Journal of Information Systems Engineering and Management
2025, 10(21s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Building Cost-Effective Cloud-Native Applications A Comparison of On-Premises and Cloud Deployment with Cost Optimization Strategies

[1] Dr. I. KALA, Priyam Pandia[2], Dr. R. Manimegalai[3], Karthik Gokare Somashekhar[4], Srivatsan Santhanam[5]

[1]*Associate Professor, Department of Computer Science and Engineering, PSG Institute of Technology and Applied Research, Coimbatore, India. ootykala@gmail.com*

[2]*Principal Architect SAP Concur, SAP Labs India Pvt. Ltd, Bangalore, India. priyam.pandia@sap.com*

[3]*Professor, Department of Computer Science and Engineering, PSG Institute of Technology and Applied Research, Coimbatore, India. drrm@psgitech.ac.in*

[4]*Engineering Manager, SAP Concur Spend Engineering, SAP Labs India Pvt. Ltd, Bangalore, India. karthik.gokare.somashekhar@sap.com*

[5]*Vice president Spend Engineering, Concur R&D, SAP Labs India Pvt. Ltd. Bangalore, India. srivatsan.santhanam@sap.com*

ARTICLE INFO

Received: 16 Dec 2024
Revised: 01 Feb 2025
Accepted: 16 Feb 2025

ABSTRACT

Modern software development demands a keen eye on cost-effectiveness. This requires careful consideration of three key aspects: cost, service quality, and performance. While on-premises development offers customization and control, cloud-based solutions provide superior scalability and flexibility. However, this necessitates robust disaster recovery and backup plans, as well as performance and cost monitoring to ensure efficient service delivery. To minimize costs and maximize performance, businesses should prioritize quality-of-service as well as effective design practices and leveraging established and modern implementation techniques. This includes techniques like code pattern implementation, parallel threads and concurrency utilization, and meticulous memory management. Ultimately, achieving cost-effective development connects on a holistic approach that strikes a balance between cost and service quality. This paper explores optimizing resource allocation, selecting suitable pricing models, and implementing performance monitoring to ensure adherence to SLAs while maintaining cost-efficiency.

Keywords: Cost-effective development, Cloud-native development, Cloud-based solutions, SLAs, Disaster recovery, Cost monitoring, Quality of service, Performance monitoring.

INTRODUCTION

This section provides a overview of cost-efficient cloud-native development, observability, cost management, resource provisioning, Software as a Service (SaaS), and disaster recovery.

A. Developing cost-efficient cloud-native applications

Key decisions are crucial for ensuring cost-effectiveness in Cloud-native Development. Various metrics play a significant role in achieving this goal, as illustrated in Figure 1. With multiple cloud providers available, each with its pricing structure, selecting a suitable provider offering cost-effective services is essential. Utilizing serverless computing enables running applications without dedicated infrastructure, leading to cost reduction by paying only for the utilized computing resources[28]. Containerization involves bundling the application and its dependencies into a single container for deployment across different cloud providers, thereby lowering infrastructure expenses.

On-premise and Cloud-native development represent distinct approaches to software development, each requiring consideration based on the business context. When evaluating these paradigms, cost serves as a primary factor, revealing numerous differences through a comparative analysis. A study focused on cost as the primary criterion and adjusting other factors is conducted, as depicted in Figure 1.

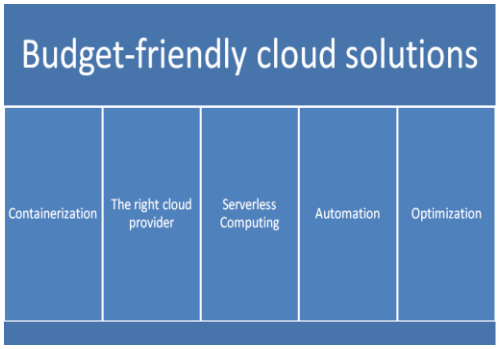

Figure 1. Metrics assessed for cost-effective cloud-native development

Automation helps in cutting down development and deployment expenses by automating repetitive tasks like testing, deployment, and monitoring. Application optimization for the cloud can lead to decreased infrastructure costs. For instance, employing caching and content delivery networks for data transfer between the application and users, along with utilizing open-source tools. Kubernetes, an open-source container orchestration tool, assists in managing containerized applications.

B. Observability

Observability plays a crucial role in cost-effective cloud-native development by offering visibility into the application's performance in production. This visibility aids in promptly identifying and addressing issues, thereby minimizing downtime and the expenses linked to issue resolution. Proactive issue detection and observability enable real-time monitoring of the application's health to prevent critical issues, ultimately reducing costly downtime and the need for reactive troubleshooting.

Regarding resource optimization, observability tools provide insights into the application's resource utilization, such as CPU, memory, and network bandwidth. Leveraging this information, the application can be optimized to use resources more efficiently, leading to reduced infrastructure costs. Observability also facilitates monitoring the application's performance and availability to enhance the customer experience. By identifying and resolving issues that impact customer satisfaction, observability helps mitigate the risk of customer churn and related expenses[29].

C. Critical relationship: cost and resource provisioning.

A significant challenge in cost-effective cloud-native development is the correlation between cost and resource provisioning, which can lead to cost inflation if not handled carefully. Over-provisioning resources can result in unnecessary expenses, while under-provisioning can cause performance issues and increased downtime. To mitigate costs and prevent resource inflation, optimizing resource usage is essential. This includes monitoring resource utilization and adjusting resource allocation based on actual usage patterns, such as implementing auto-scaling to add or remove resources based on demand[20]. Efficient resource allocation strategies like utilizing reserved instances or spot instances help avoid inflated costs. Reserved instances offer cost savings compared to on-demand instances, while spot instances can yield substantial savings for non-critical workloads. Various cost management tools are available to assist in managing cloud resource costs by providing real-time cost tracking, alerts, optimization recommendations, and forecasting for effective cost planning and management[22].

