Journal of Artificial Intelligence in Governance and Public Policy

Volume 1, No 1, January 2026
Available online at: jaigpp.org

JAIGPP
Journal of Artificial Intelligence in Governance and Public Policy

A Survey on Fine-Grained Resource Allocation in Microservice-Based Cloud Environments

Dr. Neha Upadhyay
Lakshmi Narain College of Technology (MCA),
LNCT Campus
Kalchuri Nagar, Raisen Road, P.O. Kolua,
Bhopal, Madhya Pradesh, India – 462022
neha.upadhyay887@gmail.com

***Abstract*—The recent spurt in the development of cloud-native applications, microservices, and edge computing has generated an immediate necessity regarding more accurate and flexible strategies to manage resources. The conventional coarse-grained techniques have difficulty in satisfying the dynamic, heterogeneous, and performance-sensitive requirements of the current distributed systems, which results in inefficiencies, bottlenecks, and elevated costs of operations. This research summarizes the current developments in Fine-Grained Resource Allocation (FGRA) in a microservice-based cloud computing environment, where the novel techniques should utilize genetic algorithms, microservice disaggregation, graph-based partitioning, search-tree access control, and cloud-edge orchestration technology. The poll also shows how new methods enhance resource use, minimize execution time, scale, and provide intelligent and tailored service deployment on a wide range of cloud deployment. Further stress is put on the factors of workload fluctuation, infrastructure heterogeneity, and changing service requirements on the efficiency of FGRA methods under real-life conditions. In general, the review shows that FGRA is important to optimize the work of microservices, support real-time service requirements, and facilitate efficient and resilient cloud-edge ecosystems needed by next-generation distributed application.**

Keywords*—Fine-Grained Resource Allocation, Microservices, Cloud Computing, Service Environment, Cloud-Native Applications.

I. INTRODUCTION

Cloud application deployment and development using microservice architecture and containerisation has become more popular due to the rapid advancement of information technology. The need to increase flexibility, scalability, and efficiency in the use of resources of the system shop is the factor driving this trend [1]. The agility of cloud applications is greatly enhanced by microservice design, which decouples complicated services and makes it easier to build, deploy, and scale individual services. The Information Technology (IT) environments today are moving away with the traditional capital-intensive systems to utility-driven systems that are on-demand [2]. This change includes a number of models, such as edge computing, cloud computing, and grid computing [3]. These all offer services to users based on managed resources, and most of the time, these services are delivered over the Internet. One way in which resources are managed by cloud service providers is through resource virtualization and multi-tenancy. These providers offer resources in the form of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [4].

Cloud computing has various applications, such as lowering costs, sharing and configuring computer resources, providing on-demand service, and being very flexible and scalable. Also, the internet-based services provided by cloud computing [5] have quickly grown in popularity. Building on earlier iterations of computer technology, such as virtualization, utility computing, and distributed computing, the cloud computing model now offers enterprise-level applications the scalability and availability they need [6], [7]. Cloud computing eliminates the need to invest in costly and specialized hardware by making high-performance computing resources and related services available over the Internet to everybody, at any time. One major perk of cloud computing is the pay-as-you-go pricing model [8]. When it comes to managing information technology resources, the cloud is revolutionary since it allows for scalability, affordability, and flexibility.

The new standard for cloud-native system design is microservice architecture [9]. Quick deployment, optimal scalability, and proper separation of concerns among services are advantages of the much-decentralized software development model, which is composed of relatively autonomous services [10]. Under this new structure, it is only natural to look forward to a wide scope of developments and simplifications as compared to the older systems. The transition to microservices rather than monolithic structures in the software development domain has been a major change. The lack of intermediaries between the system parts, the separation of applications into smaller [11], independently deployable services offered by microservices architecture is the opposite of a monolithic system, in which all the components are closely tied. Microservices have a variety of advantages over monolithic architecture, including improved fault tolerance, scalability and flexibility. Cloud computing also facilitates these advantages through on-demand resource and infrastructure.

A fine-grained technique (FGM) [12] can avoid resource fragmentation and over-commitment by matching task resource requirements with available server resources in phases. There is an effort to improve resource utilization and performance by compressing,[13] allocations when necessary in the allocation process according to resource characteristics and availability [14]. One of the most important areas in object

recognition's subspeciality, fine-grained categorization seeks to distinguish between lower-level classifications [15]. Classifying images at a finer level allows for the identification of specific types or species, such as birds, flowers, dogs, cats, and even airplane models and manufacturers.

A. Structure of the paper

The structure of this paper is as follows: Section II explores into the many methods of cloud deployment, including public, private, hybrid, and multi-cloud platforms. section III shows FGRA in microservices is discussed and compared to monolithic architecture. Section IV discuss about Various Advantages and disadvantages of the architectures. Section V examines the literature and associated research on FGRA techniques in the recent past. The work is concluded and future research directions are outlined in Section VI.

II. Cloud Deployment Models and Their Role in FGRA

Public, private, hybrid, and multi-cloud are the four best ways to describe popular cloud environments. All the models have exclusive benefits and disadvantages concerning their cost, performance, security, and complexity of management [16]. The public cloud is provided by vendors, including AWS, Microsoft Azure [17], and Google cloud, which enable the organization to rent the computing power operated in the large-scale shared data center as depicted in Fig. 1.

