components, reduces the dependence between the applications, and simplifies the system communication mode.

In summary, compared with the S-EMS, the K-EMS has the following advantages:

(1) *Lighter communication method.* RESTful realizes the management of and access to resources through a URI, which has stronger scalability and clearer system structure. Communication between services is carried out by accessing the specific API of the target service, which breaks the performance bottleneck of the ESB and has high reliability.

(2) *Excellent application isolation.* The K-EMS decouples EMS applications into services and deploys them in containers. Services are effectively isolated, and cascading failure is avoided.

(3) *Upgraded expansion method.* The system expansion method is refined from the system to the application, and the system can achieve higher reliability under limited resources.

IV. EMS RELIABILITY MODEL BASED ON DISCRETE MARKOV THEORY

A. INSUFFICIENCY OF THE TRADITIONAL EMS RELIABILITY MODEL

Currently, most of the reliability models used to evaluate the EMS architecture use the reliability block diagram method

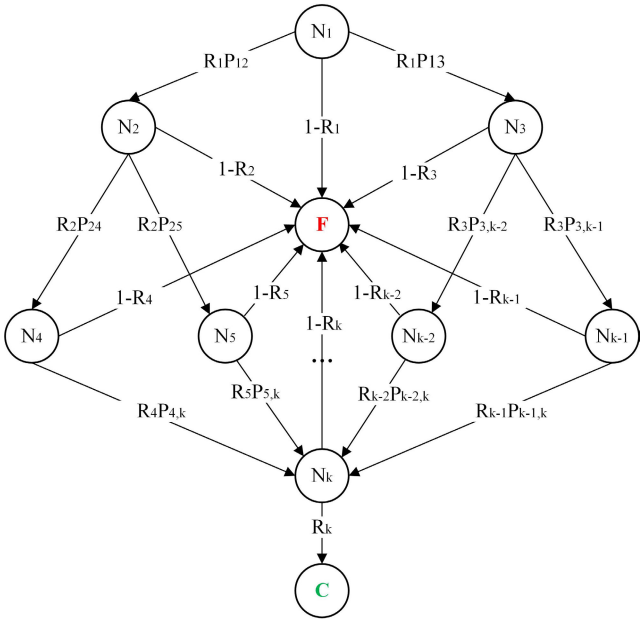

FIGURE 4. System structure diagram.

for modeling. This model treats all the applications as similar components and analyzes reliability of a system via series and parallel components [1], [7]. However, in the actual operating EMS, the frequencies used by different applications are very different, and the degree of influence on system reliability varies greatly. The reliability block diagram model cannot express the role of each application in the system, nor can it express the actual application of the system and the possible feedback of the application. Therefore, a reliability model that can more accurately reflect the actual situation of the EMS is needed.

B. DISCRETE MARKOV THEORY

The Markov process is a kind of stochastic process [19] that is widely used in power system reliability evaluation [20]. The discrete Markov chain is defined as follows:

$X = \{X_n, n \geq 0\}$ is defined in the probability space (Ω, F, P). A random process, S, is a discrete set of countable states if for any $n \geq 0$ and $i_0, i_1, \ldots, i_n, i_{n+1} \in S$:

$$P(X_{n+1} = i_{n+1} \mid X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n) = P(X_{n+1} = i_{n+1} \mid X_n = i_n) \quad (1)$$

The future state X_{n+1} is related to the current state X_n and is not related to the past state $X_0, X_1, \ldots, X_{n-1}$. Under the condition that the present state is known, the future state does not depend on a past state, that is, there is no memory.

C. INSUFFICIENT EMS RELIABILITY MODELING

Fig 4 shows the system structure diagram based on discrete Markov theory.

$N_1, N_2, N_3, \ldots, N_{(k-1)}, N_k$ represent applications in the EMS. The direction of the arrow indicates the transfer direction of the system application.