D. Subscription Models in SaaS

SaaS providers present customers with a variety of subscription options, ranging from basic to premium plans, enabling them to pay solely for the features and services they require. Figure 2 demonstrates the advantages of utilizing subscription models in SaaS development.

Subscription models offer SaaS providers a consistent and predictable revenue stream, facilitating the planning of investments and future development endeavors. This model allows providers to scale their offerings effortlessly in response to demand fluctuations. Compared to traditional software licensing models[21], subscription models are typically more cost-effective for customers because they only pay for the specific features they require and can readily adjust their subscription levels as their needs evolve[23]. Additionally, SaaS providers can introduce new features and updates more swiftly than traditional software providers by leveraging automatic cloud-based delivery mechanisms[24].

In the development of a cloud-native application utilizing subscription models, it is crucial to adhere to key best practices. Firstly, offering a diverse range of subscription plans tailored to cater to the varying needs and budget limitations of different customers is essential. Secondly, providing a complimentary trial period enables customers to evaluate the product before committing to a recurring subscription, whether on a monthly, quarterly, or other periodic basis. Additionally, incorporating self-service functionalities for customers to autonomously manage their subscriptions, including the flexibility to upgrade or downgrade their plan, ensures that the service aligns with evolving customer requirements and delivers value for their investment[26].

Figure 2. Subscription Model-SaaS

E. Disaster Recovery

Disaster Recovery (DR) plays a vital role in cloud-native development by ensuring the swift recovery of applications and data in unforeseen disasters. Adequate resource provisioning is essential for effective disaster recovery, ensuring that the required resources are accessible to restore impacted services[27]. Service Level Agreement (SLA) metrics establish the maximum acceptable downtime and data loss for applications, guiding the resource provisioning approach. Identifying and prioritizing critical workloads and data for protection is integral to the DR resource provisioning strategy. Automation of DR failover and fallback processes minimizes downtime and ensures the rapid restoration of essential services. Implementing multi-region redundancy guarantees that critical applications and data are replicated across diverse geographical locations, reducing the likelihood of data loss and downtime. Optimizing resource utilization through cloud-native services allows for scaling resources based on demand, preventing over-provisioning and lowering costs[25].

Regular testing of the Disaster Recovery (DR) plan is essential to verify its effectiveness and ensure that the resource provisioning strategy aligns with Service Level Agreements (SLAs). This validation process can be facilitated by utilizing tools like AWS Cloud Formation, which enables the creation and testing of DR scenarios within a controlled sandbox environment. By conducting routine tests using tools such as AWS Cloud Formation, organizations can validate the efficiency of their DR plan and confirm that the resource provisioning strategy complies with SLAs.

LITERATURE SURVEY

Several research studies and literature reviews have explored the application of optimization algorithms in achieving cost-effective cloud-native development, as detailed in this section.

A. Evolutionary Optimization Algorithms utilized for cloud-native development

Genetic Algorithms (GA) are commonly employed for enhancing resource allocation in cloud computing settings. For instance, a study referenced as [1] utilized GA to optimize the positioning of virtual machines within a cloud environment. The research indicated that GA-based strategies effectively manage resource utilization, decrease response time, and cut down costs. Particle Swarm Optimization (PSO) has demonstrated proficiency in reducing response time and enhancing resource utilization. In a separate study denoted as [2], PSO was applied to streamline the placement of containers in a cloud-native setup, aiming to lower the overall expenses related to resource allocation. Another investigation referenced as [3] utilized Ant Colony Optimization (ACO) to fine-tune the deployment of microservices within a cloud-native environment. Additionally, researchers in [10] leveraged Simulated Annealing (SA) to optimize container placement in a container orchestration system. Furthermore, Reinforcement Learning (RL) algorithms can acquire optimal resource allocation strategies through interactions with the cloud environment [11].

B. Optimization across different layers of the cloud

Efficient cloud-native development requires optimizing the infrastructure, platform, and software layers of cloud computing. Prior research has thoroughly explored optimization methods for each of these specific layers.

1) Infrastructure Layer Optimization

The infrastructure layer of cloud computing encompasses the physical components like servers, storage, and networking equipment. Several research studies have delved into optimization strategies for this layer, including techniques such as energy-efficient resource allocation and virtual machine positioning [4]. Zaman et al. introduced a methodology aimed at enhancing resource allocation and virtual machine deployment in cloud data centers [4]. This approach takes into account the workload and resource consumption of individual applications, dynamically adjusting resource allocation to enhance energy efficiency. Additionally, cost optimization for cloud infrastructure utilizing evolutionary algorithms is proposed in [5]. This method considers the dynamic nature of workload and resource utilization in cloud applications, employing a multi-objective evolutionary algorithm to optimize resource allocation and reduce costs.

Yang et al. have introduced a combined genetic algorithm and particle swarm optimization (PSO) algorithm to address optimization challenges [6]. Their proposal focuses on optimizing resource allocation and load balancing in edge computing. The method takes into consideration the diverse computing resources present at the edge and employs a reinforcement learning algorithm to enhance task allocation across various edge devices. The authors have suggested a multi-agent reinforcement learning algorithm that factors in the workload of edge devices, resource availability, and network conditions to effectively distribute tasks and maintain load balance among the edge devices.

