Received: 6 October 2024 | Revised: 21 October 2024 | Accepted: 5 November 2024 | Published online: 9 November 2024

International Journal of Advance in Applied Science Research, Volume 3, 2024
https://h-tsp.com/

A Dynamic Resource Allocation Strategy for Cloud-Native Applications Leveraging Markov Properties

Zhixuan Shen[1,*], Yu Ma[1], Jinnian Shen[2]

[1]Northeastern University, Portland, Maine, USA
[2]Northeastern University, Oakland, California, USA
Author to whom correspondence should be addressed.

Abstract: ***This study examines resource usage characteristics in cloud-native systems and develops a model for dynamically aligning resource allocations with application demands. Through feature analysis based on operational and monitoring data from stateless applications, we validate that time-series CPU utilization data adheres to Markovian properties. Building on this foundation, we propose a Markov-based resource allocation strategy that segments resource requirements into discrete states, enabling pre-allocation of resources tailored to each state. Additionally, an application infrastructure framework is designed to accommodate this strategy, categorizing information system services and facilitating the transmission of business workload data to the model's core. Within this framework, resource pre-allocation is abstracted into event-driven messages that dynamically manage resource scaling across service types. Finally, we validate the proposed model with data from a production system, providing insights into its efficacy through verification and analysis.***

Keywords: Cloud-native systems, resource allocation, Markov property, dynamic resource management, event-driven scaling.

1. Introduction

Dynamic pre-allocation mechanisms for resources in cloud-based data centers can significantly enhance the stability of information systems [1][2]. When integrated with operational and maintenance practices, these mechanisms contribute to reduced cluster energy consumption, highlighting their high research value. This study addresses the operational and management requirements of multiple resource entities, including computing and network resources, within the framework of cloud-native resource management. The resource management scope encompasses lifecycle considerations, virtual machine state calculations, and resource distribution [3]. The primary objective of dynamic resource adaptation is to optimize information system workloads and achieve efficient cloud resource allocation. From a business perspective, the advancement of dynamic resource matching technology holds long-term potential for improving information system service quality.

In the context of cloud computing, the resource scheduling problem is commonly approached from two main perspectives: the infrastructure resource pool and application services [4]. For the

infrastructure resource pool, scheduling strategies are designed to establish convergent boundaries, aiming to minimize both maintenance and energy costs [5]. Current research indicates that finding a global optimal solution for dynamic resource allocation strategies in cloud-native applications is unfeasible using straightforward algorithms or rules. Although approximation and deep learning methods are applied, they often yield unpredictable outcomes or impose substantial engineering costs, rendering them impractical for real-world applications. Furthermore, the cumulative workload generated by a large number of information systems often becomes highly complex and contingent on the business context, complicating the creation of a universal scheduling strategy capable of effectively handling diverse business scenarios [6]. To address these challenges, this paper proposes an event-driven resource allocation model by examining resource scheduling characteristics within the railway's main data center.

2. Main Data Center Information System Resource Scheduling Method

In traditional resource management, resources are commonly allocated to users as either physical or virtual operating systems. This approach often requires specialized configuration management tools for initial software installation and deployment, followed by standardized automation tools to streamline subsequent workflows. Despite the efficacy of this model in stable, low-demand environments, quota-based resource allocation methods struggle to keep pace with rapid business growth and variable workload demands. As a result, substantial research has been conducted, both domestically and internationally, to explore strategies for dynamically adjusting resource configurations to optimize utilization rates and minimize costs. Numerous resource allocation schemes have been proposed, primarily based on metrics such as CPU utilization and memory usage, complemented by predictive models that estimate future resource needs under varying load conditions [7]. Many of these approaches incorporate live migration techniques to facilitate vertical scaling, thus enabling dynamic resource allocation while improving overall resource utilization rates. For instance, Amazon Web Services (AWS) offers the Lambda computing service, which operates user code on a high-availability infrastructure, managing critical aspects such as server and operating system maintenance, capacity provisioning, automatic scaling, as well as continuous monitoring and logging. By executing code only when required, Lambda enables efficient scaling without the overhead of always-on resources, highlighting an industry-wide shift towards on-demand resource management.