P_{ij} represents the probability of applying n_i to n_j in the EMS.

$$\sum_{j=1}^{n} P_n(i, j) = 1 \quad (2)$$

Let C and F be the successful execution state and failure state of the system, respectively. When the system has n nodes, considering the two absorption states C and F, the random transition matrix of the system can be expressed as:

$$P = \begin{bmatrix} I & 0 \\ W & Q \end{bmatrix} \quad (3)$$

Among them, the unit matrix I indicates that the system remains in this state with a probability of 1 after transitioning to this state and waits to enter the next state. The zero matrix indicates that a transition from the absorption states C and F to the node does not occur. W is the transition probability of node n_i starting and transferring to C and F. Q is the transition probability matrix between the nodes, which describes the transition probability [21] between the nodes and the reliability of the application itself.

$$Q = \begin{bmatrix} 0 & R_1P_{12} & \cdots & R_1P_{1j} & \cdots & R_1P_{1n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & R_iP_{i2} & \cdots & R_iP_{ij} & \cdots & R_iP_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & R_{n-1}P_{(n-1)2} & \cdots & R_{n-1}P_{(n-1)/} & \cdots & R_{n-1}P_{(n-1)n} \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \quad (4)$$

In Q, R_i is the reliability of EMS application n_i. An analysis of the characteristics of Q that $Q^k(i, j)$ in matrix Q^k represents the probability of successfully transferring from node d to node f in the system through k steps. From (2), the spectral radius $\rho(Q) < 1$ of the transition probability matrix is:

$$S = \mathrm{I} + Q + Q^2 + Q^3 + \cdots = \sum_{k=0}^{\infty} Q^k \quad (5)$$

It can be indicated that the series S converges; then:

$$\mathrm{S} = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1} \quad (6)$$

The reliability R of the system can be expressed as the probability that the system successfully reaches n_n after being transferred from n_1 through several intermediate nodes.

$$R_{\mathrm{sys}} = S(1, n)R_n = (I - Q)^{-1}(1, n)\, R_n \quad (7)$$

Equation (7) shows that the system reliability is determined by the system's transfer architecture and the reliability of each application. Since the reliability of each application is independent of that of another, in (8), the sensitivity s_i can be used to characterize the importance of the system applications [18].

$$s_i = \frac{\partial R_{sys}}{\partial R_i} \quad (8)$$

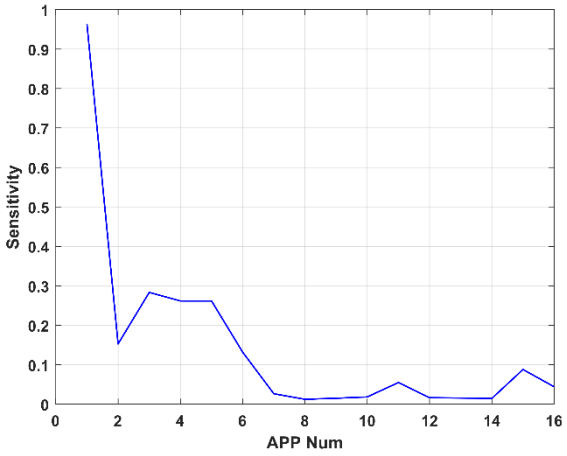

FIGURE 5. K-EMS application sensitivity.

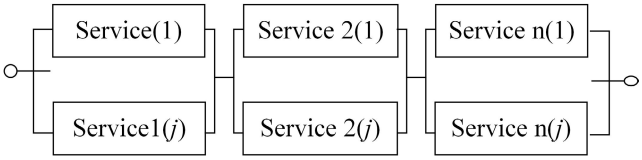

FIGURE 6. Logic diagram of the K-EMS application.

In (8), R_{sys} is the reliability of the system, R_i is the reliability of the application, the partial derivative of R_i is calculated to obtain the system reliability R_{sys}, and the result s_i is defined as the sensitivity of application N_i.

Fig 5 shows the sensitivity calculation results for 16 applications. The data show that the maximum difference in sensitivity of different applications is close to 100-fold.

The higher the application sensitivity is, the greater the impact of the application on the system. Therefore, resources should first be given to highly sensitive applications in development and maintenance.

V. SYSTEM RELIABILITY OPTIMIZATION MODEL

In the K-EMS, the smallest unit of the system is decoupled. Fig 6 shows the application logic of the K-EMS:

n represents the number of services included in a certain application. $j(j \geq 1)$ represents the number of extensions of the application in the K-EMS. The greater the number of extensions is, the stronger the fault tolerance of the application. However, the required hardware resources also increase linearly, which is evident from the EMS application resource monitoring results. Fig 7 shows the EMS application sensitivity and its physical resource consumption.