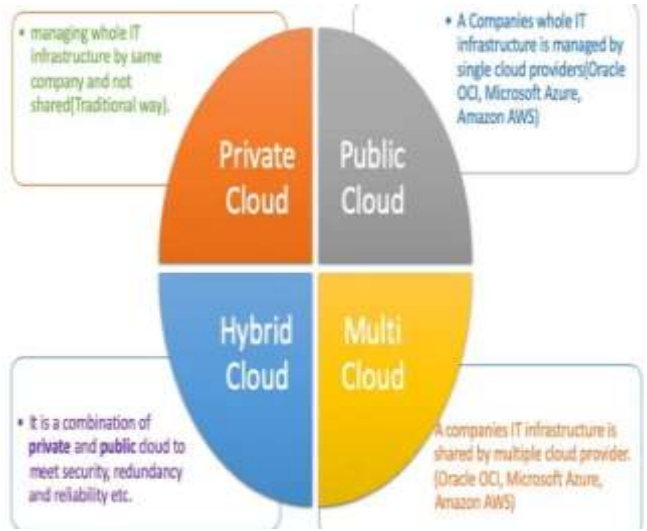

Fig. 1. Multiple Types of Cloud Environments

A. Public Cloud

The provider of cloud services is the legal owner of the underlying cloud infrastructure. A cloud provider's physical location is where can finds the cloud infrastructure. The cloud services are available on a pay-as-you-go basis to the general public or a sizable industrial group. On demand, resources in the cloud are made available to consumers. The resources are made available over the Internet on an ongoing basis. Public clouds have several advantages, especially for small and medium-sized businesses (SMEs). The benefits of public clouds include being able to scale up or down quickly, being cost-effective, reliable, flexible, and based on utility pricing. Drawbacks include a lack of personalization and security [18], as illustrated in Fig. 2.

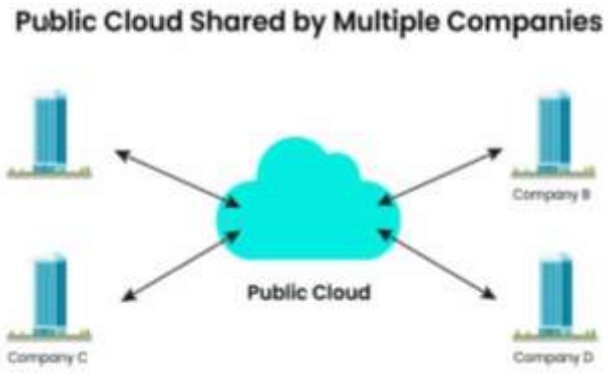

Fig. 2. Public Cloud Environment

- **Less Maintenance Cost:** Since maintenance is a difficult and expensive job, public cloud-based services [19], [20] also cover the cost of keeping IT tools in good shape.
- **Low Equipment Purchase Cost:** Using a public cloud eliminates the need to buy the many pieces of equipment that are required. Because users can only pay for and access cloud-based resources when they need them, employing public cloud-based apps is frequently less expensive than managing various software packages and other IT-related equipment.

B. Private Cloud

A computing architecture known as "private cloud" makes use of resources that are specifically allocated to company, as seen in Fig. 3. Many features of public cloud computing [21], such as resource pooling, self-service, elasticity, and pay-by-use, are also present in private clouds, but with the added control and customisation that comes with dedicated resources [22]. Private cloud comes in two varieties: on-premises private cloud and externally hosted private cloud.

Fig. 3. Private Cloud Environment

On-Premise Private Cloud: This format, which is housed in an organization's data center, is known as internal cloud. It offers a more standardized protection and procedure, however tends to be small in size and scalability. The IT department of a company would also be responsible for paying the construction and operating costs of the physical resources under this paradigm [23]. The ideal applications for on-premise private clouds are those that require complete control, infrastructure configuration, and security [24], [25].

- **Externally hosted Private Cloud:** An organization contracts with an outside cloud service provider to deploy the private cloud under the externally hosted private cloud model [26]. The cloud infrastructure is not housed on the customer organization's property, but rather on the external provider's property.

C. Hybrid Cloud

A hybrid cloud is a product that combines one or more cloud services with a private cloud solution, with proprietary software enabling communication between the various services. A hybrid cloud plan offers additional flexibility to the business since workloads can be transferred among the cloud solutions as the business requirements and costs vary. The strength of hybrid cloud services is that they provide a company with more control over its personal data. A company can use the powerful computational capabilities of a managed public cloud and keep sensitive data on a local datacenter or private cloud, all of which can be managed from a single plane of glass [27]. Fig. 4 illustrates a hybrid cloud connection to traditional infrastructure, private cloud, and public cloud.