2) Platform Layer Optimization

The platform layer of cloud computing serves as a runtime environment for deploying and overseeing cloud applications. A study detailed in [7] has introduced an optimization technique for scheduling containers and allocating resources within a cloud computing setting. This method takes into account the changing workload and resource needs of containerized applications, utilizing a genetic algorithm to enhance resource allocation and enhance the overall performance and efficiency of the system.

Yin et al. have introduced a method for dynamically scaling applications and allocating resources in cloud computing environments[8]. Through an assessment conducted in a simulated cloud computing environment, the researchers demonstrated that their approach could enhance the performance and efficiency of containerized applications while also lowering energy consumption. Their proposal includes a reinforcement learning algorithm that takes into consideration the workload and resource usage of applications to determine the most effective scaling and resource allocation strategies.

Li et al. have introduced a method for optimizing service placement and routing for microservices [9]. Their approach takes into account the interdependencies among microservices and the network structure of the system, utilizing a genetic algorithm to enhance the placement and routing of microservices for improved system performance and efficiency. The researchers have suggested a multi-objective genetic algorithm that considers factors like network latency, service availability, and resource utilization to identify the most suitable placement and routing strategies for microservices.

3) Software Layer Optimization

The software layer of cloud computing encompasses application software and data services, with optimization strategies focusing on cost-effective data storage. Alex and colleagues have introduced a method for cost-effective data storage and retrieval in cloud databases [12]. Their approach takes into consideration data access patterns and storage costs across various storage types, employing a hybrid storage scheme to enhance system performance and cost efficiency. Additionally, [13] discusses query optimization and workload balancing in cloud data processing, while [14] delves into optimizing machine learning algorithms for cloud-based predictive analytics.

C. Utilizing Queuing Theory and Stochastic Models for Enhancing Cost Efficiency in Cloud Computing.

Previous studies have investigated a variety of queuing models and stochastic models to improve different elements of cloud computing, such as resource allocation, workload scheduling, and system performance enhancement.

Queuing Theory : Queuing theory serves as a mathematical framework that characterizes the dynamics of queues or waiting lines. Within the realm of cloud computing, queuing theory has been applied to depict the behavior of cloud systems, encompassing server queues, job queues, and data queues. Research conducted in [15] leveraged queuing theory to enhance resource allocation and scheduling within a cloud environment. The study revealed that queuing models play a pivotal role in achieving a harmonious balance in resource utilization and cost minimization.

Stochastic Models : Stochastic models are probabilistic representations designed to encapsulate the inherent randomness or uncertainty within a system. In the realm of cloud computing, these models have been instrumental in characterizing different facets of cloud systems, including workload arrivals, resource demands, and system performance. Research cited in [16] has applied stochastic models to enhance job scheduling within cloud environments, revealing their capacity to efficiently minimize the overall cost associated with scheduling tasks.

Hybrid Models : Hybrid models amalgamate queuing theory and stochastic models to encompass both deterministic and probabilistic elements within a system. Within the domain of cloud computing, these hybrid models have been instrumental in representing diverse aspects of cloud systems, including server power consumption, workload variations, and system performance. Research highlighted in [17] has used a hybrid model to refine resource allocation and scheduling within a cloud environment, demonstrating its efficacy in achieving a balanced resource utilization, decreased response times, and minimized costs.

The current research has extensively investigated the utilization of queuing theory and stochastic models to enhance cost optimization in cloud computing, encompassing the development of cost-effective cloud-native solutions. These modeling approaches prove beneficial in optimizing different aspect of cloud systems, such as resource allocation, workload scheduling, and overall system performance, thereby facilitating cost reduction while enhancing system efficiency.

COST EFFECTIVE DEVELOPMENT ARCHITECTURE

This section discusses key elements for developing cloud applications that are cost-effective. These include keeping an eye on response times and how much work the system can handle, using logs to monitor when resources are being used heavily, efficiently managing resources to keep costs down, adjusting resource quotas to avoid overspending, and using on-demand resources when needed to prevent under-provisioning.

When aiming for cost-effective development architecture, it's important to make smart choices. Selecting the appropriate technology stack can greatly affect development expenses. It's crucial to pick a technology stack that not only fits the project requirements but also keeps costs in check. Using open-source technologies and frameworks can be a good option to save on both licensing and development expenses. Embracing agile development methods can also cut down costs by promoting teamwork, gradual development, and ongoing feedback. This helps prevent costly mistakes and ensures that the final product meets customer expectations.

A cost-effective development architecture diagram typically includes various components involved in a software development project and their relationship to each other. Figure 3. illustrates a basic cost-effective development architecture diagram.