Cloud-native data exhibits characteristics that complicate resource management further, including large volumes, diverse data types, and stringent requirements for timely processing. To meet these challenges, operations and maintenance (O&M) teams have initiated advanced research into intelligent O&M solutions tailored for cloud environments [8]. This includes a focus on intelligent monitoring and perceptual analysis, leveraging sophisticated tools to monitor the operational status of infrastructure [9]. Common architectures include Hadoop environments for data-intensive tasks, ELK (Elasticsearch, Logstash, Kibana) stacks for real-time log analysis, and traditional Zabbix environments for system monitoring. The ultimate aim of these systems is not only to detect faults but also to localize and resolve issues quickly. Despite advancements from passive monitoring to more proactive maintenance, the responsiveness at the application level remains limited, often leading to delays in detecting and addressing service degradation caused by resource bottlenecks. This is particularly relevant for cloud-native applications, where resource demands can change rapidly and require immediate adjustments.

In enterprise private clouds, resource response mechanisms are typically achieved through close collaboration between users, development teams, and operations personnel [10]. However, challenges

persist in adapting resource allocation across host machines, especially for application systems that lack fine-grained service decomposition. Applications with monolithic architectures or large-granularity services frequently rely on vertical scaling or even migration when resource bottlenecks occur. Although automated solutions can trigger resource adjustments based on predefined alerts, two major issues remain unresolved. First, the latency in responding to resource reconfiguration alerts can lead to temporary degradation or even interruption of application services, known as circuit-breaking events. Such disruptions can significantly impact user experience and overall service quality. Second, to prevent service degradation during peak loads, some operations teams over-provision resources based on peak traffic projections, leading to substantial idle capacity during off-peak periods. This approach, while safeguarding against performance degradation, often results in inefficient resource usage and increased operational costs due to resource wastage.

To address these limitations, ongoing research explores the use of more granular resource allocation models and predictive analytics to better align resource supply with demand fluctuations. Techniques such as containerization, service decomposition, and microservices are gaining traction, enabling more precise scaling capabilities. Moreover, event-driven models are emerging as promising solutions for real-time resource management, where system and application events can directly influence resource allocation decisions. By integrating predictive analytics with event-driven frameworks, resource allocation models are better positioned to respond proactively to usage patterns, reducing the need for over-provisioning while ensuring consistent service quality.

3. Business Volume Feature Analysis of Dynamic Allocation Strategy

Using the railway's main data center as a case study, we classify its operations into six distinct business categories. Prior research indicates that certain sectors, such as passenger transport and railway construction, exhibit marked cyclical fluctuations in resource demand [11]. Accordingly, variations in resource needs influenced by business load can be segmented into discrete, finite states that define the information system's resource requirements [12]. This paper focuses exclusively on the dynamic allocation strategy for computing resources and operates under the following assumptions: surplus resources are available for direct expansion of application systems, and each task can be completed within a finite timeframe.

Monitoring data from three systems within the data center was collected over 50 weeks, with CPU utilization data represented as PC, PM, and PQ . Using the natural week as the time dimension, data was divided and standardized, with proportional weighting applied. The CPU usage frequency for each system is represented by the set {PCn, PMn, PQn | n ∈ R}, where 0<n<50. After standardizing each type of monitoring data, we apply proportional weighting to form a feature variable T:

$$T = \sum PX \frac{PX_i \frac{\sum_{i=1}^{n} PX_i}{n}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(PX_i \frac{\sum_{i=1}^{n} PX_i}{n}\right)^2}} \tag{1}$$

where X ∈ {C,M,Q}; and 0< n < 50.

Given X ∈ {C,M,Q}; and 0< n < 50, with the mean μ and standard deviation σ. Based on the mean-reverting characteristic of the feature variable, the grading is set as shown in Table 1, and the deviation from the feature value is used for grading. According to the above description, the data is processed as follows: according to Table 2, the mean m=9.269 and standard deviation s=5.388 of the weighted CPU

utilization rate of the cluster are obtained. Since the CPU utilization rate of the cluster is relatively stable during this period, it is divided into four states: very low utilization, low utilization, high utilization, and very high utilization, for easy observation. The resource usage of the information system during this period can be divided into the four states as shown in Table 3.

