Hindawi
Security and Communication Networks
Volume 2024, Article ID 9865215, 1 page
https://doi.org/10.1155/2024/9865215

Retraction

Retracted: High-Concurrency and High-Performance Application of Microservice Order System Based on Big Data

Security and Communication Networks

Received 8 January 2024; Accepted 8 January 2024; Published 9 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] X. Zhou, X. Wu, and Y. Chen, "High-Concurrency and High-Performance Application of Microservice Order System Based on Big Data," *Security and Communication Networks*, vol. 2022, Article ID 3424283, 11 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 3424283, 11 pages
https://doi.org/10.1155/2022/3424283

Research Article

High-Concurrency and High-Performance Application of Microservice Order System Based on Big Data

Xueyun Zhou,[1] **Xinling Wu**,[1] **and Yihong Chen**[2]

[1]*School of Engineering, Guangzhou College of Technology and Business, Guangzhou 510850, Guangdong, China*
[2]*School of Computer, China West Normal University, Nanchong 637002, Sichuan, China*

Correspondence should be addressed to Xueyun Zhou; zhouxy@gzgs.edu.cn

Received 26 April 2022; Revised 27 May 2022; Accepted 9 June 2022; Published 28 June 2022

Academic Editor: Dan Deng

The order system plays the role of the central nervous system in the integrated logistics operation. Theoretically speaking, the whole operation process of logistics service is to complete the order, the order information runs through all the links of the whole logistics, and the order processing efficiency affects the logistics process. This paper studies the high-concurrency and high-performance applications of the microservice order system based on big data, which prove that on the basis of big data technology, the performance of the microservice order system will be greatly improved. To this end, the analysis provides an overview of big data technologies and related content of microservices. It describes their application scenarios and then combines the development of the order system with a comprehensive analysis to prove its feasibility. At the same time, we compared multiple algorithms applied in microservices and found the most suitable algorithm for the order system and applied it, so that the order system can achieve high concurrency and high performance. The big data-based microservices control system proved to be useful for reasonable resource planning and for improving system performance in experiments. The final experiment found that in the microservice order system using big data, its transaction average response time was 0.01 s, and the success rate was 100%.

1. Introduction

The third-party logistics enterprises are changing from the traditional outsourcing type to the integrated logistics-type operation mode. It is precisely because of this change in mode that logistics enterprises have greatly improved in resource allocation and efficiency. Logistics enterprises will concentrate more resources and business capabilities (warehousing, transportation, distribution, information processing, and other logistics auxiliary functions). It actively expands the scope of logistics services, providing raw material supply, cargo storage, and management services to manufacturers. It can provide distribution and delivery services for dealers, and at the same time, complete the transfer of business flow, capital flow, information flow, and logistics. The application of the logistics information system is increasingly important for the operation of third-party logistics, and the control system will play a central role in managing the operation of integrated logistics. In the management of the integrated logistics operation mode, a high-performance order system is used to manage the operation, which can make the logistics information management easy, and at the same time, it can play the role of a bridge to connect the client and the client. In recent years, domestic large-scale, third-party integrated logistics enterprises have actively developed and implemented order systems, such as Baosu Logistics Comprehensive Order System and Guangdong Sinotrans Order System. The successful application of these order systems in third-party logistics companies has played a positive role in promoting the development of domestic logistics companies' order systems, and has shortened the gap with foreign-order systems. However, from the overall perspective of China's third-party logistics industry, the level of informatization of order management is still relatively low, and there is still a big gap between the popularity and application level of the order system and foreign countries. Foreign-order systems are fully functional, fully consider the application of human–computer interaction, and have the advantages of high concurrency and high performance, and the application of the order system is more urgent for the third-party logistics enterprises that are transforming to the integrated logistics model. The order system can improve and optimize the operation management capability, internal resource allocation, logistics operation efficiency, and service quality of logistics enterprises. It has important application value and practical significance to meet the needs of enterprise users for specialized, integrated, and personalized high-level logistics services.

Many operational aspects of order fulfillment deal directly or indirectly with customers and affect customer satisfaction. Therefore, the order system is an important step in the successful construction of the logistics information system. The order system will play a positive role in the improvement of logistics enterprise operation and service quality: First of all, the order system will effectively integrate the internal warehousing and transportation resources of logistics enterprises and improve the efficiency of operation management and the level of collaborative management. It cooperates with suppliers, enterprises, distributors, customers, and logistics units to achieve information-sharing and resource optimization, all five of them can use the same order system for business operations at the same time, which can be well coordinated. It establishes the whole-process supervision and control of order execution, improves operation performance, reduces costs, and improves the operation and management level of logistics enterprises. Second, the order system can provide customers with real-time logistics information services through network information technology, which allows customers to grasp the execution status of orders in time and conduct logistics service supervision and complaints, The microservice order system based on big data involves the realization of these functions and improves customer satisfaction. It establishes monitoring and early warning of abnormal orders, and handles emergencies in a timely manner, improves the quality of logistics services, and enhances the customer service level of logistics enterprises.

At present, many scholars are studying the application of microservices under big data technology, and the research of many scholars is very valuable. The fast growth of machine learning and big information system stacks, according to Miao et al., encourages constant iterative upgrades of information scientific study or working methodologies. He produces data science and large data analytical cloud platform based on distributed system for schools or non-professional research disciplines, paying special attention to the goal of pleasant collaboration among existing data science teams. It is simple to alter the constituents of each component in a serverless system. The platform includes a personal software experiment atmosphere, JupyterHub depending on Diamond and HDFS for cross use, and a visual ways to construct based on data science engineering's modular architecture; however, his technique does not reach actual results [1]. So according Herman et al., big data techniques are used to develop a big data platform that combines micro and canisters for versatility and scalability in a unique way. He employs a hybrid architecture that incorporates polyglot persistence, data lake, and Lambda architectural characteristics. The system was built for an architectural services firm, and MongoDB was utilized as the major data store. Seven studies were produced as a consequence of the implementation, demonstrating the influence of big data on choice at all levels of an organization. However, his approach is not ideal for the approach of servers in the case of huge data [2]. To decrease system cost while ensuring system QoS and stability, Zhao et al. structured bucket web service activation as a random optimization problem. To address the outlined problem, he created a value elastic serverless deployment mechanism that balances the transfer between operating costs and QoS. However, the results demonstrate that this approach has a relatively low cost [3]. The fast growth in the number computing paradigms trying to leverage the cloud continuum, according to Taherizadeh et al., has had a serious influence on microservices adoption, notably in dynamic systems for which workload numbers wax and wane or Online world of Things (IoT) gadgets change their locales dynamically. A diverse range of complicated technologies must be deployed at the same time to fully realize the promise of cloud-persistent computing applications. In this modern computer context, this complicated mix of technologies is now causing data interoperability challenges. As a result, a conceptual model is required to formally explain the notions of distinct cloud application concepts. It is often difficult to put this statement into practice [4]. Big data, according to Gramigna, has altered the way the financial enterprise provides clients. When coupled with current data records, skill in use of the Program Administrator Data (PAD), which itself is part of the usual provision of services, allows the government to comprehend cost and practical paradigms and utilize this as a foundation for circular decision-making. It is ideal to utilize links to specific data records when records are connected with unique IDs. His study is not effectively integrated in microservices because using these connected datasets necessitates a robust legal and technological infrastructure for data exchange and data consumption [5]. Zahra et al.'s distributed software concept helps HPC applications to enhance processing capacity while lowering communication expenses. To ensure sustainability for Single Programs Multiple Data (SPMD) systems, templates create a valid virtual compute nodes (MsVPUs) collaborate utilizing an offline connectivity via the Sophisticated Network Control Protocol (AMQP) protocol. Fine-grained parallel algorithms for large data categorization are used to test and evaluate their proposed virtual machine. However, the processing unit's message passing paradigm has a substantial influence on HPC, and communication costs are large, which can degrade the speed of these models [6].