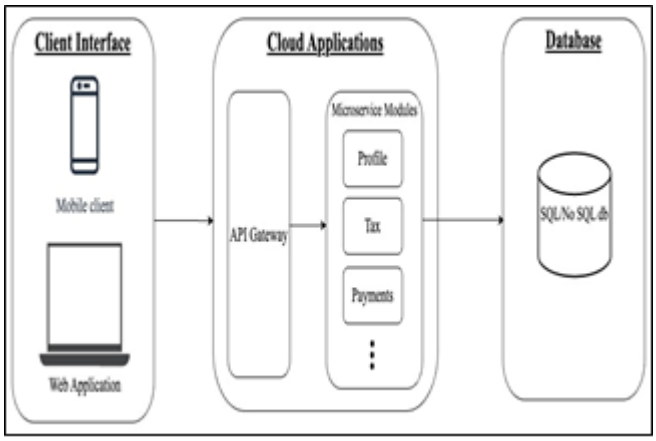

Figure 3. Cost-Effective Development Architecture

Users access cloud-native applications through a client interface on devices like mobile phones or desktop computers, using a graphical user interface (GUI). These interfaces connect to an API gateway, which acts as a middle layer between the front end and the applications. The API gateway contains the necessary APIs with data relevant to the user interface, allowing users to view information. Additionally, the gateways can function as reverse proxies to balance the incoming load from user interactions. API calls received at the gateway are then directed to the appropriate microservice. The architecture outlined in Figure 3 emphasizes separating microservices by domain, promoting a loosely coupled design. Each microservice accesses its own database, which may be either relational or

NoSQL, depending on the business requirements. When deployed efficiently, this architecture can significantly impact cost planning while maintaining other key performance indicators such as scalability and throughput.

D. Effective Architecture - Integration challenges, Database I/Os, Resource Reuse

Within a cost-effective development architecture, handling the complexities of integration requires utilizing economically efficient integration technologies, like RESTful APIs. It's crucial to make sure that the development team has the necessary skills for effective integration.

As data volumes increase, it becomes crucial to optimize database schema and queries. Advanced methods such as denormalization and caching can help reduce the resource-heavy nature of database I/O operations. Enabling developers to reuse code and services relies on carefully designing a modular architecture, often seen in microservices. Promoting a culture of code sharing and collaboration among developers is essential for encouraging cost-effective development practices.

E. Monitoring response times and throughput

Keeping a close watch on response times and throughput is crucial for ensuring the optimal operation of a cost-effective development architecture. Monitoring these aspects provides valuable insights into potential performance bottlenecks within the architecture. Various tools, such as Application Performance Monitoring (APM) solutions, log analysis tools, and Real-User Monitoring (RUM) tools, assist in this effort by revealing how well the application is functioning and identifying any issues that could slow down response times.

F. Observability Logs for higher resource usage RCA

In cost-effective development architectures, observability logs are essential in identifying anomalies in resource utilization. They provide information about CPU, memory, and disk usage, enabling early detection of problems such as memory leaks or excessive CPU usage. Resource monitoring relies on tools like APM solutions, log analyzers, and infrastructure monitors. These tools send alerts when resource consumption surpasses predefined thresholds, enabling proactive measures. Root Cause Analysis (RCA) for resource issues involves several steps, including data collection and analysis, identifying the cause, refining precision, taking corrective action, and ongoing monitoring.

G. Cost efficient resource provisioning , Inflated quota, On-spot/ on-demand, Under provisions, Effect on the cost

Cost-effective development relies on accurately provisioning resources to optimize spending while efficiently managing workloads. This involves finding the right balance between resource requirements and their costs, which can be achieved through different tactics. These strategies include adjusting resource quotas and utilizing on-demand resources. Ongoing monitoring and adjustments ensure both performance and cost-effectiveness, guarding against under-provisioning, which could lead to performance issues and downtime, potentially resulting in higher expenses than investing in additional resources initially.

COST EFFECTIVE SERVICE IMPLEMENTATION

This section discusses the importance of efficient memory management through garbage collection, utilizing parallel threads and concurrency, implementing various code patterns such as creational or structural patterns, incorporating disaster recovery (DR) measures, and choosing an appropriate pricing model. It highlights the need to optimize resource utilization while ensuring the delivery of dependable and scalable services for cost-effective service implementation.

H. Deliberate memory clean up(Garbage collection),Parallel Threads and Concurrency Usage, Code Patterns Implementation

This section highlights the importance of achieving cost-effective service implementation by prioritizing three crucial elements: deliberate memory management, parallel threading, and efficient code pattern utilization. In contemporary programming languages such as Java and Python, effective memory management, which includes garbage collection, is paramount. Garbage collection entails releasing memory held by unused objects. Employing a deliberate approach to memory cleanup, such as object pooling, caching, and memory reuse, can enhance memory utilization and reduce the memory leaks, resulting in more cost-efficient service implementation.

Usage of parallel threads and concurrency is essential for constructing high-performance services capable of managing numerous simultaneous requests. Employing multiple threads can aid in distributing workloads and

improving response times. Nonetheless, effectively managing concurrency is vital to avoid issues such as race conditions, deadlocks, and thread starvation. Methods like thread synchronization, locks, and semaphores, illustrated in Figure 4, can be utilized for efficient concurrency control. Furthermore, opting for pessimistic locking mechanisms instead of traditional optimistic ones can result in substantial cost reductions in cloud resource utilization.

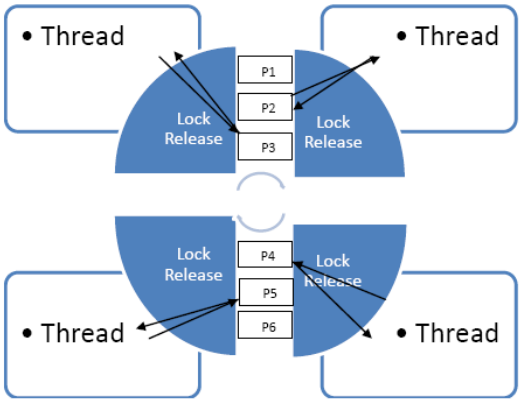

Figure 4. Thread synchronization, Locks, Semaphores

Utilizing code patterns can enhance service implementation by streamlining code maintenance, decreasing complexity, and improving scalability. Creational patterns such as Singleton and Factory are effective in managing object instances, thereby optimizing resource utilization. Structural patterns like Proxy and Decorator enhance code modularity and extendibility. Effective integration of these patterns improves code quality, making it more cost-effective to maintain and scale. Furthermore, deliberate memory management, parallel threading, and concurrency usage also play a significant role in reducing service implementation costs. Overall, optimizing memory, implementing better locking mechanisms, and leveraging code patterns all contribute to cost-effective and dependable service implementation with scalability.