Table 1: Grading standard

Status	Level	Classification
1	Large Low Usage	[0,m-s]
2	Low Usage	[m-s,m-0.5s]
3	Normal	[m-0.5s,m+0.5s]
4	High Usage	[m+0.5s,m+s]
5	Large High Usage	[m+s,∞]

Table 2: 50 weeks of characteristic data and status

Num	CPU Usage	Status	Week Num	Average CPU Usage	Status
1	12.519	4	26	16.002	4
2	24.202	4	27	8.169	2
3	8.939	2	28	5.143	1
4	2.411	1	29	13.896	4
5	8.764	2	30	6.048	1
6	23.453	4	31	5.290	1
7	9.692	3	32	4.339	1
8	1.745	1	33	8.816	2
9	10.598	3	34	5.561	1
10	8.737	2	35	17.761	4
11	7.306	2	36	2.418	1
12	5.356	1	37	19.683	4
13	10.279	3	38	13.821	4
14	11.985	3	39	7.336	2
15	5.152	1	40	5.043	1
16	1.986	1	41	10.727	3
17	20.450	4	42	5.171	1
18	13.897	443	43	13.057	4
19	11.073	3	44	2.115	1
20	2.885	1	45	7.036	2
21	10.283	3	46	3.241	1
22	14.109	4	47	6.50	1
23	12.276	4	48	7.201	2
24	4.707	1	49	6.832	2
25	8.750	2	50	11.325	3

A chi-squared statistical test can be used to test the Markovian property of the above-mentioned weighted CPU utilization rate sequence. Let the research sequence contain c possible states, and the transition frequency probability matrix is denoted as fij(i,jϵE) . The calculation method for the marginal probability P`j is as follows:

$$P_j = \frac{\sum_{i=1}^{C} f_{ij}}{\sum_{i=1}^{C}\sum_{j=1}^{C} f_{ij}} \quad (2)$$

where fij represents the frequency of transition from state i to state j. The marginal probability Pj' is the sum of the frequencies of transitions to state j divided by the total number of transitions in the sequence.

Table 3: System state classification

Status	Level	Classification	Numerical interval
1	Large Low Usage	[0,m-s]	[0,6.583]
2	Low Usage	[m-s,m-0.5s]	[6.583,9.295]
4	High Usage	[m+0.5s,m+s]	[9.295,11.891]
5	Large High Usage	[m+s,∞]	[11.891,∞]

Table 4: One-step specific frequency marginal probability

Status	1	2	3	4
Marginal probability	0.367	0.224	0.163	0.245

Table 5: Chi-square statistic calculation

Status	fi1	Fi2	Fi3	Fi4	Total
1	7.838	1.339	0.655	1.466	11.299
2	1.407	0.422	1.539	1.581	4.949
3	1.078	1.539	0.267	1.278	4.384
4	1.395	2.692	0.119	4.504	24.794

The transfer probability matrix element is Pi,j(i,j∈E), and the chi-square statistics can be calculated as follows:

$$X^2 = 2\sum_{i=1}^{C}\sum_{j=1}^{C} f_{ij}\left|ln\frac{P_{i,j}}{P_j}\right| \tag{3}$$

If the Chi-square distribution statistics obey the degree of freedom of (c-1)2, by querying the Chi-square distribution table, for a given confidence α, if

$$X^2 > X_a^2\langle(c-1)^2\rangle \tag{4}$$

Rejects the null hypothesis and considers the weighted CPU utilization of the cluster to be Markov under the given confidence α, as shown in Table 2: α=0.05, c=4. The one-step transition probability matrix is:

$$P^1 = \begin{bmatrix} 0.706 & 0.058 & 0.118 & 0.118 \\ 0.182 & 0.182 & 0.273 & 0.363 \\ 0.125 & 0.375 & 0.125 & 0.375 \\ 0.231 & 0.384 & 0.154 & 0.231 \end{bmatrix} \tag{5}$$

Null hypothesis rejected, test complete. The above results are calculated using CPU utilization. In a real production environment, more detailed monitoring data, such as database or cache compute resource usage, can be used for stateless applications.

4. Design and Verification of a Dynamic Resource Allocation Model

4.1 Model Design

The primary objective of the model design is to generate a resource pre-allocation strategy based on the current state of the information system using real-time monitoring data. When a state transition occurs, the model preemptively adjusts resource allocations to meet anticipated demands. This requires defining the input data sources, setting state transition probability thresholds, and establishing a front-end expansion strategy for the policy. Let the discrete state space of system resource consumption be represented by S, and let TS denote the consumption threshold corresponding to each state. The model

selects the state with the highest transition probability to determine the resource allocation in the current monitoring state. The dynamic resource allocation policy can be expressed as:

$$\min[TS_1, TS_2, \dots, TS_n] \quad \text{where } n \in S_m,\ S_m = \text{TableMap}\left[\max\left(P_j'\right), S\right] \tag{6}$$

Here, various random events serve as input source data, which can be categorized into the following five types:

System events: These events are triggered by changes in computing resources, such as CPU or memory usage alerts.