Virtualization technology can expand the capacity of hardware and simplify the process of software reconfiguration. The CPU virtualization technology can simulate multiple CPUs in parallel on a single CPU, allowing one platform to run multiple operating systems at the same time. The computer system's resource scheduling problem has a significant impact on the order system's computing performance, efficiency, and service quality. However, when confronted with the diversity and dynamism of large-scale microservice servers' systems and resources, the user groups alter, as do the limits placed by users on activities. In a large data-based system, these jobs are planned. It has become a nice and warm and complex question in order to develop the system to efficaciously pace, reasonably allocate, and allocate funds in the serverless system, so that if the scheduling algorithms cost of a huge number of participants is low, the submission time is of the essence, the adequate testing is balanced, and the usage rate is high. It is a very innovative idea to apply big data technology and use microservices to design and develop the order system.

2. Overview of Big Data and Microservices and Model Building

2.1. Big Data. Various big data systems, machine learning based, open source tools or platforms, and big data data technologies have been created and may be utilized for distributed databases as a result of the rapid growth of big data technology and analytical technology [7, 8]. Big data processing and commercial services have pushed large-scale commercial needs and demands into people's daily lives, as seen by this. Today, big data-based application systems such as recommender system, predicting, analytical thinking, and analytical report tools are extensively employed. Business, science education, science and technology, school, biomedical, medicine and related fields, online networks and networks, smart cities and transit, and traffic, among other fields and applications, may all benefit from catastrophic big data processing and services. Big data-based applications, on the other hand, provide additional challenges and issues to quality assurance scientists owing to the massive volume of data created, the rapidity at which data comes, and the large range of heterogeneous data [9, 10]. There are still certain challenges and problems in the development of the microservice order system. Verifying the validity of analysis and prediction based on big data, for example, is challenging owing to the features of massive data volume and timeliness. Figure 1 shows an example of a large data application.

These quality parameters relate to different big data application scenarios, including parallelism and accuracy. Quality assurance scope and procedure for big data apps. It also includes the primary quality metrics as well as associated aspects. Statistical computation based around diversified massive datasets, system development machine learning algorithms and understanding, rational decision with uncertainty, semifunctions, and sophisticated visualization are some of the distinctive characteristics of big data applications. These distinct capabilities result in more intriguing QA and QoS needs, problems, and requirements [11]. According to recent feedback from engineers, how to ensure the quality of application systems based on big data has become an important concern.

Utilizing massive datasets and complicated intelligence algorithms, big data solutions deliver functions for prognosis, recommendation, and decision assistance. Big data program quality assurance, in general, refers to the study and implementation of different assurance procedures, methodologies, standards, guidelines, and systems in order to assure the quality of database systems based on a set of performance characteristics [12]. Figure 2 is an example of the extent of validation for a big data application's quality assurance.

2.2. Microservices. The design concept of microservices is to divide a huge business system into independent microservices, and each microservice contains a complete structure from data storage to business logic. This also means that they can function as a separate entity, but they are connected to each other. Each microservice is highly "autonomous," which is reflected in: each microservice can use different technologies to implement architecture and data storage technology as needed; each microservice can be deployed and maintained independently, with an independent life cycle and service boundary; there is no dependency between microservices from development, deployment to operation, and data exchange is only carried out through lightweight interface calls [13, 14]. Data transmission can be performed when calling through the lightweight interface, and at the same time, it can improve cohesion and reduce coupling. The microservice architecture realizes the high cohesion of a single service and the effect of low coupling between each service, which is more conducive to the local flexible change and deployment of the system.

The concept minibuses has lately gained a lot of traction in the context of distributed system software architecture. Microservices may be regarded of as a type of software architecture [15]. In a nutshell, microservices is a way of building a system out of a collection of tiny services: each service runs individually (processed zone), uses its own knowledge (database), and communicates with other services using a lightweight method (usually over HTTP or HTTPS). Each of their independent services has a complete structure and can perform corresponding operations independently. Services are designed on business skills in this manner, which fosters separation of concerns. Because big systems' design is often arranged in layers, team is usually made up of professionals that specialize in certain levels such as user experience, processing, and information [16]. The structure of this strategic priority will be comparable to the system's architecture. Even if the system is arranged in a variety of separate service modules on the middleware layer, updates or extra services may only be merged with a lot of work after it reaches a certain degree. The latter is due to the interconnectedness of layers inside them, as well as layers directly beneath them. Changing a single service frequently necessitates rebuilding and deploying the complete middleware layer [17]. Due to information systems dependencies, a modification, for example, is not confined to a single job. Microservices teams must be organized differently since they focus on a particular business feature and employed a large execution stack (containing user experience, storage, and external communication). The skills