I. Disaster Recovery Backups according to business

Creating a customized disaster recovery (DR) plan tailored to business requirements is essential for cost-effective development. This involves establishing Recovery Objectives, identifying critical processes, and defining Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) for each. Prioritizing backup efforts based on these objectives helps minimize costs for less critical functions. Once objectives are clear, selecting suitable backup strategies, such as incremental backups or cloud-based solutions, helps optimize costs. Automating backup processes reduces the time and resources required for backups, further reducing DR costs. Regular testing of the DR plan helps identify weaknesses, enabling cost-effective improvements. Overall, defining objectives, selecting strategies, automation, and testing work in tandem to optimize backup costs while ensuring efficient resource allocation.

J. Cost Effective Development and Pricing Model

Choosing the right pricing model is essential for cost-effective development. It starts with a thorough understanding of both fixed and variable costs, such as infrastructure, licensing, and staffing expenses. This understanding helps establish the minimum price necessary to cover costs and generate profits. Market analysis is essential for grasping the pricing environment, including competitors' strategies and unique value propositions. This insight informs the setting of an optimal price that strikes a balance between profitability and market demand.

Several pricing models, such as pay-per-use, subscription-based, tiered pricing, and freemium, offer distinct advantages and disadvantages, catering to different business needs and customer preferences. For instance, a pay-per-use model is suitable for variable usage patterns, while a subscription-based model may be preferable for predictable usage. Continuous monitoring and adjustments are crucial. Following the selection of a pricing model, ongoing evaluation is essential. This involves collecting customer feedback, analyzing usage patterns, and adapting pricing tiers or models as necessary to optimize profitability and customer satisfaction. By understanding costs, assessing the market, selecting the right pricing model, and consistently monitoring and adjusting, businesses can achieve cost-effective pricing that strikes a balance between profitability and meeting market demand.

SURVEY ON COST OPTIMIZATION IN CLOUD-NATIVE DEVELOPMENT

A survey is conducted to gather information about current practices, challenges, and trends related to cost optimization in cloud-native development. The resulting report indicates the findings, highlighting the outcomes, emerging trends, and technologies associated with cost-effective cloud-native development.

The questions are organized into distinct categories as:

- Cost Optimization Strategies and challenges
- Cost Drivers and Pricing Models
- Cloud-Native Design Patterns
- Code Patterns and Concurrency
- Deliberate Memory Clean Up
- Automated Testing
- Deployment Models and Infrastructure used
- Monitoring Tools and Strategies
- Service Level Agreements (SLAs)
- Disaster Recovery (DR)

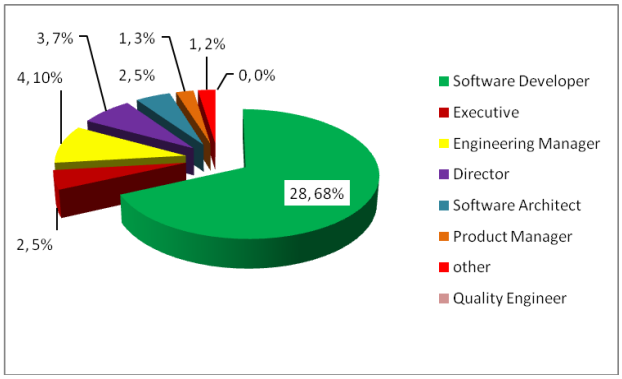

Figure 5. Survey – Users

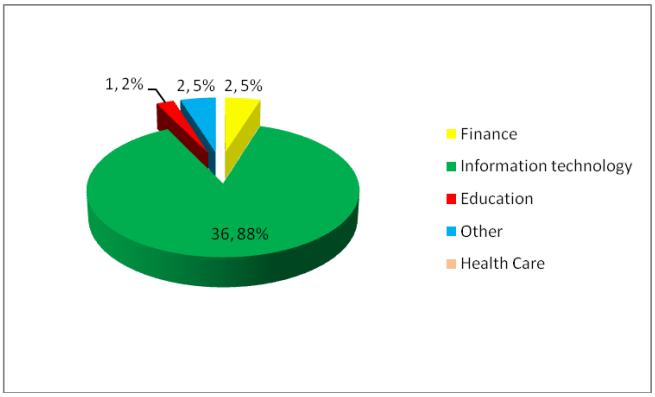

Figure 6. Survey – Industry sector

The respondents are based out of different geographic locations as - APJ (Asia-Pacific and Japan) and EMEA (Europe, Middle East and Africa) regions as shown in the Figure 5 and Figure 6.

The survey findings suggest that cloud-native development is a familiar concept for the majority of respondents, though their understanding of cost optimization strategies varies. Organizations utilize a combination of cloud infrastructure types, including hybrid, public, and private clouds. The primary cost drivers for cloud-native projects are infrastructure, development, and licensing costs. Challenges in cost optimization include difficulties in tracking usage and lack of visibility into costs. To minimize costs, respondents implement various cloud-native design patterns such as containerization, automation, microservices architecture, and cloud-native monitoring and analytics. There are differing opinions on whether implementing code patterns and using parallel threads and concurrency can contribute to cost-effectiveness. Pay-per-use and subscription-based models are considered the most suitable pricing models for cost-effective service implementation.