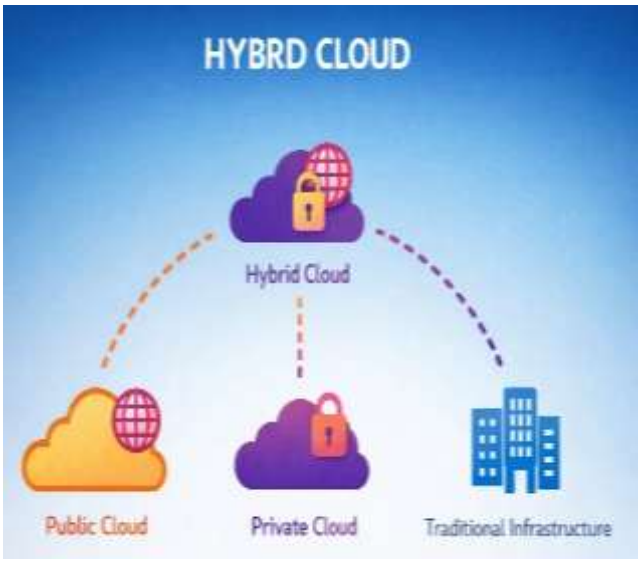

Fig. 4. Hybrid Cloud Environment

- **Hybrid Cloud Requirement:** Hybrid cloud allows for a variety of services to be offered to various users. Numerous IT-related factors influence an organization's needs [28]. varying application designers, business developers, and infrastructure support staff have varying expectations for the system because of this.

D. Multi Cloud

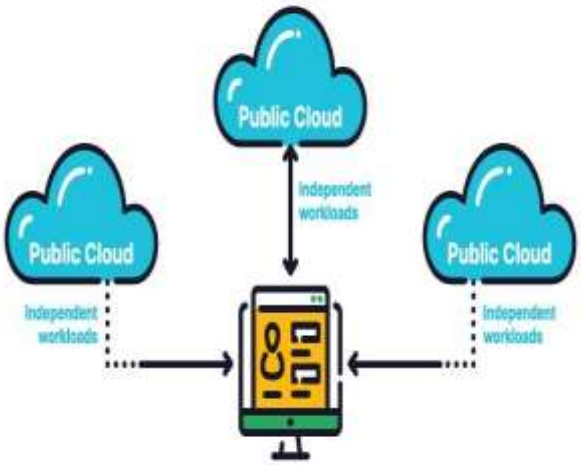

Fig. 5. Multi-Cloud Environment

Multiple cloud services from different public or private cloud providers make up a multi-cloud architecture. All hybrid clouds are multiclouds, even though not all multiclouds are hybrid clouds [29]. When many clouds are orchestrated or integrated, they form hybrid clouds [30], [31] and connect to public clouds, as seen in Fig. 5.

- **Cloud Busting:** The hybrid/multi-cloud paradigm that is most popular and straightforward is cloud bursting [32]. When the need for processing power increases, an application running in a private cloud "bursts" onto a public cloud, according to this paradigm.
- **federated cloud:** A cloud provider that subcontracts capacity from other providers and offers spare capacity to a federation of cloud providers is what defines the federated cloud scenario.

Table I shows how different types of cloud offer different levels of control, flexibility, and efficiency by comparing public, private, hybrid, and multi-cloud environments based on ownership, infrastructure, scalability, security, and cost.

TABLE I. COMPARATIVE SUMMARY OF MULTIPLE TYPES OF CLOUD ENVIRONMENTS

Aspect	Public Cloud	Private Cloud	Hybrid Cloud	Multi-Cloud
Ownership	Cloud Service Provider (e.g., AWS, Azure)	Single Organization (internal or hosted externally)	Combination of Public and Private Cloud	Multiple Cloud Vendors (public/private)
Infrastructure Location	At provider's data center	On-premise or external provider	Mix of on-premise and public provider infrastructure	Distributed across several providers
Accessibility	Open to general public or multiple clients	Restricted to one organization	Accessible to organization with selective control	Varies; managed across several vendors
Scalability	High and on-demand	Limited by internal resources	Flexible – public cloud scales up when needed	Highly scalable across services
Security	Lower compared to private cloud	High (dedicated infrastructure)	Balanced – sensitive data in private, rest in public	Depends on vendor policies and orchestration
Customization	Low	High	Moderate – combines standard and custom options	Depends on vendor and integration
Cost Efficiency	Pay-per-use, cost-effective for SMEs	High capital and operational costs	Optimized cost by balancing private and public usage	Can be costly if not managed effectively
Use Case Examples	Startups, SMEs, web hosting, SaaS applications	Banks, government, enterprises with strict compliance	Enterprises needing flexibility with some sensitive workloads	Enterprises avoiding vendor lock-in, using best-in-class tools
Variants	N/A	On-Premise & Externally Hosted	Dynamic integration via proprietary software	Cloud bursting, federated clouds

III. Resource Allocation in Microservices

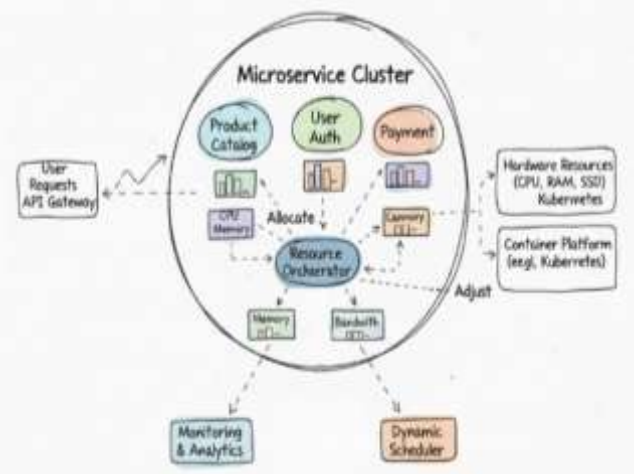

Fig. 6. Microservice Environment for Fine-Grained Resource Allocation

Fine-Grained Resource Allocation in Microservice, is the careful and dynamic allocation of computing resources (such as CPU [33], memory, bandwidth and so on) to each microservice or even smaller units (such as containers or function calls) according to their particular workload needs can also find in Fig. 6.