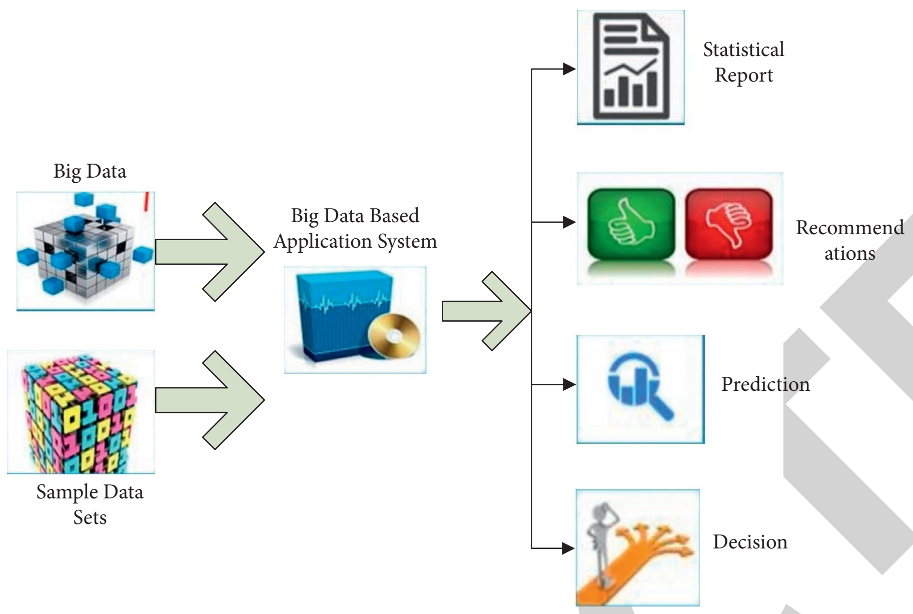


Figure 1: Typical types of big data applications.

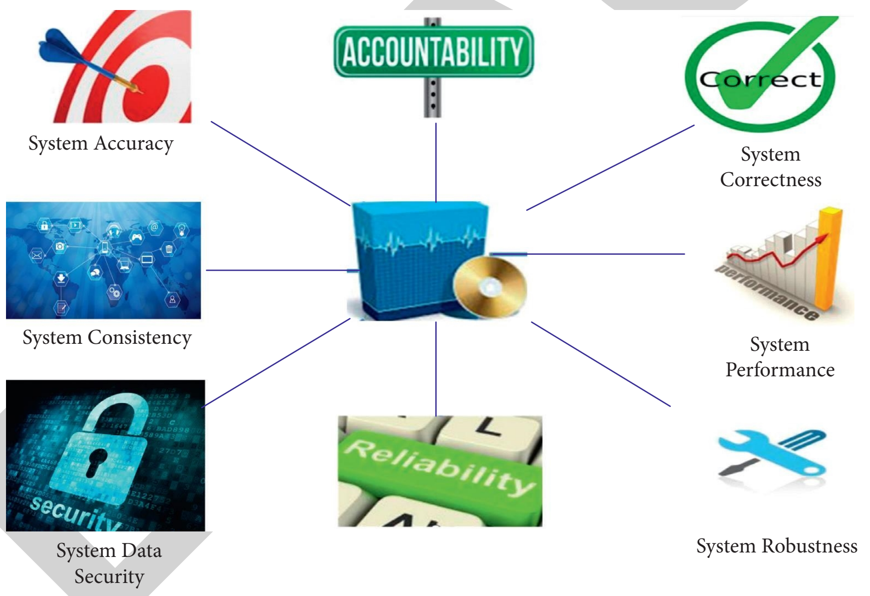


Figure 2: Example of validation scope for quality assurance for big data applications.

required to design uis, functionality, data warehouses, and product development are often found in cross-functional teams.

When necessary, companies can relocate their services. Of course, this comes at a cost, and most teams employ a completely automated deployment technique to make updates live. In addition, because modifications to one service may have an impact on another, each team must keep an eye on the overall health of the service. As a result, operations are delegated to teams (often called devops). Each vertical has all technological layers (temperature, functionality, and data services), and each vertical can contain numerous microdevices [18, 19]. An additional layer is required to handle communication between the many verticals, as well as to integrate the findings of the several verticals into a single page and provide it to the user. Figure 3 is a vertical decomposition instance of an e-commerce platform.

In the serverless workflow scheduling system, the user initially accesses the web service asset scheduling system over the web. It enters in to the software platform via various terminals, chooses cloud services that fit its requirements, and submits tasks, or cloud tasks. Microservices and cloud services are closely related, but there are many differences. Cloud service technology can be applied in the construction of microservice frameworks. Because most cloud computing uses the Map/Reduce methodology, the microservice platform separates user-submitted cloud jobs into several

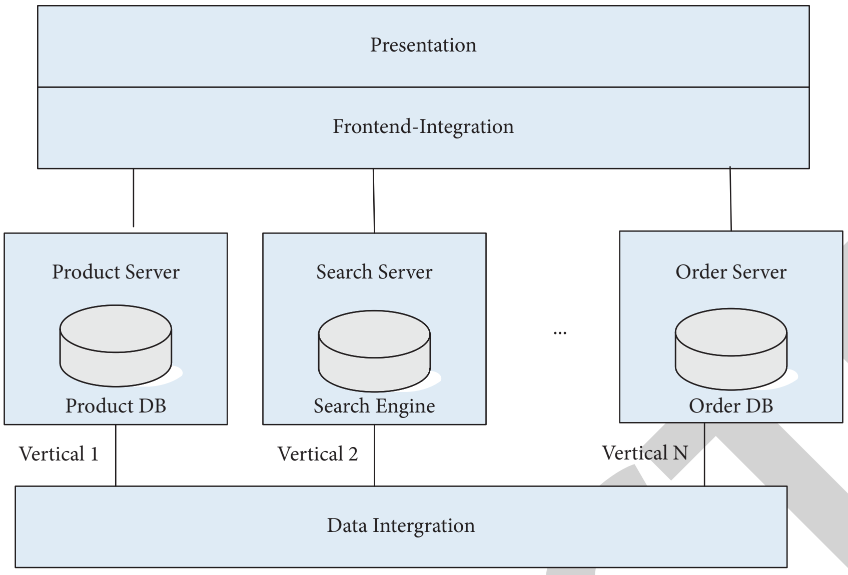


Figure 3: Decomposition based on organizational abilities and use situations.