Garbage collection is viewed as crucial for achieving cost-effective service implementation as it effectively manages memory and prevents issues such as memory leaks. Automated testing and deployment are recognized as effective strategies in attaining cost-optimized cloud-native development by reducing manual effort and enhancing application quality and reliability. Various deployment models, including canary deployment, blue/green deployment, and rolling deployment, are mentioned. Respondents employ a mix of cloud provider cost monitoring tools, customized monitoring and alerting systems, and resource utilization monitoring and scaling techniques to optimize cloud resources. Service Level Agreements (SLAs) are established to outline the scope, performance metrics, scalability, and other key performance indicators for services. Lastly, most respondents emphasize the critical importance of disaster recovery in achieving cost-effective service implementation.

The survey's goal is to collect data and insights regarding the challenges organizations encounter in achieving cost-effective development. It offers a detailed understanding of the practical hurdles faced by professionals, aiding in pinpointing specific pain points and areas for enhancement. By examining the responses, researchers can recognize patterns, trends, and effective strategies, which can be disseminated to the broader community. This sharing of best practices enables others to derive benefits and improve their approaches accordingly.

VI. EXPERIMENTAL RESULTS

1) *Infrastructure*

Cloud-based solutions present significant cost savings by removing the need for initial hardware investments and offering a scalable, pay-as-you-go model, thereby reducing capital expenditure as illustrated in Figures 7 and 8.

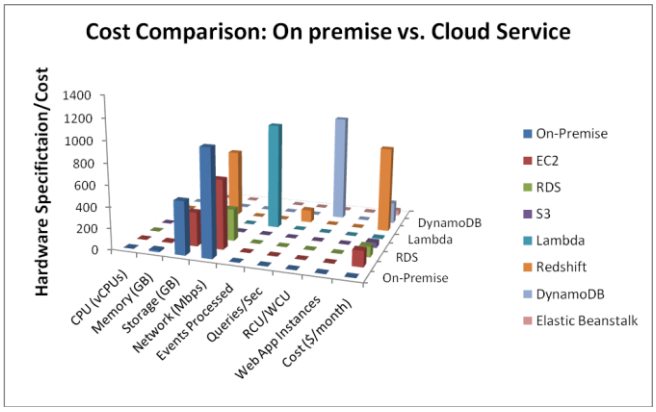

Figure 7. Cost Comparison: On premise vs. Cloud Service

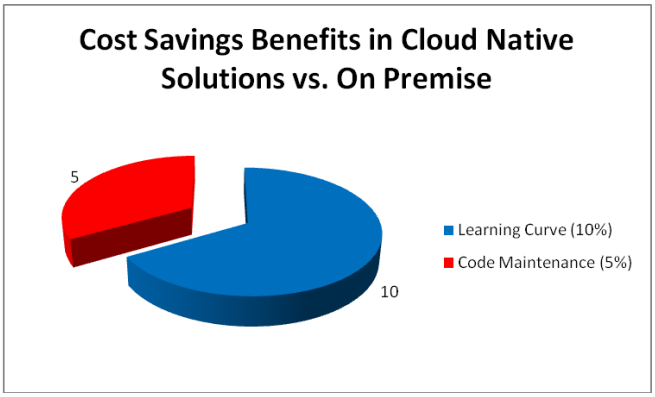

Figure 8. Cost Savings Benefits in Cloud Native Solutions vs. On Premise

2) *Development and Testing Efficiency*

Cloud-based solutions facilitate cost savings by efficiently allocating resources, allowing organizations to decrease expenses related to maintaining dedicated environments. A 15% reduction in overall expenses related to learning curve and code maintenance can be achieved. Additionally, adopting cloud-native development practices can lead to savings of 11%, while optimizing code using cost-effective key performance indicators (KPIs) can result in a 7% reduction in costs, as depicted in Figure 9.

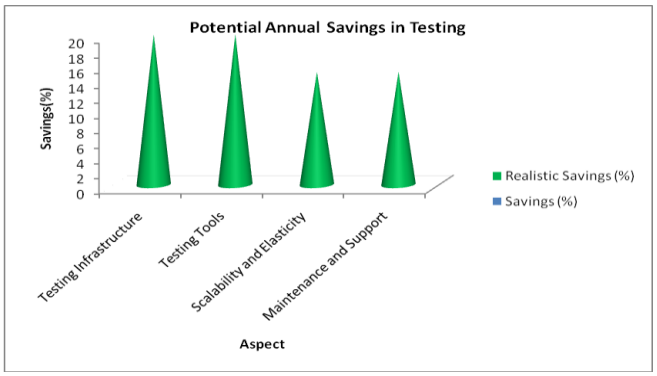

Figure 9. Potential Annual Savings in Testing

3) *Streamlined Upgrades*

Cloud-based solutions automate and streamline hardware and software upgrades, leading to cost savings estimated at 10% to 20%, as shown in Figure 10.