There is no obvious correlation between the sensitivity and resource consumption. Therefore, application reliability should be selectively improved under limited resources. According to (7), an optimization model with the maximum system reliability R as the objective function and the total resource usage as the constraint is established:

$$\max \ R_{\mathrm{sys}}(\boldsymbol{k}) \quad s.t. \begin{cases} \mathbf{1} \leq \boldsymbol{k} \leq \boldsymbol{n}, & \boldsymbol{k} \in \mathbb{Z} \\ 0 \leq \boldsymbol{k}^{\mathrm{T}} \boldsymbol{m} \leq \mathrm{M} \\ 0 \leq \boldsymbol{k}^{\mathrm{T}} \boldsymbol{c} \leq C \end{cases} \tag{9}$$

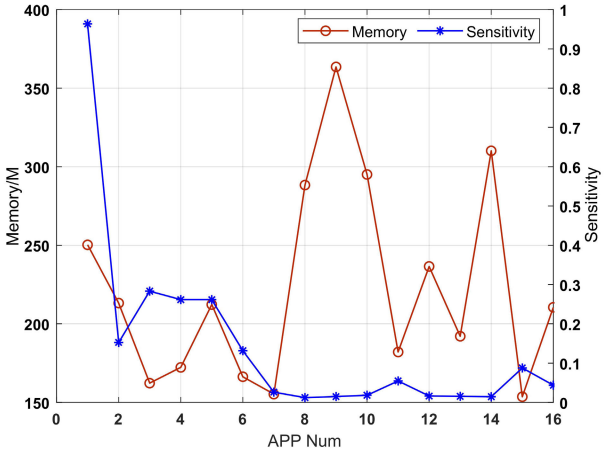

FIGURE 7. K-EMS application resource depletion.

Among them, the specific expression of the objective function R_{sys} is given in (7). $\boldsymbol{k}$ is the number of EMS application extensions, which is the decision variable of the model, and the variable type is a positive integer. $\boldsymbol{m}$ and $\boldsymbol{c}$ are the memory and CPU occupied by each application of the system when running. There are obvious differences in $\boldsymbol{m}$ and $\boldsymbol{c}$ at different time points. M and C are the upper limits of the system memory and CPU. Therefore, to improve system reliability, it is necessary to fully utilize resources as much as possible.

VI. MODEL SOLVING AND RELIABILITY ANALYSIS

A. MODEL SIMPLIFICATION

Equation (7) can be simplified to:

$$\mathrm{R}_{sys} = (I - Q)^{-1}(1, n) R_n = \frac{|X|}{|I - Q|} R_n \tag{10}$$

The determinant $|X|$ is the remainder formula corresponding to the element $(n, 1)$ of the matrix $I - Q$, which can be expressed by a specific polynomial through elementary transformation.

R_n is the reliability of the absorbing state component C in Fig 4, which has nothing to do with the reliability of the system. To simplify the calculation, R_n is set to 1 in this article.

$$|X| = \begin{vmatrix} -R_1P_{12} & -R_1P_{13} & \cdots & -R_1P_{1n} \\ 1 & -R_2P_{23} & \cdots & -R_2P_{2n} \\ 0 & 1 & \cdots & -R_3P_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & -R_{n-1}P_{(n-1)n} \end{vmatrix} \tag{11}$$

$$R_i = (1 - f^{k_i})^{s_i} \quad i = 1, 2, \ldots, 16 \tag{12}$$

f is the failure rate of a single service, which can be calculated by the reliability calculation model of the repairable component [22]:

$$f = \frac{\mathrm{MTTR}}{\mathrm{MTBF} + \mathrm{MTTR}} \tag{13}$$

k_i is the number of extensions of R_i and the decision variable of the model. s_i is the number of services included in

TABLE 1. K-EMS optimization and expansion results at 24 time nodes.