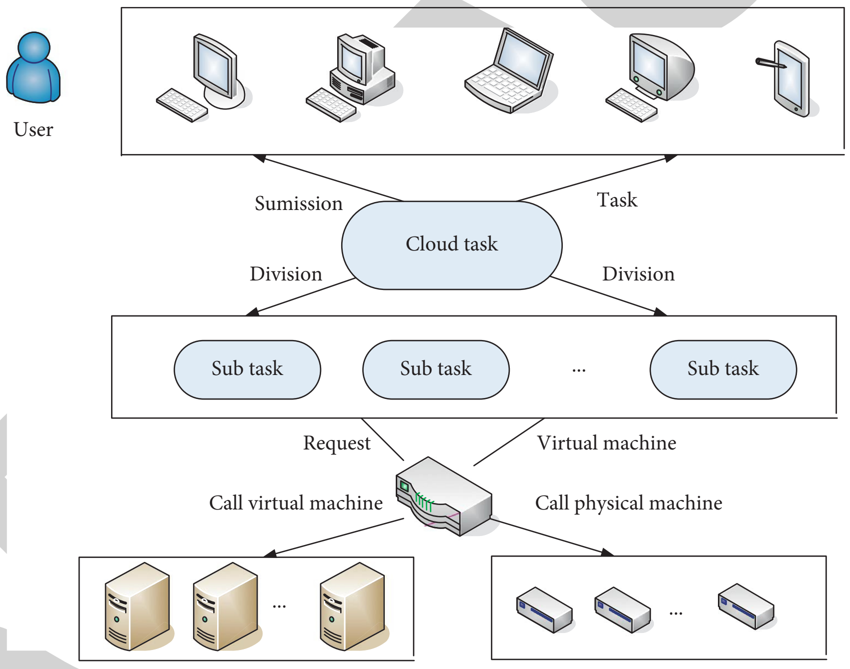


Figure 4: Microserver resource scheduling architecture diagram.

subtasks [20]. These subtasks are self-contained and can operate in parallel, after which virtual machine requests are initiated. Figure 4 depicts the web service resource scheduling structure.

The schedule center calls the relevant virtual machine after obtaining the virtualization request signal from each subtask by evaluating the information and resources for every vms and matching it with the related scheduling algorithm to execute the planned task resource. Power control, which incorporates virtual machine load balanced and physically host load balancing, should be considered throughout this planning. It not only makes high-performance hosts vulnerable to failure, but it also influences job completion times.

2.3. Scheduling Model of Microservices in the Order System.

Because each demand response and container have a one-to-one relationship, the architecture of the matching container should be defined. To accommodate for variations in execution time, sufficient resource redundant should be introduced to the same kind of jobs in various workflows.

To do so, we first compile records on cpu workloads for various jobs, and then use that data to determine the most cost-effective arrangement for each type of web service instance. We receive the carton setup scheme CE = {ce|j = 1, 2, . . ., h} when there are h forms of administrations, for which ce is a matrix indicating the set-up of the view that includes the j-th type of web service, including the number of processor cores. When an application wishes to scale up the amount of service instances for the j-th microservice, it must first construct a container with the appropriate configuration.

To handle multiworkflows, we combine workflows by adding tasks to a workflow's predecessor tasks, then all ingress tasks and subsequent tasks' tasks to exit all tasks (Line 1 Algorithm 1). Because there is no data transmission or additional jobs between computer workload and other activities, this technique can ensure that multiworkflow scheduling is fair. This strategy ensures that the initial state of each task is controlled at the same level, avoiding unfair situations. As the foundation for task ranking, we calculated the scheduler urgency u_i from each ready job in the combined workflow. An available task is one that has finished its previous task; therefore, an entrance assignment is also a ready job.

$$u_i = \frac{sd - \mathrm{XFT}(t_i)}{\mathrm{hop}(t_i)}, \tag{1}$$

where $\mathrm{hop}(t_i)$ is the percentage of unfilled tasks mostly on critical route from the exit task to the release task and $\mathrm{XFT}(t_i)$ is the estimated completion time, which is defined as follows:

$$\mathrm{XFT}(t_i) = \min_{ms\in \mathrm{MS}(t_i)} \{\mathrm{EFT}(t_i, ms, T)\}, \tag{2}$$

where $\mathrm{MS}(t_i)$ denotes the collection of instances capable of dealing with t_i. The frequency of the example necessary to determine the EFT is adjusted to the pace of instance ms because the specified instance is still undetermined.

Select the ready job with the simplest user interface for schedule based on the urgency. Task mapping must take into account how to take the most of existing capabilities as well as how to establish new eventually transfer and containers. In $\mathrm{MS}(t_i)$, laxity is calculated for all occurrences:

$$\begin{aligned}\mathrm{Laxity}\left(t_i ms_{j,k}\right) &= sd_i - \mathrm{EFT}(t_i, ms, T),\\ \mathrm{incr\,Cost}\left(t_i ms_{j,k}\right) &= \cos t' - \cos t,\\ \min\mathrm{Speed} &= \frac{w_i}{sd_i - \mathrm{IT}\left(c_{j,k}\right)}.\end{aligned} \tag{3}$$

If budget and costs represent the costs that was over, and the assignment is assigned. It is difficult to accomplish in time if minSpeed is larger than the largest speed of the allocated virtual machine. There are two plans in the EFT that we compute: the initial stage is to determine tasks to previously created instances. The second option is to construct a fast instance and then allocate tasks to it using the plan with the smallest EFT.

The divergence rate between the wealth of programs offered by the vessel and the quantity of required resources for computation is calculated using the following formula:

$$\begin{aligned}F_{\mathrm{ram}} &= \frac{s_{i,j,\mathrm{cpu}} - r_{i,j,\mathrm{cpu}}}{r_{i,j,\mathrm{cpu}}} \times 100\%,\\ F_{\mathrm{ram}} &= \frac{s_{i,j,\mathrm{ram}} - r_{i,j,\mathrm{ram}}}{r_{i,j,\mathrm{ram}}} \times 100\%,\\ F_{\mathrm{ram}} &= \frac{s_{i,j,\mathrm{ram}} - r_{i,j,\mathrm{ram}}}{r_{i,j,\mathrm{ram}}} \times 100\%.\end{aligned} \tag{4}$$

Here, F_{cpu} and F_{ram} can represent the supply and demand deviation rate of the container CPU resources and memory resources of the j-th basic service in the ith aggregate service. The delay sensitivity model expression is

$$\mathrm{t}_{i,j} = \frac{t_{ij}}{K_{\mathrm{cpu}}\left(1 + F_{\mathrm{cpu}}\right) + K_{\mathrm{ram}}\left(1 + F_{\mathrm{ram}}\right)}. \tag{5}$$

The model expression for the edge computing terminal to provide computing resources for aperiodic and nonassociative aggregated services is

$$\begin{aligned}A_{\mathrm{cpu}}(t) &= \sum_{j=1}^{n} s_{i,j,\mathrm{cpu}}\left(e\left(t - t_{i,j,imi}\right) - e\left(t - t^{*}_{i,j}\right)\right),\\ \mathrm{A}_{\mathrm{ram}}(t) &= \sum_{j=1}^{n} s_{i,j,\mathrm{ram}}\left(e\left(t - t_{i,j,imi}\right) - e\left(t - t^{*}_{i,j}\right)\right),\end{aligned} \tag{6}$$

where $A_{\mathrm{cpu}}(t)$ and $A_{\mathrm{ram}}(t)$ are the timing curves of the edge-computing terminal supplying CPU resources and memory resources for the ith aperiodic nonassociative aggregate service, respectively.