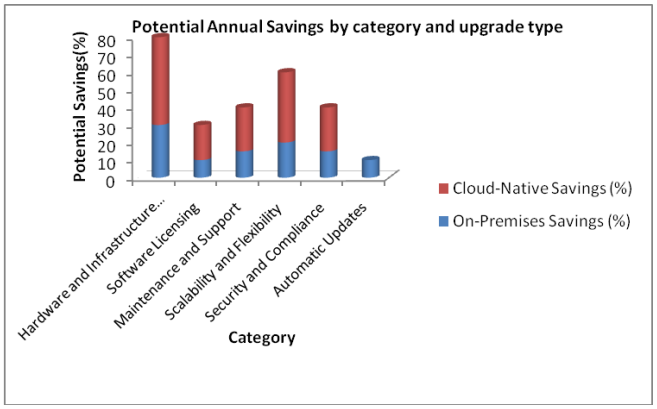

Figure 10. Potential Annual Savings by category and upgrade type

4) Consulting Services

Cloud-based solutions frequently diminish the requirement for extensive consulting services, owing to the inclusion of vendor support and expertise, which can result in potential cost reductions. According to Figure 11, organizations can realize approximately 15% in cost savings by transitioning to the cloud.

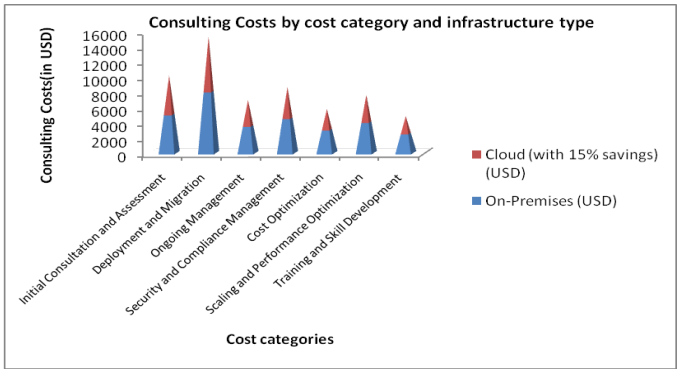

Figure 11. Consulting Costs by cost category and infrastructure type

5) Backup and Disaster Recovery

Cloud-based solutions stand out for providing cost-effective integrated backup and disaster recovery options. This reduces the need for separate investments in hardware and off-site storage. According to Figure 12, savings of up to 20% can be achieved due to the elimination of upfront hardware investments and decreased operational costs.

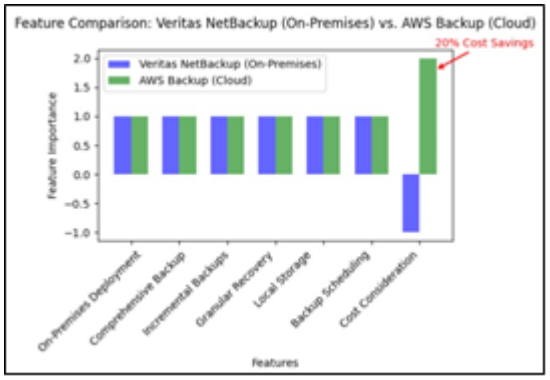

Figure 12. Feature Comparison

As a result of this comprehensive analysis, it is evident that Cloud-Based Solutions provide a solid foundation for achieving cost-effectiveness in cloud-native development. Nevertheless, there are challenges in realizing these cost efficiencies, such as difficulties in tracking and optimizing costs, limited visibility, and a lack of tools for cost optimization.

CONCLUSION

Cloud-based services must be aligned to meet customer requirements while also ensuring cost-effectiveness. Striking a careful balance between cost management and maintaining Quality of Service (QoS) is essential. This involves making informed decisions regarding pricing models, optimizing resource allocation, and continuously monitoring performance to uphold service standards while managing costs efficiently. Additionally, during service implementation, key factors such as memory management, design patterns, and optimal concurrency are prioritized.

To conclude, achieving cost-effective development involves implementing a design that is effective and efficient, utilizing advanced implementation techniques for cost optimization, and delivering cloud services that find the optimal balance between cost and Quality of Service (QoS). This approach enables businesses to provide reliable and scalable services while staying competitive and profitable in a dynamic market landscape.

REFERENCES

[1] Kumar, A., Mohapatra, D.P., & Rath, S.K. (2020). Optimal virtual machine placement in cloud environment using genetic algorithm. Journal of Ambient Intelligence and Humanized Computing, 11(7), 2777-2786.

[2] Jiang, Y., Zhou, M., Li, J., & Zomaya, A.Y. (2020). A PSO-based container placement approach for cost-effective resource allocation in cloud-native environment. Journal of Parallel and Distributed Computing, 144, 109-118.

[3] Wang, W., Cai, Y., Jin, H., & Zhang, Z. (2021). An Ant Colony Optimization Approach for Microservice Deployment in Cloud-Native Environment. IEEE Transactions on Services Computing, 14(2), 367-377.

[4] M. Zaman, M. R. Amin, M. S. Hossain, S. Lee, and H. Kim, "Energy-efficient resource allocation and virtual machine placement in cloud data centers," Journal of Parallel and Distributed Computing, vol. 117, pp. 166-178, Aug. 2018.

[5] Liu, B., Wang, H., Chen, L., Zhang, Y., & Huang, S. (2020). Cost optimization for cloud infrastructure using evolutionary algorithms. Future Generation Computer Systems, 110, 121-130.

[6] Yang, X., Huang, S., Chen, L., Zhang, Y., & Huang, J. (2020). Resource allocation and load balancing optimization in edge computing. Future Generation Computer Systems, 102, 501-510.

[7] Qian, J., Zeng, D., Wen, Y., & Zhang, Y. (2020). Optimization of container scheduling and resource allocation in cloud computing using a multi-objective genetic algorithm. Sustainable Computing: Informatics and Systems, 26, 100381.