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
Time 1	3	2	3	3	3	2	2	2	2	2	2	2	2	2	3	2
Time 2	3	2	2	3	4	2	2	2	2	2	2	2	2	2	2	2
Time 3	3	2	2	2	3	2	3	2	2	2	2	2	4	2	4	2
Time 4	3	2	3	2	3	2	2	2	3	2	2	2	2	2	2	3
Time 5	3	2	2	2	3	2	2	2	2	4	2	2	2	2	2	2
Time 6	3	3	2	2	2	2	2	2	2	4	2	2	2	2	2	2
Time 7	3	2	2	2	3	2	3	2	2	2	2	2	2	2	2	2
Time 8	3	2	2	2	3	2	3	2	3	2	2	2	2	4	2	2
Time 9	3	2	2	2	2	2	2	2	2	2	2	2	2	4	3	2
Time 10	3	2	2	3	4	2	3	2	2	3	2	2	3	2	2	2
Time 11	3	3	3	3	3	3	3	2	2	2	2	2	2	2	3	3
Time 12	3	3	3	3	3	3	4	3	2	3	2	2	2	3	2	3
Time 13	3	3	3	3	3	3	3	2	2	3	2	2	2	2	3	2
Time 14	3	2	2	3	3	2	2	2	2	2	2	2	2	2	2	2
Time 15	3	2	3	3	3	2	2	2	2	2	2	2	2	2	2	2
Time 16	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2
Time 17	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
Time 18	3	2	2	3	3	2	3	2	2	2	2	2	2	2	3	2
Time 19	3	3	2	3	3	2	2	2	2	2	2	2	2	2	2	2
Time 20	3	3	3	3	3	3	2	2	2	2	2	2	3	2	2	2
Time 21	3	3	3	3	3	3	3	2	2	3	2	2	2	2	3	2
Time 22	3	2	3	3	3	3	3	2	2	2	2	2	2	2	3	2
Time 23	3	2	3	3	3	2	3	2	2	2	2	2	2	2	3	2
Time 24	3	2	3	3	3	3	2	2	2	2	2	2	2	2	2	2

R_i, which may consist of several or even dozens of services. $P(i, j)$ represents the probability that application N_i is transferred to application N_j, and the matrix is determined by the actual use of the system.

B. SOLVING ENVIRONMENT

The proposed optimization model is a mixed-integer nonlinear model, which can be solved by calling BONMIN. The solution environment is as follows:

Operating system: Windows 10 × 64
CPU: Intel(R)Core(TM)i5-8500
RAM:8.00 GB
Programming environment: MATLAB 2018b
BONMIN version: v1.8.6

C. SOLUTION RESULTS

The hardware resource requirements of EMS applications change over time. To achieve dynamic optimization of the system reliability, the data loss of an application resource at 24 time instances is used for the optimization calculation. As shown in Tab 1, Time 1-Time 24 are 24 time instances of physical resource monitoring, each timer has an interval of 5 minutes, and N1-N16 are the application numbers. The calculation result is 24 groups of 16-dimensional column vectors.

Tab 2 shows the comprehensive resource utilization, reliability and theoretical failure time at different points in time. The results show that the average reliability of the 24 time nodes of the system is 99.99995%. Similarly, the S-EMS reliability of the dual-system hot standby can be calculated; the resulting S-EMS reliability is 99.9992724%. To further describe the reliability of the system, the theoretical failure time of the system can be calculated as:

$$T_{f(k-ems)} = (1 - R_{k-ems}) \times 8760 \times 60 \approx 0.26\,\text{min}$$
$$T_{f(s-ems)} = (1 - R_{s-ems}) \times 8760 \times 60 \approx 3.83\,\text{min} \quad (14)$$

In summary, the annual theoretical failure time of the K-EMS is 1/15 that of the S-EMS, and the comprehensive hardware utilization rate of the K-EMS is approximately 20% higher than that of the S-EMS. As shown in the Tab 3.

D. PERFORMANCE TEST

The K-EMS was deployed and tested in the Alibaba Cloud container cluster to test the practicability of the architecture. The deployment environment of the container cluster is as follows:

The cluster contains two elastic compute services, both configured with the Windows Server 2019 R2 standard, 4- core 8G ROM, 100G, and respectively deployed in South China Region I and North China Region I of the Alibaba Cloud, with communication carried out through internal high-speed channels. The performance of the system is tested after the deployment has been completed. The test time was 5 minutes, and the total number of requests exceeded 200,000.

TABLE 2. Comprehensive resource utilization and reliability.