The model expression for counting the memory resources of microservices is

$$\begin{aligned}B_{\mathrm{cpu}}(t) &= \sum_{f=1}^{N} A_{l,f,\mathrm{cpu},B}\left(t - \sum_{y=1}^{f} T_{y-1}\right),\\ B_{\mathrm{ram}}(t) &= \sum_{f=1}^{N} A_{l,f,\mathrm{ram},B}\left(t - \sum_{y=1}^{f} T_{y-1}\right),\end{aligned} \tag{7}$$

where $\sum_{f=1}^{N} A_{l,f,\mathrm{cpu},B}(t - \sum_{y=1}^{f} T_{y-1})$ and $\sum_{f=1}^{N} A_{l,f,\mathrm{ram},B}(t- \sum_{y=1}^{f} T_{y-1})$, respectively, constitute the l-th aperiodic associative aggregate.

The timing curve of the CPU resources and memory resources of the f-th aperiodic nonassociative aggregate service of the service; $\sum_{y=1}^{f} T_{y-1}$ is the sum of all aggregation service delays before the f-th aggregation service in the association sequence.

The model expression for counting the external memory resources of microservices is

$$\begin{aligned}C_{\mathrm{cpu}}(t) &= \sum_{h=0}^{Q} A_{q,g,\mathrm{cpu},C}(t - hT - h\Delta t),\\ C_{\mathrm{ram}}(t) &= \sum_{h=0}^{Q} A_{q,g,\mathrm{ram},C}(t - hT - h\Delta t),\end{aligned} \tag{8}$$

where $C_{\mathrm{cpu}}(t)$ and $\mathrm{C}_{\mathrm{ram}}(t)$ are the timing curves of the microservice computing terminal supplying CPU resources and memory resources for the qth cycle nonassociative aggregated service, respectively, and Q is the number of executions.

The investment cost of resource integration is represented by C_{inv}, C_{ope} represents the cost of container usage, and C_{pen} represents the penalty cost of aggregate service delay exceeding the limit. Their expressions are as follows:

$$\begin{aligned}
\min C &= C_{\text{inv}} + C_{\text{ope}} + C_{\text{pen}},\\
C_{\text{inv}} &= E_{\text{cpu}}c_{\text{cpu}} + E_{\text{ram}}c_{\text{ram}},\\
C_{\text{ope}} &= E_{\text{ope}}c_{\text{cpu}} + E_{\text{ope}}c_{\text{ram}},\\
C_{\text{ope,cpu}} &= \sum_{i=1}^{m} s_{i,j,\text{cpu}} i_{i,j}^{*} a,\\
C_{\text{ope,ram}} &= \sum_{i=1}^{m} s_{i,j,\text{ram}} i_{i,j}^{*} a,\\
C_{\text{ope}} &= M \sum_{i=1}^{N} \left[\max\left(0, T_{\text{AS},i} - T_{\text{AS},i}^{*}\right)\right].
\end{aligned} \tag{9}$$

Here, E_{cpu} and Eram are the configuration amounts of CPU resources and memory resources of edge computing terminals, respectively; C_{cpu} and C_{ram} are the configuration costs of unit CPU resources and memory resources, respectively. The microservice running logs can be collected and stored through the real-time data collection engine Logstash and the log framework Log4j. The search server Elastic search and the visual analysis platform Kibana are used to query and analyze logs, which greatly facilitate the operation and maintenance personnel to analyze and quickly locate the failure of microservice calls.

3. Experiment and Result Analysis of the Microservice Order System

3.1. Experimental Process. Although many order solutions provide web-based modeling editors, the majority of order systems are client-server oriented. The degree of application logic embedded on the client side of the user can be separated between large and small client architectures. On the server side, a monolithic design with a database (large user) and business logic is typically used (small client). Individual aspects in business logic frequently use the same technological stack and interface via method calls, resulting in tight coupling between them. Figure 5 shows the architecture of the order system business process management tool.

Two functional scenarios are selected to test the running performance of the platform: The test environment is two 8-core, 16G virtual machines, one deploys the microservice Docker, and the other deploys the MySQL database. They are connected with 100 M network bandwidth and use Load Runner-software. The test environment and equipment of this test are shown in Figure 6.

A task's computational effort is defined as the time it takes to complete it on a typical compute service (in seconds). A workflows application is employed as a server application in this case, with each job corresponding to a microservice and each microservice handling only one type

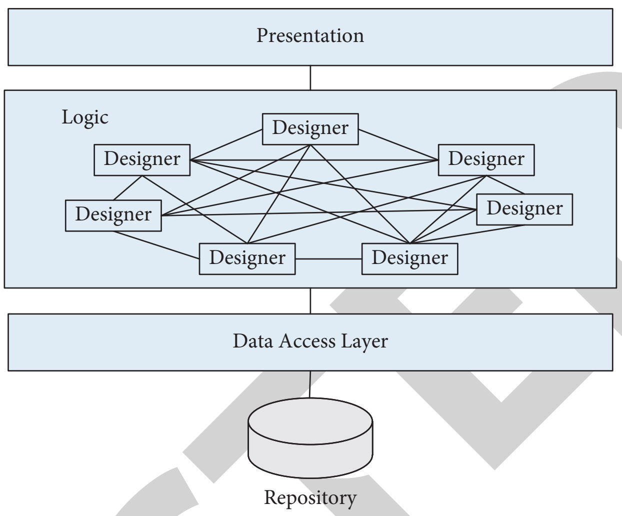


Figure 5: Architecture of business process management tools.

of task. Each task's cost of execution and other details are recorded. The ECU of Linux environment has been used to represent the virtual machine's computational capacity and resources, and the discontinuous unit is considered to be 0.5 ECU. In addition, the VM and container starting times are chosen based on relevant tests. Four distinct types of virtual machines (VMs) were employed in the studies, as indicated in Table 1.

3.2. Experimental Results. A number of methods may be employed in the building of a big data-based microservice order system to make it operate with diverse workflows, workload patterns, and workloads. We ran many sets of trials to evaluate each method's performance, then chose a steady workload pattern and ran the algorithm in a variety of workflow applications. The number of jobs in each workflow is fixed at around 50 (Because of its unusual structure, SIPHT is around 30 square feet in size).

We observed that, with the exception of SCS, the success rate gradually increased with increasing factors. The results show that ESMS outperforms the others. According to the data given in Table 2, ESMS can produce more appropriate solution than other methods.