Platform events: These include events resulting from changes in the cloud platform, such as variations in platform load.

Monitoring events: Application monitoring events are derived from monitoring platforms like Prometheus, which track system metrics.

Middleware events: These events originate from middleware within the application system, including database and message queue events.

External events: Additional external events, such as those received via gRPC, provide further extensions beyond the aforementioned event types.

Within the model, two core roles are established to maintain balanced boundaries:

Threshold adaptation: This service exposes a large set of event-related metrics, such as queue length, allowing for event-based scaling by consuming specific types of event data. It communicates directly with cloud platform scaling mechanisms like the Horizontal Pod Autoscaler (HPA) to adjust the number of deployment replicas dynamically. Events are consumed directly from the source by the deployment, ensuring a rich integration of event data for processing or discarding. Queue messages and other event data are then available for immediate use.

Expansion policy: This component supports functions such as activation, deregistration, and dynamic scaling, allowing the system to scale resources down to zero in the absence of state-changing events. Scaling policies determine whether a deployment should be activated or deactivated, and they send feedback to the originating event source. This feedback loop, illustrated in Figure 1, facilitates efficient model deployment by adapting to real-time resource demands.

The model's preemptive approach, relying on state transition probabilities and event-driven mechanisms, aims to enhance resource utilization by dynamically aligning resource allocation with system demand.

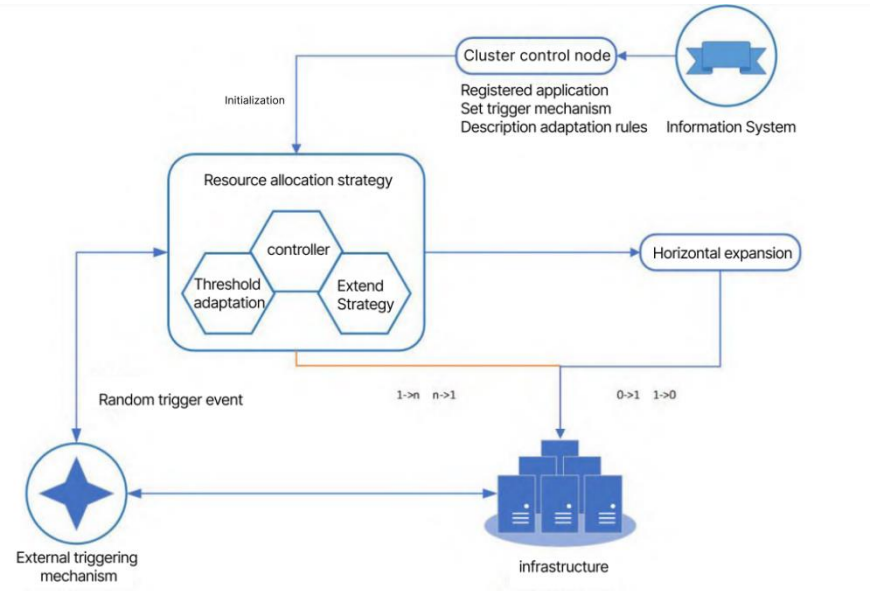

Figure 1: Model deployment

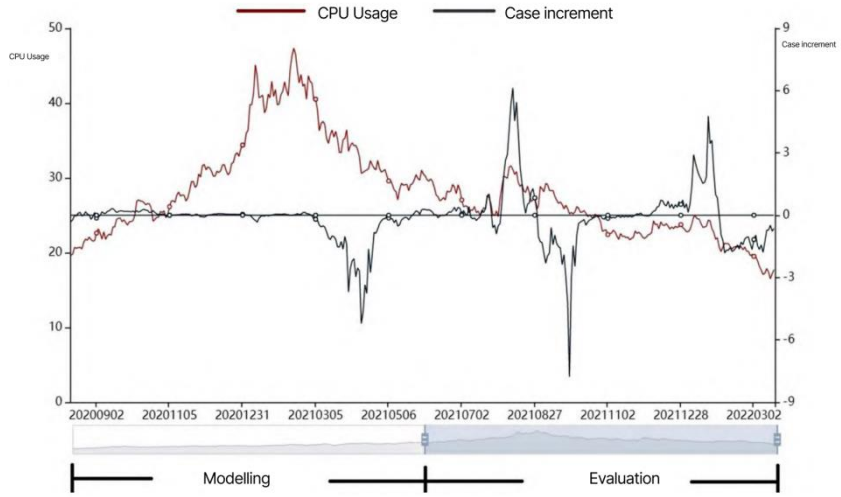

Figure 2: CPU Usage and Case increment along time

4.2 Model Evaluation

The model evaluation focuses on assessing the responsiveness of cluster resources, the accuracy of resource allocation policies, and the response time for capacity expansion in stateless application services. For this purpose, CPU utilization monitoring data from system K, deployed in an isolated environment, was used as the input data source. The data covers a period of 372 days, with the first 186 days used to determine states according to the grading standards in Table 1, which were then used to calculate the state transition probability matrix. The remaining 186 days of data served as validation data.