A. Monolithic Architecture

Monolithic software architecture is among the most ancient. This design aims to build an application with all the necessary elements [34]. Every component depends on the others and frequently cannot function or even compile.

- **Maintainability:** Monolithic architecture decreases maintainability since the components get increasingly tightly coupled [35]. Alterations in one module influence others making debug easier and updates challenging.
- **Flexibility:** Monolithic systems are not flexible, It is hard to adapt to new technologies or make architectural changes and often requires a lot of refactoring.

B. Microservices Architecture:

A distributed application using a microservices architecture has microservices in every module. Because each service in a microservices architecture (MA) has its own functionality, the application as a whole is not affected when one service fails. A service with loose coupling outwardly and high cohesion internally is created by encapsulating related modules in microservices [36], [37]. Many businesses, including Google, Amazon, IBM, and Netflix, switched from monolithic to microservices architectures for this reason.

- **Microservices:** Small enough to demonstrate a business necessity or function, microservices are functionally meaningful apps [38]. In addition to being independent during the service development or deployment stages, microservices are simple to include into third-party application systems, allowing for flexible and simple integration as well as automatic deployment [39].
- **Flexibility:** Maximum flexibility is offered by microservice-based applications, which are used to enable business logic operations or to control IoT networks and devices [40].
- **Components of Microservices:** each component like CPU, memory, bandwidth, containers, and even function-level are controlled with accuracy. The processing power, memory, and bandwidth allotted to a group of microservices is proportional to the workload of each individual service [41]. Isolation and scalability are isolated with containers, whereas narrower resources can be used with serverless execution [42] through components on a functional level. A mesosource like this means that these resources are coordinated and assigned in a dynamic way by a central authority to make sure that the orchestration of the microservices design works at its best.

IV. Advantages and Disadvantages of Microservices and Monolithic Architectures

A. Microservices

The benefits of microservices include scalability, flexibility, independent deployment, isolation of fault, diversity of technology and ease of maintenance. They however present drawbacks such as greater complexity, inter-service communication difficulty, greater operational overhead, and advanced monitoring, orchestration, and data consistency management among the distributed components.

Advantages: In Microservices are various types of advantages such as Flexibility, Easy development, optimized etc,

- **Flexibility and Scalability:** Microservices allow scaling individual components of the application accordingly [43], which also allows them to run services on resource-limited devices and effectively utilize resources.
- **Continuous and Easy Deployment:** MSA encourages the practice of continuous deployment and enables quick updates and bug fixes and addition of features resulting in more responsive and agile IoT apps.
- **Resource Optimization and Management:** MSA allows both the allocation and management of resources to be efficient hence leading to an optimized performance and the decrease in waste of computing resources.
- **Fault Tolerance and Anomaly Detection:** With MSA the fault of single services does not impact a complete application since all services are independent of each other which results in enhanced fault isolation and system resilience.

Disadvantages: Despite all the advantages of microservices there are also some disadvantages.

- **Increased System Complexity:** Microservices create many smaller services within applications that makes it more complex to manage, configure and coordinate many components across systems environments.
- **Challenging Inter-Service Communication:** The fact that microservices are networked makes it more complex and likely to result in latency or failures in the maintenance of reliable, secure, and efficient communication between services.
- **Difficulties in Maintaining Data Consistency:** Individual microservices commonly operate their database and it is more difficult to ensure transactional

consistency and synchronization of data throughout the system.

B. *Monolithic*

Monolithic design has a centralized administration system, is easy to develop, and has good performance due to fewer inter-service calls. Nonetheless, it also possesses such disadvantages as low scalability, component tightness, and lack of flexibility, which complicate the maintenance and updating process as the application expands.

Advantages: Just like Microservices there are also various advantages of monolithic architecture like easier debugging, testing and simple deployment etc.

- **Less cross-cutting concerns:** The application as a whole is affected by problems with handling, caching, logging, and performance monitoring. When it comes to monolithic applications, this feature is only applicable to one of them.
- **Easier debugging and testing:** It is much simpler to debug and test monolithic applications. In monolithic software, on the contrary, there is one item which cannot be segregated.
- **Simple to deploy:** Monolithic apps only require a single file or directory, eliminating the need for multiple deployments.
- **Simple to develop:** Any competent technical group can create a monolithic app so long as it follows the standard practice for app development.

Disadvantages: With the advantages there also come some disadvantages in monolithic architecture.

- **Modification:** A monolithic application's modification or update procedure is complex and dangerous [44]. Any adjustments could have far-reaching, unforeseen effects that would disrupt the system.
- **Collaboration:** The monolithic architecture also impacts the collaboration and the rate of developing speed, which is worth mentioning. Monolithic codebases are big and therefore are cumbersome, and as such, collaborating between development teams becomes hard.