If the order system wants to get the advantage of high performance, it needs to ensure its success rate and also control the cost. ESMS uses minimal machines while keeping costs to a minimum. Reduced operational costs complexity can also be achieved by using fewer equipment. ProLiS has the most machines in Montage. As previously discussed, its arbitrary deadline allocation causes greater and greater instances to be created, faster and faster, requiring more VMs. In obviously, the cost is determined not only by the number of virtual machines (VMs) but also by their configuration. When costs are near, however, fewer VMs are preferable, as shown in Table 3, which illustrates the VM used by various workflows.

The success rate of ESMS in the LIGO experiment was 18.31%, 36.39%, and 6.02% greater than ProLiS, SCS, and IC-PCPD2, respectively. SCS's success rate varies greatly because

Figure 6: Experiment apparatus.

Table 1: Virtual server layouts and costs.

Version	ECU	vCPU	Memory (GB)	Cost
m4.large	5.4	3	9	0.23
m5d.large	6	4	5	0.24
m4.xlarge	12	2	13	0.31
m5d.xlarge	18	3	13	0.321

Table 2: The number of reasonable choices for various workflows.

	Series of pictures	LIGO	Chromosome	SIPHT
ProLiS	7	2	8	9
SCS	1	0	1	0
IC-PCPD2	0	2	15	4
ESMS	9	9	12	9

Table 3: The number of virtual machines (VMs) employed in certain workflows.

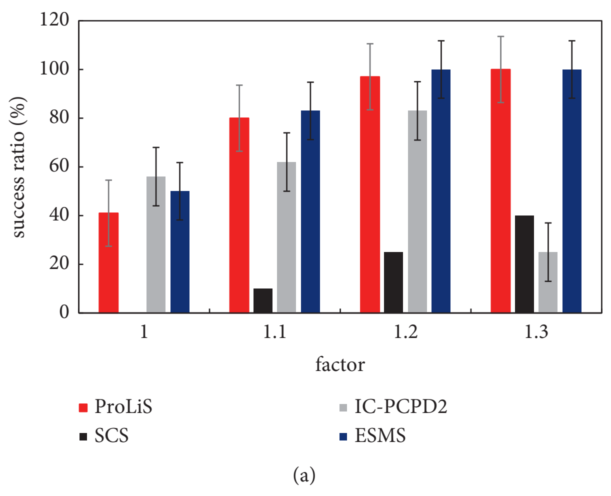
	Series of pictures	LIGO	Chromosome	SIPHT
ProLiS	7598	2154	3684	758
SCS	654	2015	2357	421
IC-PCPD2	357	1567	2458	363
ESMS	125	1587	1029	245

there is no practical answer. SCS determines number of cases based on the network vector before using the EDF scheduling method. However, the load vector's expected outcomes diverge somewhat from the EDF scheduling algorithm's requirements, forcing some activities to queue. The ensuing delay builds up during workflow execution, eventually leading it to time out. Additionally, as the factor rises, the VM's cost-efficiency varies and the configuration steadily reduces. As a result, so if the limit is loosened, the success rate remains unchanged. The success rate is higher because the other three mechanisms are all dependent on scheduling problem, and resource requirements may be established through the task-scheduling problem. The findings further show the value of scheduling tasks and autoscaling integration. IC-performance PCPD2's is likewise inconsistent: it can obtain a viable solution in the genome with high success rate, low cost, and quantity close to ESMS, but could not get any working solution in Montage. Figure 7 shows the success rates for different workflows and stable workloads.

The mean hit rate of Circuit is similar to that of ESMS for SIPHT; however, IC-PCPD2 has just three possible schemes. This is due to the fact that the hit rate of Circuit is usually between 99% and 100%, as shown in Figure 8.

In terms of value for money, all possible solutions from trials with various factor values are gathered, their costs are standardized, and the overall price is computed. ESMS costs 79.08% less than ProLiS, 6.80% less than ProLiS, 15.37% less than ProLiS, and 18.29% less than ProLiS. Except for IC-PCPD2, which has no realistic option at Montage, and cost of ESMS in other three processes is 4.57%, 0.58%, and 11.39% cheaper than IC-PCPD2. Their typical expenses are shown in Figure 9.

The improvements in success rate and cost for four workflow applications (referred to as M, L, G, and S) and three workload modes (referred to as S, D, and I) for five



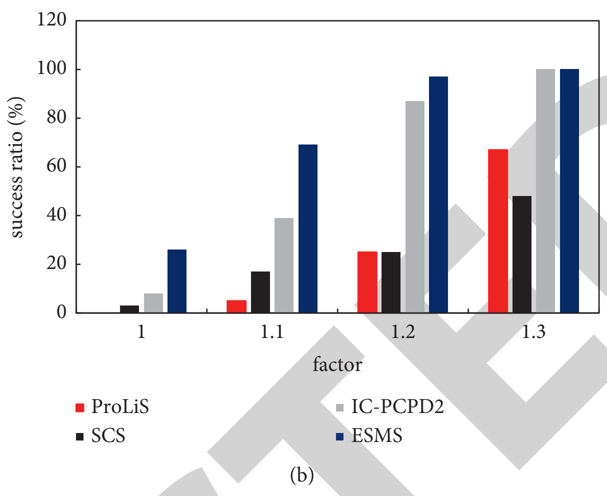
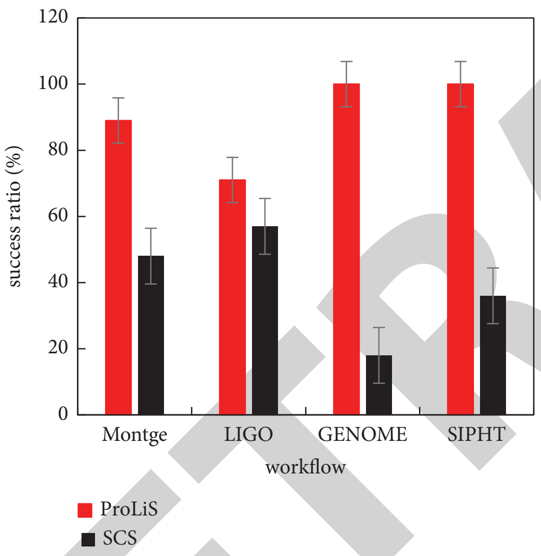


Figure 7: With a variety of workflows and consistent workloads, the success rate is high. (a) Montage. (b) LIGO.