	Comprehensive resource utilization rate	Reliability	Theoretical failure time
Time 1	96.602	99.9999216	0.4121
Time 2	99.613	99.99997939	0.1083
Time 3	97.897	99.99999017	0.0517
Time 4	99.423	99.99999263	0.0388
Time 5	99.441	99.99990613	0.4934
Time 6	97.715	99.99995402	0.2417
Time 7	99.099	99.99999513	0.0256
Time 8	94.151	99.99994425	0.2930
Time 9	97.601	99.99997609	0.1257
Time 10	98.661	99.99990597	0.4942
Time 11	97.374	99.99998379	0.0852
Time 12	99.673	99.99998862	0.0598
Time 13	97.003	99.99997989	0.1057
Time 14	97.021	99.99988456	0.6068
Time 15	96.233	99.99990735	0.4870
Time 16	95.212	99.99996705	0.1732
Time 17	97.501	99.99994968	0.2645
Time 18	97.550	99.99991617	0.4406
Time 19	97.653	99.99990735	0.4870
Time 20	98.664	99.99995068	0.2592
Time 21	94.202	99.9999585	0.2181
Time 22	97.548	99.99998379	0.0852
Time 23	94.188	99.99993896	0.3208
Time 24	94.219	99.99992689	0.3843

TABLE 3. Failure time and resource utilization.

System architecture	Annual failure time (min)	CPU utilization	Memory utilization
K-EMS	0.26	96.6%	97.9%
S-EMS	3.83	81.9%	75.3%

The test used a stepwise method to increase the number of concurrencies to 100. Fig. 8 shows the test results, including the concurrency change of the test process, the response time and the access success rates of the K-EMS and S-EMS.

As shown in Tab 4, when the amount of concurrency increased, the response time, the access success rate, the resource utilization, and the system reliability of the K-EMS were better than those of the S-EMS, thus exhibiting the obvious advantages and practicality of the K-EMS.

E. COMPARATIVE ANALYSIS

Compared with the traditional single architecture, the S-EMS, MS-EMS and K-EMS all yield improved EMS reliability. The previous test and calculation results show that the

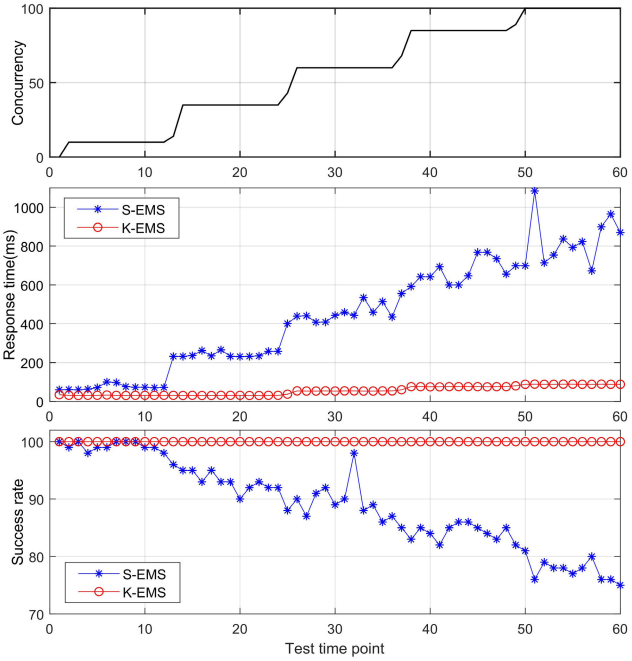

FIGURE 8. Performance test results.

TABLE 4. Software performance comparison.

Performance indicators	S-EMS	K-EMS	Trend of the indicators
Mean response time	255.67 ms	60.21 ms	↓
Peak access success rate	75%	100%	↑
CPU utilization	81.9%	96.6%	↑
Memory utilization	75.3%	97.9%	↑
System reliability	99.99927%	99.9999504%	↑

K-EMS is superior to the S-EMS in reliability and a variety of performance indicators. This section compares the reliability of the K-EMS and MS-EMS and discusses their advantages and limitations.

Taking into account the different resource management strategies of the MS-EMS and K-EMS, to ensure comparability, this section analyzes the following three different resource environments.

CASE 1: The MS-EMS has sufficient computing resources, and the upper limits of the memory and CPU of the K-EMS are 7,500 and 75, respectively.

CASE 2: The MS-EMS has sufficient computing resources, and the upper limits of the memory and CPU of the K-EMS are 9,000 and 90, respectively.

CASE 3: The MS-EMS has sufficient computing resources, and the upper limits of the memory and CPU of the K-EMS are 10,000 and 100, respectively.

TABLE 5. Reliability comparison between K-EMS and MS-EMS with different resource constraint.

Resource ceiling	System structure	Reliability	Theoretical year failure time/ min
Infinite	MS-EMS	99.9996322%	1.93
7500,75	K-EMS	99.9983712%	8.56
9000,90	K-EMS	99.9999504%	0.26
10000,100	K-EMS	99.9999886%	0.06

Substituting resource constraints into the optimization model, the application expansion method and system reliability can be solved according to equations (10) and (12), and the theoretical annual failure time of the system can be calculated from equation (14). The results are shown in Tab 5.