V. Literature Review

The literature review provides the analysis of the new techniques in cloud, edge, and microservice-based systems, emphasizing such techniques as genetic algorithms, graph clustering, and microservice orchestration, which address the issues of resources allocation and architectural complexity, as well as system scalability and provide an opportunity to deploy services more effectively and to provide better system performance.

Lian et al. (2025) came up with a new resource allocation algorithm (GARA) of GDML in fgOTN. GARA takes into account both the completion of tasks and the bandwidth modification using the population generation in accordance with the previous knowledge and adaptive mutation in accordance with the ratio of accomplishments. Analysis of simulation shows that GARA is able to prioritize resource allocation to high priority tasks so that there is no resource competition and the task completion ratio is maximum and that the network reconfigurations are not high [45].

Li, Fu and Ding (2025) proposed a pre-selection of servers stage, where the users can select the edge servers adaptively in regard to their costs depending on location. An intra-user resource reallocation strategy is also proposed to aggregate user tasks and reduce allocation complexity. Simulation results show that Q-ERA matches the classical VCG in social welfare but with significantly lower execution time. In various situations, it results in the maximization of social welfare subject to a small cost of computation compared to other the considered methods [46].

Mo et al. (2024) developed an intelligent collaboration architecture in the cloud-edge. Based on the idea of micro-service, and decouple the function layers of the Artificial Intelligence plane into six network functions (NFs) on a four-layer, three-plane network architecture. Then, orchestrating Cloud-Edge NFs with Kubernetes to provide customized services, and monitoring and managing the NFs in real time throughout their respective lifecycles [47].

Zhu and Cai (2024) suggested a microservice architecture service partitioning. Briefly, the extraction of activity instances is realized based on Domain-Driven Design (DDD). Semantic information templates are constructed to realize the formal semantic information entry of activity instances. Finally, low-coupling and high-cohesion service partitioning is realized based on the Louvain graph clustering method [48]

Munjal, Gangodkar and Lohumi (2023) suggests a new design and approach that could solve the problems with monolithic apps. Microservices and cloud computing are brought together in the proposed architecture to facilitate application development and deployment. The primary findings of this study are the benefits of using microservices architecture and the obstacles that must be overcome during cloud deployment of microservices [49].

Zou et al. (2023) introduced a search tree-based search tree-based fine-grained access control. This feature offers protection of resources on the element level and also speeds up policy search with the help of an index tree, which minimizes the time overhead of the fine-grained access control system. Experimentally, it is shown that this mechanism provides element-level protection of resources whilst incurring low-performance overhead [50].

Wang et al. (2022) delivered a service monitoring solution for cloud computing, letting administrators and users optimize cloud computing resources according to evolving business needs. Initially, the OpenStack cloud monitoring system's primary features are presented, which primarily encompass data collecting, processing, analysis, presentation, and alert notification. Second, applications like OpenStack-exporter, Libvirt, Ceph-exporter, and Grafana make up the bulk of the system [51].

Table II provides recently conducted research on cloud, edge, Resource Allocation, and microservice architectures, including their methods, major findings, problems, and perspectives. The most common are scalability, complexity of orchestrations, and overheads in the system, and the future trends revolve around automation, greater efficiency, and wider deployment in real-world environments.

TABLE II. OVERVIEW OF RECENT STUDIES ON FINED-GRAINED IN MICROSERVICES

Reference	Study On	Approach	Key Findings	Challenges / Limitations	Future Directions
Lian et al. (2025)	Resource allocation of GDML in fgOTN	Resource Allocation (GARA) with genetic Algorithms that utilize previous knowledge-based generation of population and adaptive mutation	Profiles best task completion ratio, priority tasks, less resource competition, less reconfiguring network	Large networks need high computation with May; requires correct estimation of completion ratio	Scalability improvement, real-time feed-back of networks, testing on actual network edges
Li, Fu & Ding (2025)	Reactive edge server choice and redistribution of resources	Q-ERA the pre-selection of a server according to the location dependent cost and intra-user resource redistribution	Maximizes VCG social welfare at a lower execution time; gives maximum social welfare in various situations with minimal computer cost	Performance is dependent on proper estimation of location costs; can experience a decline in performance when there are extreme users	Scaling to multi-user collaborative case, hybridizing dynamic pricing models
Mo et al. (2024)	Smart-edge intelligent collaboration architecture	Microservice based architecture in four layers, three plane network with six NFs; NF coordination through Kubernetes	POffers customizable AI-driven services, real-time monitoring and lifetime management	Complex orchestration; relies on the Kubernetes performance under the heterogeneous environment	Improve the efficiency of orchestration, introduce self-management capabilities, introduce autonomy of scaling
Zhu & Cai (2024)	service partitioning by microservice architecture	DDD based activity extraction; semantic template construction; Louvain graph clustering of service partitions that are low-coupling	Brings about low coupling and high cohesion to service partitioning; enhances modularity and scale	Semantic extraction is the area that needs domain experts, graph clustering can be inefficient with very large systems	Automated semantic extraction through NLP; optimization of clustering in great microservice ecosystems
Munjal, Gangodkar & Lohumi (2023)	Cloud–microservice fusion architecture	Proposal of a microservices-cloud combined architecture for scalable application deployment	Highlights advantages of microservices and identifies cloud deployment challenges; improves flexibility and modularity	Migration from monolithic to microservice architecture remains complex; handling distributed failures is difficult	Develop automated migration tools; improve resilience and fault-tolerance mechanisms
Zou et al. (2023)	Fine-grained access control	Index tree search-tree based access control (policy search acceleration)	Enforcement of element level resource protection and low performance overhead; better policy retrieval	Index maintenance overhead; suffer when workloads are large in terms of policy updates	Optimize index structures; integrate adaptive policy caching mechanisms.
Wang et al. (2022)	Monitoring of open stack cloud services	Monitoring framework with OpenStack-exporter, Libvirt, Ceph-exporter, Grafana.	Facilitates the cloud resources optimization according to business requirements; offers real-time data gathering, processing	High reliance on OpenStack ecosystem; might have to be changed to high-traffic systems	Monitor hybrid/multi-cloud, include predictive analytics as a way of optimizing proactively