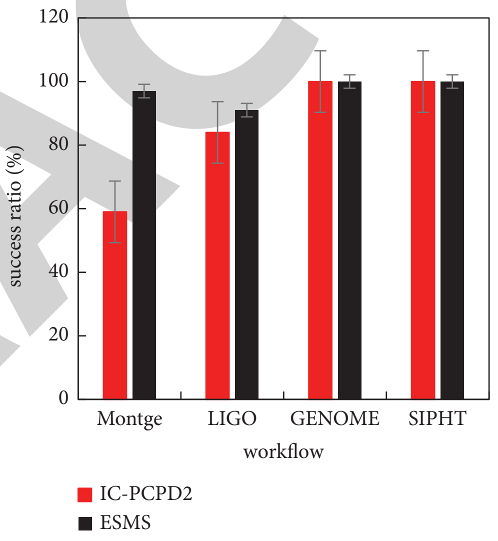


Figure 8: The success ratio.

groups are presented in Table 4. Compared with IC-PCPD2, ESMS has a performance improvement of 0.58–14.64% in cost. Overall, ESMS has a success rate of 0.00–7.40% greater than IC-PCPD2, and in Montage, it may reach 72.97%.

In the algorithm comparison experiment of the microservice framework based on big data technology, it can be seen that the performance of the ESMS algorithm is excellent in all aspects. Applying it in the development order system can make the system meet the requirements of high concurrency and high performance and can well meet the needs of order system development. The graphic simply does not matter when in the longer processes, the stores are utilized, it always fails, its response time basically does not change much, and the effect of real-time response can be achieved. The order system can perform up to six tasks at the same time, with good concurrency. The figure shows the trend of the response time over time when the big data-based microservice order system is used and also describes how many tasks the system can perform at the same time in different states. Figure 10 shows its response time and number of parallel tasks.

4. Discussion

It can be seen from the above experiments that the algorithms for selecting microservices based on big data have their own advantages. When compared to ProLiS, the ESMS algorithm offers an overall success rate improvement of 0.16%–1.30%. The improvement of LIGO, in example, is as high as 18.31%. In terms of money, the improvement ranges from 6.80 to 22.66%, with a Montage of 85.84%. Except in two rare circumstances, when the success rate is 82.01% and 99.8%, the gain in accuracy rate is generally between 27.50% and 67.62% when compared to SCS. SCS can also be compared in terms of the cost since it is unable to find a feasible solution. In GE-NOME, IC-PCPD2 can reach performance comparable to ESMS, as previously stated. In the platform performance test, multiple virtual uses, and

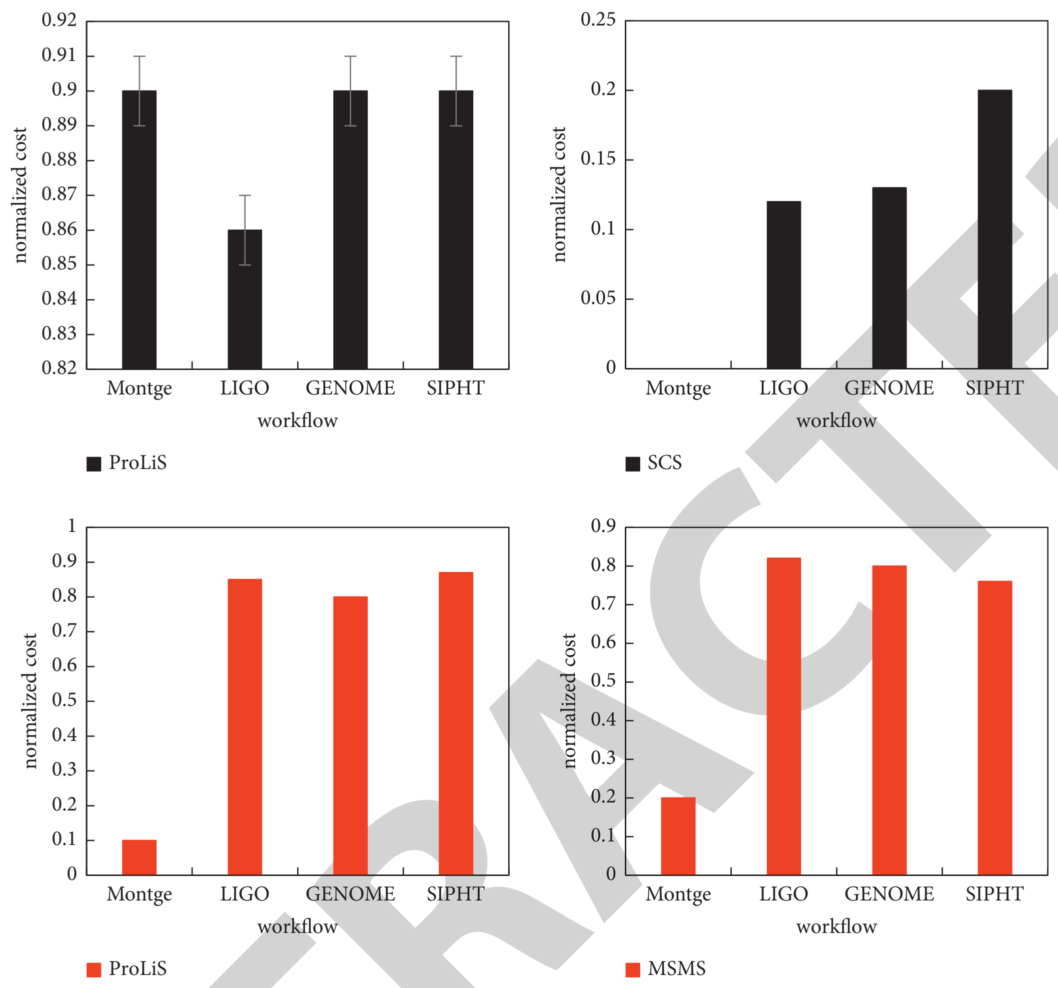


Figure 9: The normalized cost.

Table 4: The percentage improvement of ESMS above other algorithms.

