Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 05| Issue- 01| 2026

Research Article **Received:** 10-12-2025| **Accepted:** 05-01-2026 | **Published:** 21-01-2026

A Reference Architecture for Stateful Autoscaling of Virtual Machines and Containers

Ramkinker Singh

Carnegie Mellon University, USA

Abstract: The transformation in cloud-native application design has further emphasized the necessity for smarter virtual machines and autoscaling systems in containerized environments. While autoscaling for stateless services has been established as a viable design, the use of more stateful applications introduces specific challenges related to session integrity, data consistency, and service availability during scaling operations. This paper describes a stateful autoscaling reference architecture that integrates proactive and reactive models, an SLA-aware algorithm, lifecycle management, and live migration strategies across multi-cluster environments. The architecture considers the operational semantics of containers and virtual machines and leverages orchestration tools such as Kubernetes and Docker to ensure high availability and efficient resource utilization. The proposed solution addresses the scalability limitations of existing autoscaling models by integrating principles of state management, predictive analytics, and orchestration—thus overcoming prior constraints. This architecture enables robust and flexible scaling of modern applications deployed in complex distributed systems.

Keywords: Stateful Autoscaling, Cloud-native Architecture, Virtual Machines, Container Orchestration.

INTRODUCTION

Application deployment and scaling have been significantly influenced by cloud computing. The shift towards cloud-native applications has created the need for an efficient, responsive, and cost-effective autoscaling system capable of handling dynamic workloads without compromising application state. Stateless autoscaling, being relatively low in complexity, has been widely adopted in elastic system design. However, modern applications increasingly require autoscaling capabilities for stateful services, where instances maintain persistent data or store session information between user interactions.

Traditional autoscaling models are not effective in this context, especially within containerized environments and virtual machine (VM) infrastructures, where performance, resilience, and availability are critical factors. Autoscaling mechanisms in cloud environments must therefore be designed with awareness of stateful behaviors to ensure that the integrity of the system remains intact throughout the scaling process. In addition, orchestration platforms such as Kubernetes and Docker Swarm, which provide foundational support for autoscaling, face challenges in managing state-aware services. These challenges arise from issues such as session stickiness, database consistency, and inter-service dependencies. As organizations increasingly rely on containers and VMs to deliver mission-critical services, there is a growing necessity for a reference architecture that integrates scalable and state-aware design principles. The proposed research paper presents a structured reference architecture for stateful autoscaling that unifies container and virtual machine scaling approaches. It is based on both proactive and reactive autoscaling strategies, supports multi-cluster environments, and incorporates live migration techniques. The architecture includes detailed discussions on scaling decision algorithms, resilience planning, and service continuity, all grounded in empirical research and relevant literature on the subject.

BACKGROUND AND MOTIVATION

This is where the concept of autoscaling comes in, whereby a specified system can dynamically adjust computational resources—either increasing or decreasing them—based on workload variations. In stateless autoscaling, additional services are layered on top of stored data; however, stateful autoscaling must ensure that in-progress processes and active user sessions are preserved, which adds complexity to system design.

Containers and virtual machines (VMs) can both be deployed with cloud-native applications, each offering its own set of strengths and limitations. A robust architecture must consider the operational semantics of both environments. Recent innovations in cloud-native autoscaling strategies point toward the development of scaling mechanisms tailored for stateful settings. Strategies range from threshold-based policies to machine learning-based predictive scaling. To

***Corresponding Author:** Ramkinker Singh

DOI- https://doi.org/10.5281/zenodo.18928501

demonstrate how decision-making algorithms adapt to changes in workload, system metrics such as CPU utilization, memory consumption, and latency are employed to interpret workload variations within autoscaling taxonomies.

More complex stateful systems, however, require scaling methods that account for session information, configuration management, and concurrent workload forecasting. Reactive scaling is a strategy that adjusts resources in response to real-time load fluctuations, initiating resource allocation when predefined thresholds are exceeded. While reactive scaling is effective, it may fall short in environments experiencing rapid demand changes—particularly in container-based systems, where cold starts can result in performance degradation. In contrast, proactive scaling relies on workload forecasting and has become a stronger approach in recent developments. It anticipates traffic spikes and workload surges ahead of time, allowing the system to scale out preemptively. This leads to smoother transitions and minimizes service disruptions.

DESIGN REQUIREMENTS FOR STATEFUL AUTOSCALING

Implementing stateful autoscaling introduces challenges not only in the scaling logic but also in maintaining the resilience, consistency, and responsiveness of applications. Several design factors must be addressed:

- Session Persistence: Ensuring users remain connected to the correct instance during scaling operations.
- Data Synchronization: Guaranteeing consistency of application state and databases.
- Scaling Granularity: Determining whether to scale containers, entire VMs, or microservices based on performance indicators.
- Orchestration Compatibility: Supporting tools like Kubernetes that handle pod lifecycle and service discovery.