[8] Yin, Z., Xu, X., Xu, Y., Jin, H., & Wang, Z. (2019). Dynamic application scaling and resource allocation in cloud computing using a reinforcement learning approach. IEEE Transactions on Parallel and Distributed Systems, 30(1), 45-58.

[9] Li, Z., Xu, C., Zhang, L., Wu, J., & Wei, T. (2020). Service placement and routing optimization for microservices in edge-cloud computing. Future Generation Computer Systems, 107, 784-793.

[10] J. Kim and K. Lee. (2019). Function Bench: A Suite of Workloads for Serverless Cloud Function Service. In 2019 IEEE 12th International Conference on Cloud Computing (CLOUD).

[11] Wali Ullah Khan; Tu N. Nguyen; Furqan Jameel; Muhammad Ali Jamshed; Haris Pervaiz (2021) Learning-Based Resource Allocation for Backscatter-Aided Vehicular Networks, IEEE Transactions on Intelligent Transportation Systems ,Volume: 23, Issue: 10.

[12] Alex Depoutovitch, Chong Chen, Jin Chen, Paul Larson, Shu Lin, Jack Ng, Wenlin Cui, Qiang Liu, Wei Huang, Yong Xiao, Yongjun He(2020), Taurus Database: How to be Fast, Available, and Frugal in the Cloud, SIGMOD '20: Proceedings of the 2020 ACM SIGMOD International Conference on Management of DataJune 2020 Pages 1463–1478, https://doi.org/10.1145/3318464.3386129

[13] Liangmin Wang, Zhendong Yang, Xiangmei Song(2020), SHAMC: A Secure and highly available database system in multi-cloud environment, Future Generation Computer Systems, Elsevier,Volume 105, April 2020, Pages 873-883

[14] Mengdan Feng; Sixing Hu; Marcelo H Ang; Gim Hee Lee(2019), 2D3D-Matchnet: Learning To Match Keypoints Across 2D Image And 3D Point Cloud, IEEE International Conference on Robotics and Automation (ICRA), 10.1109/ICRA.2019.8794415

[15] Jinke Ren; Yinghui He; Guan Huang; Guanding Yu; Yunlong Cai; Zhaoyang Zhang(2019), An Edge-Computing Based Architecture for Mobile Augmented Reality, IEEE Network,Volume: 33, Issue: 4, 10.1109/MNET.2018.1800132

[16] Zhe Feng, L. Ruby Leung, Nana Liu, Jingyu Wang, Robert A. Houze Jr, Jianfeng Li, Joseph C. Hardin, Dandan Chen, Jianping Guo(2019) A Global High-Resolution Mesoscale Convective System Database Using Satellite-Derived Cloud Tops, Surface Precipitation, and Tracking,JGR Atmospheres, https://doi.org/10.1029/2020JD034202.

[17] Zhao Lianheng a b, Huang Dongliang a, Chen Jingyu a, Wang Xiang a, Luo Wei c, Zhu Zhiheng a, Li Dejian d, Zuo Shi(2020), A practical photogrammetric workflow in the field for the construction of a 3D rock joint surface database, Engineering Geology, Elsevier,Volume 279, 20.

[18] Rajkumar, R., Dharanipragada, J., & Arumugam, S. (2020). A Comprehensive Survey on Cost-Effective Cloud Computing Architectures for Big Data Analytics. IEEE Access, 8, 197059-197079.

[19] Zhang, M., & Qiu, M. (2020). Cost-effective resource allocation in cloud computing: A survey. Future Generation Computer Systems, 108, 674-694.

[20] Sharma, S., & Sinha, P. (2021). Cost Effective Cloud Computing Architecture for Internet of Things (IoT) Applications. Wireless Personal Communications, 118(4), 2137-2158.

[21] Choudhary, P., Varshney, R. K., & Saini, R. (2021). A systematic literature review of cost optimization techniques in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 12(6), 6749-6775.

[22] Marimuthu, M., & Jabeen, N. (2019). Cost-effective cloud computing infrastructure for big data applications. Journal of Ambient Intelligence and Humanized Computing, 10(2), 703-719.

[23] Kumar, N., & Mohapatra, S. (2021). A survey on cost-effective cloud-based big data analytics. Journal of Big Data, 8(1), 1-21.

[24] Zou, Y., Yu, C., Zhu, Y., & Jiang, P. (2020). Cost-Effective Scheduling of Scientific Workflows in Clouds. IEEE Transactions on Cloud Computing, 8(4), 1384-1396.

[25] Mekki, K., & Dridi, F. (2020). Cost-effective scheduling for scientific workflows in the cloud. Future Generation Computer Systems, 105, 851-861.

[26] Yang, G., Xu, Y., & Liu, Y. (2021). Cost-Effective Resource Allocation for Fog Computing. IEEE Transactions on Industrial Informatics, 17(4), 2348-2358.

[27] Alghamdi, A. M., & Elleithy, K. (2020). Cost-Effective Deployment of Virtual Machines in Cloud Computing. Journal of Network and Systems Management, 28(3), 691-725.

[28] Murali Dhar, M.S., Manimegalai, R., A policy-oriented secured service for the e-commerce applications in cloud, Personal and Ubiquitous Computing, Vol.22, pp.911–919, 2018.

[29] K. S. Arvind, R. Manimegalai, and K. S. Suganya, Privacy Preserving Public Auditing for Cloud Storage Using Elliptic Curve Digital Signature, Journal of Computational and Theoretical Nanoscience, Vol.15, pp.1568–1572, 2018.