Group	ProLiS (%)	IC-PCPD2 (%)	SCS (%)
1	1.30	35.54	47.68
2	0.69	70.51	48.09
3	1.17	72.97	62.00
4	18.31	6.02	36.39
5	12.30	7.20	40.25

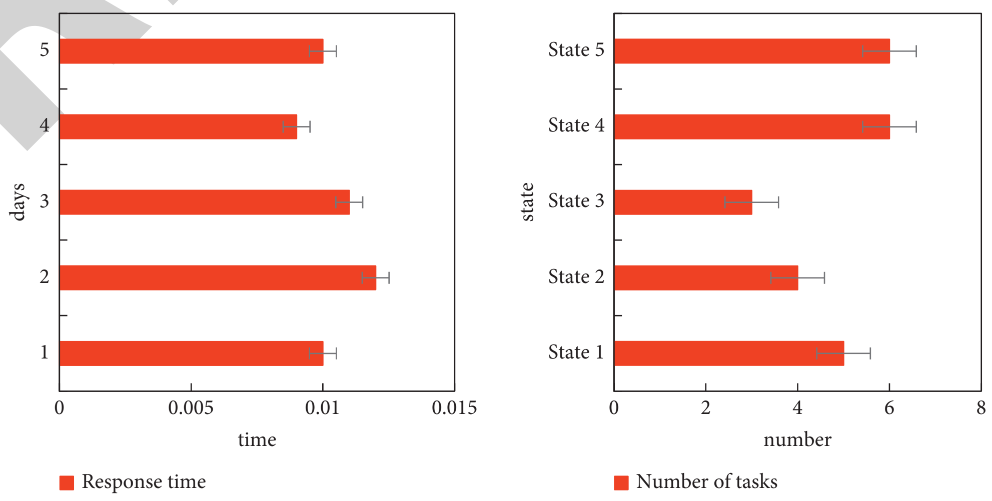


Figure 10: System response time and number of parallel tasks.

external users concurrently executed the order system for 5 days. The average transaction response time was 0.01 s, and the success rate was 100%. The results show that the smart energy service platform designed and developed according to the microservice architecture performs well in the functional performance of the order system according to the industry performance test.

5. Conclusions

The application of the microservice architecture based on big data enables the platform to easily release tests and achieve continuous integration. During the construction of the order system, as the microservices are completed one after another and have the conditions for deployment and testing, the update and launch of a single microservice is realized by relying on the microservice architecture without affecting other microservices, which greatly improves the flexibility of deployment. During service debugging, the demand response microservice had a problem and crashed. However, thanks to the microservice architecture and front-end and back-end separation technology, the platform's portal, energy trading, multienergy collaboration, and other modules still operate normally. In the operation and maintenance phase, the upgradation of demand–response microservices does not require shutting down the overall service, but only needs to update a single microservice. This not only improves the flexibility of update deployment but also improves the reliability of platform operation, and truly realizes the uninterrupted service of the system.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61871330). Quality engineering project: University-level quality engineering research project of Guangzhou College of Technology and Business in 2021 (Project's number: ZC20211122).

References

[1] K. Miao, J. Li, W. Hong, and M. Chen, "A microservice-based big data analysis platform for online educational applications," *Scientific Programming*, vol. 2020, no. 239, Article ID 6929750, 13 pages, 2020.

[2] J. Herman, H. Herman, M. J. Mathews, and J. C. Vosloo, "Using big data for insights into sustainable energy consumption in industrial and mining sectors," *Journal of Cleaner Production*, vol. 197, no. 1, pp. 1352–1364, 2018.

[3] P. Zhao, P. Wang, X. Yang, and J. Lin, "Towards cost-efficient edge intelligent computing with elastic deployment of container-based microservices," *IEEE Access*, vol. 8, no. 9, pp. 102947–102957, 2020.

[4] S. Taherizadeh, D. Apostolou, Y. Verginadis, M. Grobelnik, and G. Mentzas, "A semantic model for interchangeable microservices in cloud continuum computing," *Information*, vol. 12, no. 1, 40 pages, 2021.

[5] G. Gramigna, "Evaluating SME policies and programmes-micro-level datasets, analytical toolkits and institutional factors," *Journal of Entrepreneurship and Innovation in Emerging Economies*, vol. 3, no. 2, pp. 134–142, 2017.

[6] F. Zahra, M. Youssfi, O. Bouattane, O. Serrar, and H. Ouajji, "Toward a new massively distributed virtual machine based cloud micro-services team model for HPC: SPMD applications," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 8, pp. 238–249, 2017.

[7] Z. Jia, J. Zhan, L. Wang et al., "Understanding big data analytics workloads on modern processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1797–1810, 2017.

[8] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables DevOps: migration to a cloud-native architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, 2016.

[9] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in practice, Part 1: reality check and service design," *IEEE Software*, vol. 34, no. 1, pp. 91–98, 2017.

[10] S. Verma and S. Sekhar Bhattacharyya, "Micro-foundation strategies of IOT, BDA, cloud computing," *Strategic Direction*, vol. 32, no. 8, pp. 36–38, 2016.

[11] R. Alkurd, I. Abualhaol, and H. Yanikomeroglu, "Big-data-driven and AI-based framework to enable personalization in wireless networks," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 18–24, 2020.

[12] H.-N. Kang, H.-R. Yong, and H.-S. Hwang, "Brand clustering based on social big data: a case study," *International Journal of Software Engineering and its Applications*, vol. 10, no. 4, pp. 27–36, 2016.

[13] F. Zhao, J. Liu, J. Zhou, and L. T. Yang, "LS-AMS: an adaptive indexing structure for realtime search on microblogs," *IEEE Transactions on Big Data*, vol. 1, no. 4, pp. 125–137, 2015.

[14] P. Barkavi and J. S. Vimali, "Sentiment analysis methods and approaches: a survey," *International Journal of Pharmacy and Technology*, vol. 8, no. 4, pp. 22814–22823, 2016.

[15] S. Heitmann, S. Buri, G. Davico, and F. Reitzug, "Operationalizing ethnographic research to grow trust in digital financial services," *Ethnographic Praxis in Industry - Conference Proceedings*, vol. 2018, no. 1, pp. 537–565, 2018.

[16] R. Alkurd, I. Y. Abualhaol, and H. Yanikomeroglu, "Personalized resource allocation in wireless networks: an AI-enabled and big data-driven multi-objective optimization," *IEEE Access*, vol. 8, no. 9, pp. 144592–144609, 2020.

[17] D. Rojas-Torres and N. Kshetri, "Big data solutions for micro-, small-, and medium-sized enterprises in developing countries," *IT professional*, vol. 21, no. 5, pp. 67–70, 2019.

[18] C. Yu, "A method of public opinion analysis in big data environments," *International Journal of Simulation: Systems*, vol. 17, no. 10, pp. 1–16, 2016.

[19] A. Elmouatamid, Y. Naitmalek, M. Bakhouya et al., "An energy management platform for micro-grid systems using Internet of Things and Big-data technologies," *Proceedings of the Institution of Mechanical Engineers-Part I: Journal of Systems & Control Engineering*, vol. 233, no. 7, pp. 904–917, 2019.

[20] K. C. Finch, K. R. Snook, C. H. Duke et al., "Public health implications of social media use during natural disasters, environmental disasters, and other environmental concerns," *Natural Hazards*, vol. 83, no. 1, pp. 729–760, 2016.