Stateless services are not effectively scaled using conventional scaling systems, particularly during periods of burst loads. It is crucial to optimize both resource utilization and user experience through the implementation of dynamic scaling policies and intelligent scheduling mechanisms. Vertical scaling within container orchestration systems is one of the most significant innovations in this area. By dynamically reallocating CPU and memory resources of already deployed containers, systems can minimize initiation delays and avoid performance degradation in existing containers, rather than deploying new ones.

Vertical scaling is considered highly efficient in unforeseen bursting scenarios where time is critical and new instance creation may lag behind the rate of demand spikes. However, challenges remain in applying vertical scaling, particularly in container scheduling and accurately estimating the required resources.

REFERENCE ARCHITECTURE OVERVIEW

A reference architecture for stateful autoscaling must include monitoring, prediction, scaling, and persistence management as integrated components within the architecture. It should incorporate feedback loops and interface seamlessly with cloud orchestration platforms. Figure 1 illustrates the proposed reference architecture.

Publisher: SARC Publisher

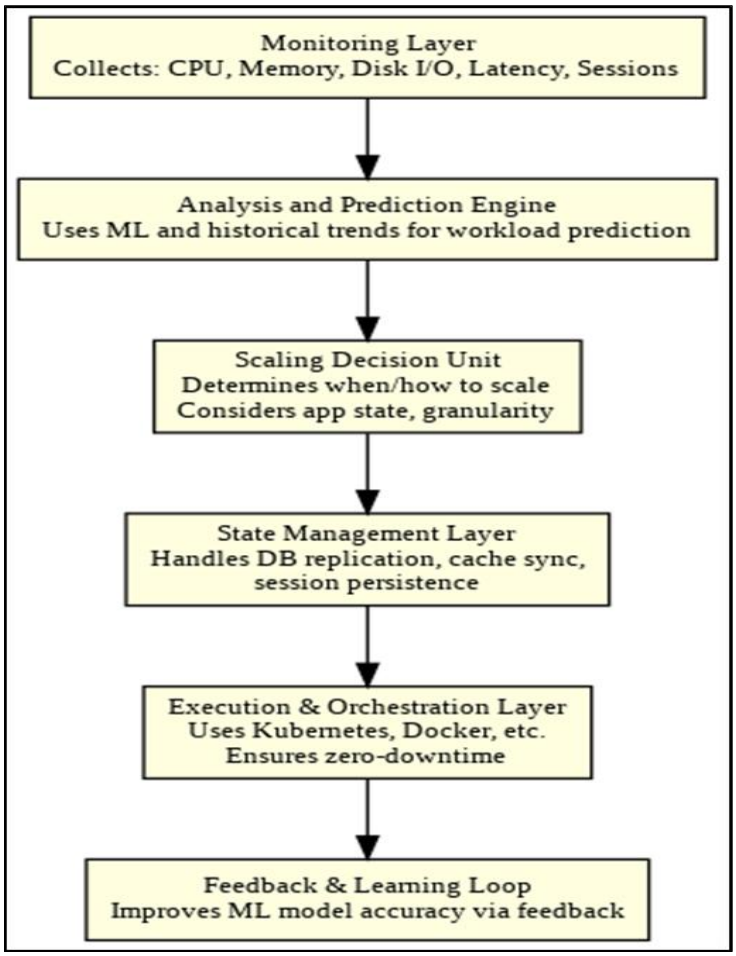

Figure 1: Reference Architecture for Stateful Autoscaling in VM and Container Environments (Diagram adapted from architectural principles and models discussed in (Xu, M. et al., 2025), (Hollander, P. 2025), and (Mekki, M., & Ksentini, A. 2025))

The architecture consists of the following key modules:

Monitoring Layer: Collects telemetry data including CPU usage, memory, disk I/O, session state, and latency from containers and VMs.

Analysis and Prediction Engine: Utilizes machine learning models to predict upcoming loads using historical trends.

Scaling Decision Unit: Applies scaling policies (proactive and reactive) to determine scaling actions. It takes into account whether the services are stateful and identifies the appropriate scaling mechanism.

State Management Layer: Ensures synchronization of session and persistent state. Includes database replication, cache sharing, and configuration handling.

Execution and Orchestration: Executes scaling actions through Kubernetes or similar tools. Ensures zero-downtime during scaling events.

Feedback and Learning Loop: Continuously updates the prediction model based on outcomes, improving future scaling accuracy.

AUTOSCALING POLICIES AND SLA CONSIDERATIONS

Autoscaling in stateful environments must be aligned with Service Level Agreements (SLAs), incorporating performance constraints and system availability guarantees. Autoscaling plans should be SLA-conscious to ensure the fulfillment of SLA objectives, including acceptable error rates, latency, and throughput thresholds (Peri, A. et al., 2025). This approach helps improve system reliability and minimizes SLA violations in high-throughput applications. An SLA-aware autoscaling algorithm can be employed to prioritize critical services, assign weights to different workloads, and adjust scaling frequency to prevent over-provisioning. Furthermore, it can proactively migrate workloads to mitigate the risk of overload. Table demonstrates that SLA fulfilment is correlated to the autoscaling strategy.