VI. CONCLUSION AND FUTURE WORK

Fine Grained Resource Allocation is one of the core developments in the management of the computational resources in the cloud-based applications. The use of microservices and containerization has resulted in very dynamic and modular systems, which need more subtle approaches to resource allocation. FGRA meets these requirements by enabling resources to be allocated precisely and flexibly to enhance the performance, scalability, as well as the overall system efficiency. It goes on to demonstrate that although there have been major strides in the orchestration of resources on a microservice basis, very few solutions currently have the capability to completely optimize through the various cloud deployment models. This study highlights the relevance of matching FGRA strategies and service-level objectives (SLOs), workload variability, and real-time supervision, which is important in the sphere of operational excellence in a contemporary cloud infrastructure. In addition, since services grow fast due to the demand of users, the capability to handle resource consumption on a fine-grained basis becomes critical in preventing bottlenecks and minimizing the cost of operation.

Future investigations ought to concentrate on creating more autonomous FGRA designs that are able to learn real-time workload trends and become dynamically adjusted to a heterogeneous workload cloud-edge environment. The combination of AI-based decision-making, predictive monitoring, and self-scaling will be critical to enhance accuracy and minimize overhead levels of operation. Moreover, the FGRA solutions of the future should be able to solve the issues associated with the complexity of orchestration, multi-cloud interoperability, and data consistency and allow the smooth deployment.

REFERENCES

[1] W. Li, X. Li, L. Chen, and M. Wang, “Microservice Workflow Scheduling with a Resource Configuration Model Under Deadline and Reliability Constraints,” *Sensors*, vol. 25, no. 4, 2025, doi: 10.3390/s25041253.

[2] F. Alqahtani, M. Almutairi, and F. T. Sheldon, “Cloud Security Using Fine-Grained Efficient Information Flow Tracking,” *Futur. Internet*, vol. 16, no. 4, 2024, doi: 10.3390/fi16040110.

[3] S. Thangavel, K. Narukulla, and R. Sundaram, “Edge-Enabled Distributed Computing for Low-Latency IoT Applications: Architectures, Challenges, and Future Directions,” *Int. J. Emerg. Res. Eng. Technol.*, vol. 3, no. 1, pp. 28–41, 2022, doi: 10.63282/3050-922x.ijeret-v3i1p104.

[4] N. K. Prajapati, “Cloud-based serverless architectures: Trends, challenges and opportunities for modern applications,” *World J. Adv. Eng. Technol. Sci.*, vol. 16, no. 1, pp. 427–435, Jul. 2025, doi: 10.30574/wjaets.2025.16.1.1225.

[5] M. R. R. Deva, “Advancing Industry 4.0 with Cloud-Integrated Cyber-Physical Systems for Optimizing Remote Additive Manufacturing Landscape,” in *2025 IEEE North-East India International Energy Conversion Conference and Exhibition (NE-IECCE)*, IEEE, Jul. 2025, pp. 1–6. doi: 10.1109/NE-IECCE64154.2025.11182940.

[6] S. P. Bheri and G. Modalavalasa, “Advancements in Cloud Computing for Scalable Web Development: Security Challenges and Performance Optimization,” *JCT Publ.*, vol. 13, no. 12, pp. 01–

07, 2024.

[7] M. Dawood, S. Tu, C. Xiao, H. Alasmary, M. Waqas, and S. U. Rehman, "Cyberattacks and Security of Cloud Computing: A Complete Guideline," *Symmetry (Basel).*, vol. 15, no. 11, 2023, doi: 10.3390/sym15111981.

[8] M. Chauhan and S. Shiaeles, "An Analysis of Cloud Security Frameworks, Problems and Proposed Solutions," *Network*, vol. 3, no. 3, pp. 422–450, 2023, doi: 10.3390/network3030018.

[9] M. Menghnani, "Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications," *Int. J. Innov. Sci. Res. Technol.*, vol. 10, no. 2, 2025, doi: 10.5281/zenodo.14959407.

[10] V. Bushong *et al.*, "On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study," *Appl. Sci.*, vol. 11, no. 17, 2021, doi: 10.3390/app11177856.