As shown in Figure 2, the slider at the bottom illustrates the segmentation of the dataset used for evaluation. In the main graph, represented as a broken line, all data points are upsampled according to the daily average values. Since the maximum peak CPU utilization of the test system remains below 50%, the overall task volume is relatively low, resulting in the resource allocation strategy primarily reducing or maintaining the number of instances. Forward capacity expansion is triggered only for a

few unexpected high-demand tasks, indicating that the strategy employs a somewhat "greedy" approach—allocating resources preemptively to handle potential surges in demand. While this approach provides responsiveness during business surges, it can lead to occasional resource wastage.

The Pearson correlation coefficient between the resource allocation increments and the CPU utilization data is calculated to be 0.56, indicating a moderate positive correlation. This outcome suggests that the pre-allocation strategy in the dynamic resource allocation model designed in this study correlates to some extent with subsequent fluctuations in information system traffic, affirming its potential for real-world application.

5. Conclusion

Dynamic resource allocation challenges faced by most cloud platforms, including railway master data centers, are classified as non-deterministic polynomial (NP) complete problems, which are inherently difficult to solve within polynomial time constraints. Given the specific service characteristics of the system, this study divides system resource demands into finite states, each associated with a fixed resource increment. This pre-allocation of resources for the upcoming cycle requires a balance between response accuracy and application efficiency.

The model's final allocation results reveal a "greedy" approach, which proves advantageous for capacity reduction but, to some extent, increases the energy consumption of the data center and leads to occasional underutilization of resources. Thus, to optimize both the stability of resource response and resource utilization, further research should focus on refining the greedy weight coefficient within the allocation strategy. This adjustment aims to reduce resource wastage while maintaining a responsive system.

References

[1] Mohamed A, Hamdan M, Khan S, et al. Software-defined networks for resource allocation in cloud computing: A survey[J]. Computer Networks, 2021, 195: 108151.

[2] Yan J, Huang Y, Gupta A, et al. Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach[J]. Computers and Electrical Engineering, 2022, 99: 107688.

[3] Goiri Í, Julia F, Ejarque J, et al. Introducing virtual execution environments for application lifecycle management and SLA-driven resource distribution within service providers[C]//2009 Eighth IEEE International Symposium on Network Computing and Applications. IEEE, 2009: 211-218.

[4] Gmach D, Rolia J, Cherkasova L, et al. An integrated approach to resource pool management: Policies, efficiency and quality metrics[C]//2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). IEEE, 2008: 326-335.

[5] Liu X, Buyya R. Resource management and scheduling in distributed stream processing systems: a taxonomy, review, and future directions[J]. ACM Computing Surveys (CSUR), 2020, 53(3): 1-41.

[6] Wu Z, Liu X, Ni Z, et al. A market-oriented hierarchical scheduling strategy in cloud workflow systems[J]. The Journal of Supercomputing, 2013, 63: 256-293.

[7] Weingärtner R, Bräscher G B, Westphall C B. Cloud resource management: A survey on forecasting and profiling models[J]. Journal of Network and Computer Applications, 2015, 47: 99-106.

[8] Yu S D. Research on cloud computing in the key technologies of railway intelligent operation and maintenance sharing platform[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1800(1): 012010.

[9] Aiftimiei C, Costantini A, Bucchi R, et al. Cloud Environment Automation: from infrastructure deployment to application monitoring[C]//Journal of Physics: Conference Series. IOP Publishing, 2017, 898(8): 082016.
[10] Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges[J]. Journal of internet services and applications, 2010, 1: 7-18.
[11] Grebler L, Burns L S. Construction cycles in the United States since world war II[J]. Real Estate Economics, 1982, 10(2): 123-151.
[12] Castro J, Kolp M, Mylopoulos J. Towards requirements-driven information systems engineering: the Tropos project[J]. Information systems, 2002, 27(6): 365-389.