Under the resource constraints described in Case 1, the K-EMS cannot achieve the expansion of $k \geq 2$ in all applications, and the reliability of the system is lower than that of the MS-EMS. However, the MS-EMS cannot be deployed under the above resource conditions.

Under the resource constraints described in Case 2, the system can achieve $k \geq 2$ at 24 time nodes, and there are resource balances at some time nodes. The K-EMS can make full use of physical resources to maximize system reliability.

Under the resource constraints described in Case 3, the MS-EMS has greater resource redundancy, and the K-EMS can use these resources for further system expansion. Substituting resource constraints into the optimization model can allow calculation of the system reliability.

In summary, the reliability of the MS-EMS will not increase with the increase in system physical resources when the current resources are sufficient, while the K-EMS will flexibly expand the application of the system according to different the resource constraints involved, make full use of resources, and maximize system reliability.

The resource management strategies and target application scenarios of the MS-EMS and K-EMS are different, and each has advantages and limitations.

Compared with those of the S-EMS, the system reliability and computing resource utilization efficiency of the MS-EMS and K-EMS are greatly improved. The resource adjustment strategy of the MS-EMS can improve the efficiency in utilizing computing resources under the premise of ensuring reliability, and the efficiency is outstanding in the case of large changes in user requests. The K-EMS can achieve system reliability under the constraints of established computing resources and maximize the use of physical resources.

1) LIMITATIONS OF THE MS-EMS

1) The reliability model of the MS-EMS is a simple ''series-parallel'' component model. Compared with the related K-EMS model, the fit of the MS-EMS reliability model with the actual system and the calculation accuracy are weaker.

2) The MS-EMS adopts a fixed Pod fault-tolerant strategy, that is, a $k \geq 2$ fault-tolerant strategy. Compared with the K-EMS, the MS-EMS has lower adjustability.

2) LIMITATIONS OF THE K-EMS

1) The proposed optimization model is a mixed-integer nonlinear programming (MINLP) model. It is difficult to obtain the global optimal solution. Therefore, there is no guarantee that the optimization result will be globally optimal. A better expansion scheme may be available.

2) All services of the K-EMS are abstracted into flawless and repairable components, and all services have the same reliability. In fact, there are differences in the reliability of services with different functions. Follow-up work will further clarify the reliability of the service, making the system reliability results more accurate and more realistic.

VII. CONCLUSION

This paper proposes a K-EMS architecture to solve the reliability problem of EMS under limited resources. Containerized development and deployment methods provide reliable software isolation for EMS, and the scalability and reliability of EMSs and the performance of various software programs have been improved. A reliability optimization model based on discrete Markov theory is proposed, which realizes the maximum system reliability under the given upper limit of computing resources. The results of the performance analysis show that the reliability of the K-EMS reaches 99.9999504%. The annual theoretical failure time is only 50% that of the MS-EMS and 1/15 that of the S-EMS. The utilization rate of physical resources increases by approximately 20%. High EMS reliability can reduce power system outages caused by software failures. This architecture was implemented in an actual power system in China and exhibited excellent operating performance.

REFERENCES

[1] D. Yang, H. Wei, Y. Zhu, P. Li, and J.-C. Tan, ''Virtual private cloud based power-dispatching automation system—Architecture and application,'' *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1756–1766, Mar. 2019.

[2] H. Zhang, J. Zuo, B. Yan, and H. Huang, ''Study on the SOA construction methods in power system based on service proxy mode,'' in *Proc. IEEE Adv. Inf. Manage., Commun., Electron. Autom. Control Conf. (IMCEC)*, Xi'an, China, Oct. 2016, pp. 620–623.

[3] F. Ma, X. Luo, and E. Litvinov, ''Cloud computing for power system simulations at ISO New England—Experiences and challenges,'' *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2596–2603, Nov. 2016.

[4] Z. Cao, J. Lin, C. Wan, Y. Song, Y. Zhang, and X. Wang, ''Op-timal cloud computing resource allocation for demand side management in smart grid,'' *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1943–1955, Jul. 2017.

[5] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang, ''A secure cloud computing based framework for big data information management of smart grid,'' *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 233–244, Apr. 2015.