[11] D. K. Pandiya, "Performance Analysis of Microservices Architecture in Cloud Environments," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 10, no. 12, pp. 264–274, 2022.

[12] N. Raveendran, "Fine-Grained Indexing Strategies for Social Feeds," *J. Eng. Comput. Sci.*, vol. 4, no. 7, pp. 157–164, 2025.

[13] J. E. Kofi, "Monitoring Cloud Performance Metrics Utilizing AI to Estimate the Efficiency of Cloud Operations," in *2025 7th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/ISAECT68904.2025.11318827.

[14] M. Zhou, X. Dong, H. Chen, and X. Zhang, "Fine-grained scheduling in multi-resource clusters," *J. Supercomput.*, vol. 76, no. 3, pp. 1931–1958, 2020, doi: 10.1007/s11227-018-2505-4.

[15] A. Anwar, H. Anwar, and S. Anwar, "Towards Low-Cost Classification for Novel Fine-Grained Datasets," *Electronics*, vol. 11, no. 17, 2022, doi: 10.3390/electronics11172701.

[16] A. Narang, "A Comprehensive Review of Cloud Environments and Their Comparative Analysis," no. 2, pp. 158–162, 2025.

[17] B. R. Ande, "Enhancing Cloud-Native AEM Deployments Using Kubernetes and Azure DevOps," *Int. J. Commun. Networks Inf. Secur.*, vol. 15, no. 8, 2023.

[18] B. K. Rani, B. P. Rani, and A. V. Babu, "Cloud Computing and Inter-Clouds – Types, Topologies and Research Issues," *Procedia Comput. Sci.*, vol. 50, pp. 24–29, 2015, doi: 10.1016/j.procs.2015.04.006.

[19] M. N. Tiwari, "Perspective study of Public Cloud: A highly scalable Cloud deployment model." Sep. 13, 2022. doi: 10.36227/techrxiv.21075805.

[20] S. Amrale, "Proactive Resource Utilization Prediction for Scalable Cloud Systems with Machine Learning," *Int. J. Res. Anal. Rev.*, vol. 10, no. 4, pp. 758–764, 2023.

[21] V. Varma, "Secure Cloud Computing with Machine Learning and Data Analytics for Business Optimization," *ESP J. Eng. Technol. Adv.*, vol. 4, no. 3, 2024, doi: 10.56472/25832646/JETA-V4I3P119.

[22] M. Amini, N. S. Safavi, M. Dashti, and A. Abdollahzadegan, "Type Of Cloud Computing (Public And Private) That Transform The Organization More Effectively," *SSRN Electron. J.*, vol. 2, pp. 1263–1269, 2013.

[23] G. Sivi and T. Narayanan, "A Review on Matching Public, Private, and Hybrid Cloud Computing Options," *Int. J. Comput. Sci. Inf. Technol. Res.*, vol. 2, no. 2, pp. 213–216, 2014.

[24] A. Syed and M. I. Ahmad, "Advanced Data Collection Techniques in Cloud Security: A Multi-Modal Deep Learning Autoencoder Approach," *ArXiv*, pp. 13, Nov, 2025, doi: 10.20944/preprints202510.0767.v1.

[25] V. Shah, "Securing the Cloud of Things: A Comprehensive Analytics of Architecture, Use Cases, and Privacy Risks," *ESP J. Eng. Technol. Adv.*, vol. 3, no. 4, pp. 158–165, 2023, doi: 10.56472/25832646/JETA-V3I8P118.

[26] W. A. Aziz, E. Babulak, and D. Al-Dabass, "Network Function Virtualization over Cloud-Cloud Computing as Business Continuity Solution," in *Digital Service Platforms*, IntechOpen, 2021, pp. 1–40. doi: 10.5772/intechopen.97369.

[27] D. R. Patil, C. S. Arage, and P. S. Gaikwad, "Comprehensive Study on Deployment Models and Service Models in Cloud Computing," *Int. Res. J. Eng. Technol.*, vol. 9, no. 12, pp. 182–186, 2022.

[28] M. Deb and A. Choudhury, "Hybrid Cloud: A New Paradigm in Cloud Computing," in *Machine Learning Techniques and Analytics for Cloud Security*, Wiley, 2021, pp. 1–23. doi: 10.1002/9781119764113.ch1.

[29] A. Parupalli and H. Kali, "An In-Depth Review of Cost Optimization Tactics in Multi-Cloud Frameworks," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 3, no. 5, pp. 1043–1052, Jun. 2023, doi: 10.48175/IJARSCT-11937Q.

[30] S. Katangoori and A. Katangoori, "AI-Augmented Data Governance: Enabling Intelligent Access, Lineage, and Compliance Across Hybrid Clouds," *Am. J. Auton. Syst. Robot. Eng.*, vol. 1, pp. 716–738, November, 2021.

[31] R. Islam *et al.*, "The Future of Cloud Computing: Benefits and Challenges," *Int. J. Commun. Netw. Syst. Sci.*, vol. 16, no. 04, pp. 53–65, 2023, doi: 10.4236/ijcns.2023.164004.

[32] A. J. Ferrer, D. G. Pérez, and R. S. González, "Multi-cloud Platform-as-a-service Model, Functionalities and Approaches," *Procedia Comput. Sci.*, vol. 97, 2016, doi: 10.1016/j.procs.2016.08.281.

[33] G. Maddali, "An Efficient Bio-Inspired Optimization Framework for Scalable Task Scheduling in Cloud Computing Environments," *Int. J. Curr. Eng. Technol.*, vol. 15, no. 3, pp. 229–238, 2025.

[34] N. S. Elgheriani and N. D. A. S. Ahme, "Microservices Vs. Monolithic Architectures [The Differential Structure Between Two Architectures]," *MINAR Int. J. Appl. Sci. Technol.*, vol. 4, no. 3, pp. 500–514, Sep. 2022, doi: 10.47832/2717-8234.12.47.

[35] A. A. Padvekar and V. B. Gupta, "Comparative Analysis of Monolithic vs. Distributed Architecture.," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 4, no. 3, pp. 433–442, Jun. 2024, doi: 10.48175/IJARSCT-18946.

[36] R. Tandon and D. Patel, "Evolution of Microservices Patterns for Designing HyperScalable Cloud-Native Architectures," *ESP J. Eng. Technol. Adv.*, vol. 1, no. 1, pp. 288–297, 2021, doi: 10.56472/25832646/JETA-V1I1P131.

[37] M. Seedat, Q. Abbas, and N. Ahmad, "Systematic Mapping of Monolithic Applications to Microservices Architecture," pp. 1–24, 2023.

[38] D. C. Li, B.-H. Chen, C.-W. Tseng, and L.-D. Chou, "A Novel Genetic Service Function Deployment Management Platform for Edge Computing," *Mob. Inf. Syst.*, vol. 2020, no. 1, pp. 1–22, Oct. 2020, doi: 10.1155/2020/8830294.

[39] S. Grover and S. K. Das, "Flaky test automation and mitigating test crashes in agile releases," *Int J Comput Exp Sci Eng*, vol. 11, no. 3, 2025.

[40] M. Driss, D. Hasan, W. Boulila, and J. Ahmad, "Microservices in IoT Security: Current Solutions, Research Challenges, and Future Directions," *Procedia Comput. Sci.*, vol. 192, pp. 2385–2395, 2021, doi: 10.1016/j.procs.2021.09.007.

[41] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices," 2020, doi: 10.48550/arXiv.2008.08509.

[42] S. Narang and V. G. Kolla, "Next-Generation Cloud Security: A Review of the Constraints and Strategies in Serverless Computing," *Int. J. Res. Anal. Rev.*, vol. 12, no. 3, pp. 1–7, 2025, doi: 10.56975/ijrar.v12i3.319048.

[43] A. El Akhdar *et al.*, "Exploring the Potential of Microservices in Internet of Things: A Systematic Review of Security and Prospects," *Sensors*, vol. 24, no. 20, 2024, doi: 10.3390/s24206771.

[44] S. Mooghala, "A Comprehensive Study of the Transition from Monolithic to Micro services-Based Software Architectures," *J. Technol. Syst.*, vol. 5, no. 2, pp. 27–40, Nov. 2023, doi: 10.47941/jts.1538.

[45] M. Lian *et al.*, "Resource Allocation in Flexible-Bandwidth Fine-Grained Optical Transport Networks for Geo-Distributed Machine Learning," *IEEE Internet Things J.*, vol. 12, no. 13, pp. 25601–25619, 2025, doi: 10.1109/JIOT.2025.3558933.

[46] Y. Li, X. Fu, and J. Ding, "Incentive Compatible and Efficient Resource Allocation in Mobile Edge Computing with Quantized Resource Auctions," in *2025 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2025, pp. 460–467. doi: 10.1109/ISPA67752.2025.00067.

[47] W. Mo, J. Zheng, C. Zhang, and L. Xiong, "Microservices-Based Design of Cloud-Edge Collaboration Architecture," in *2024 International Conference on Future Communications and Networks (FCN)*, 2024, pp. 1–6. doi: 10.1109/FCN64323.2024.10985716.

[48] H. Zhu and H. Cai, "Low-Coupling and High-Cohesion Microservice Partitioning Method to Support Rapid Development of Steel Industry Management Systems," in *2024 6th International Conference on Computer Communication and the Internet (ICCCI)*, 2024, pp. 78–82. doi: 10.1109/ICCCI62159.2024.10674297.

[49] P. Munjal, D. Gangodkar, and Y. Lohumi, "Microservices based Ticket Booking Platform deployed on Cloud," in *2023 Second International Conference on Informatics (ICI)*, 2023, pp. 1–5. doi: 10.1109/ICI60088.2023.10420939.

[50] X. Zou, C. Zheng, H. Lin, L. Du, W. Xu, and C. He, "A Fine-Grained Access Control Mechanism Based on Search Trees," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2023, pp. 1614–1620. doi: 10.1109/TrustCom60117.2023.00220.

[51] H. Wang, X. Zhang, Z. Ma, L. Li, and J. Gao, "An Microservices-Based OpenStack Monitoring System," in *2022 11th International Conference on Educational and Information Technology (ICEIT)*, 2022, pp. 232–236. doi: 10.1109/ICEIT54416.2022.9690713.