[6] D. A. Chekired and L. Khoukhi, ''Smart grid solution for charging and discharging services based on cloud computing scheduling,'' *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3312–3321, Dec. 2017.

[7] Z. Lyu, H. Wei, X. Bai, and C. Lian, ''Microservice-based architecture for an energy management system,'' *IEEE Syst. J.*, vol. 14, no. 4, pp. 5061–5072, Dec. 2020.

[8] G. Kappes, A. Hatzieleftheriou, and S. V. Anastasiadis, ''Multitenant access control for cloud-aware distributed filesystems,'' *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 1070–1085, Nov. 2019.

[9] S. Kehrer, F. Riebandt, and W. Blochinger, "Container-based module isolation for cloud services," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, San Francisco, CA, USA, Apr. 2019, p. 17709.
[10] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Philadelphia, PA, USA, Mar. 2015, pp. 171–172.
[11] A. Dubey, W. Emfinger, A. Gokhale, P. Kumar, D. McDermet, T. Bapty, and G. Karsai, "Enabling strong isolation for distributed real-time applications in edge computing scenarios," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, no. 7, pp. 32–45, Jul. 2019.
[12] *Kubernetes*. Accessed: Feb. 3, 2021. [Online]. Available: https://kubernetes.io/
[13] G. Chen, H. Jin, D. Zou, Z. Liang, B. B. Zhou, and H. Wang, "A framework for practical dynamic software updating," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 941–950, Apr. 2016.
[14] M. E. Segal and O. Frieder, "On-the-fly program modification: Systems for dynamic updating," *IEEE Softw.*, vol. 10, no. 2, pp. 53–65, Mar. 1993.
[15] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.
[16] Q. Liu, W. Zheng, M. Zhang, Y. Wang, and K. Yu, "Docker-based automatic deployment for nuclear fusion experimental data archive cluster," *IEEE Trans. Plasma Sci.*, vol. 46, no. 5, pp. 1281–1284, May 2018.
[17] L. Gazzola, D. Micucci, and L. Mariani, "Automatic software repair: A survey," *IEEE Trans. Softw. Eng.*, vol. 45, no. 1, pp. 34–67, Jan. 2019.
[18] R. C. Cheung, "A user-oriented software reliability model," *IEEE Trans. Softw. Eng.*, vol. SE-6, no. 2, pp. 118–125, Mar. 1980.
[19] J. P. P. Pukite, "Markov process fundamentals," in *Proc. Modeling Rel. Anal.*, Feb. 1988, pp. 49–65.
[20] W. Li, "Risk evaluation of wide area measurement and control system," in *Proc. Risk Assessment Power Syst., Models, Methods, Appl.*, 2014, pp. 313–350.
[21] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, "Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges," *IEEE Access*, vol. 6, pp. 71497–71523, 2018.
[22] W. G. Schneeweiss, "Computing failure frequency, MTBF & MTTR via mixed products of availabilities and unavailabilities," *IEEE Trans. Rel.*, vol. R-30, no. 4, pp. 362–363, Oct. 1981.

ZONGSHENG LI received the B.S. degree in electrical engineering and automation from the Guangdong University of Technology, Guangzhou, China, in 2018, where he is currently pursuing the master's degree with the Guangxi Key Laboratory of Power System Optimization and Energy Technology.

His current research interest includes the application of cloud computing to power systems.

HUA WEI received the B.S. and M.S. degrees in power engineering from Guangxi University, Nanning, China, in 1981 and 1987, respectively, and the Ph.D. degree in power engineering from Hiroshima University, Higashihiroshima, Japan, in 2002. He is currently a Professor with Guangxi University. He is also the Director with the Institute of Power System Optimization, Guangxi University. His research interests include power system operation and planning, particularly in the application of optimization theory and cloud technology to power systems.

ZHONGLIANG LYU received the B.S. degree in agricultural electrification and automation from Nanjing Agricultural University, Nanjing, China, in 2015. He is currently pursuing the Ph.D. degree in power engineering with Guangxi University, Nanning, China. His research interests include the application of optimization methods and cloud technologies to power systems.

CHUNJIE LIAN received the B.S. degree in electrical engineering from Guangxi University, Nanning, China, in 2018, where he is currently pursuing the master's degree with the Guangxi Key Laboratory of Power System Optimization and Energy Technology. His current research interest includes the application of optimization methods to power systems.