Publisher: SARC Publisher

Table 1: SLA Metrics and Autoscaling Strategy Impact

SLA Metric	Reactive Scaling Impact	Proactive Scaling Impact	Vertical Scaling Impact
Latency	High variability	Low variability	Minimal latency increase
Availability	Dependent on triggers	High availability	Sustained availability
Resource Cost	Potential overspend	Optimized cost	Low cost with constraints
Service Continuity	Risk of session loss	Stable sessions	Sessions retained

The table reveals that proactive and vertical scaling techniques offer superior SLA compliance for stateful applications. However, implementing these strategies at scale requires comprehensive resource prediction and sophisticated monitoring tools (Pozdniakova, O. 2025).

PROACTIVE AUTOSCALING AND PERFORMANCE OPTIMIZATION

Proactive autoscaling is highly beneficial for stateful applications, particularly in scenarios where loads are unpredictable and variable. In proactive strategies, changes in load are forecasted, and resources are provisioned prior to the occurrence of performance bottlenecks. This scaling model is based on the assumption of workload prediction algorithms that are informed by historical usage trends (Hollander, P. 2025).

Time-series data—such as CPU load, memory pressure, and request rates—are commonly used as input for predictive models. These predictions can be integrated into the autoscaling logic, offering the advantage of enhancing application availability and responsiveness to changing traffic conditions (Hollander, P. 2025).

Performance overhead resulting from container or VM deployment can be reduced through anticipation, leading to lower cold start latency and higher system throughput. Scaling granularity should also be effectively managed through proactive scaling. For instance, some systems may be scaled at the level of individual microservices, but not at the level of the entire application stack. Such precision ensures that resources are allocated precisely where needed, enhancing cost-effectiveness. Therefore, orchestration layers must possess advanced capabilities to enable selective scaling, maintain consistency, and support inter-service communication (Hollander, P. 2025).

In dynamic environments, security and stability are highly significant. Stateful workloads can expose the system to vulnerabilities arising from design flaws, which may compromise the security of containers—such as unauthenticated replicas or insecure endpoints. A biologically inspired security model is necessary to mitigate these risks, as dynamic security systems should respond to evolving threat environments similarly to how biological immune systems operate (Moomaw, A. H. 2025).

LIFECYCLE MANAGEMENT OF STATEFUL CONTAINERS

Stateful autoscaling is a key attribute that enables the management of container and virtual machine lifecycles without disrupting service delivery. Unlike stateless services, stateful containers require tight coordination during both upscaling and downscaling operations. Lifecycle management in this context involves handling checkpoints, maintaining persistent storage, synchronizing backups, and ensuring that containers remain at their designated locations within the cluster (Suthari et al. 2025).

The system must continuously monitor both intra- and inter-cluster communication to support robust stateful microservices, particularly in multi-cluster environments. Lifecycle policies should proactively manage scaling sequences, determine the optimal placement of instances, and migrate them without interrupting data flow. Proactive lifecycle management schemes serve as resilience frameworks by enabling predictive failure handling, effective workload distribution, and seamless live container migration (Mekki, M., & Ksentini, A. 2025).

This is a solution that would ensure high availability and minimize downtime, since workloads are migrated to nodes or clusters in a smooth way. Interdependence among the microservices must also be taken into consideration in proactive policies, according to which when one microservice is being scaled, others should remain stable. Any scaling plan should have an engine that is aware of the topology and service health metrics, and takes them into account in the scaling decision (Mekki, M., & Ksentini, A. 2025).

Such an architecture allows one to experience graceful degradation in the case of resource depletion, by means of rollback mechanisms and traffic diversion to lessen the effect on the user.

Publisher: SARC Publisher

These are mandatory in distributed systems where reliance on services and maintenance of a session have direct effects on business continuity.

STABILITY AND SCALABILITY THROUGH KUBERNETES AND DOCKER

The powerful capabilities of container orchestration systems such as Docker and Kubernetes are a necessity for deploying scaled and stable cloud-native applications. Such platforms support horizontal and vertical scaling per se, yet need to be properly configured and optimized to enable stateful applications. One of the key abstractions applied in Kubernetes to manage persistent identities and storage for stateful pods is the StatefulSet. System architects must ensure that service and data inconsistency do not arise as a result of scaling activities involving StatefulSets. Persistent Volumes also tend to be provisioned and dynamically attached during pod recreation and pod migration, which is critical (Ogungbola, A. 2025).

Moreover, there exist readiness probes and liveness probes to ensure that traffic is redirected to healthy and fully upgraded containers, preventing temporary failures. Kubernetes also includes node affinity/anti-affinity policies and taints, which can be utilized to regulate the placement of pods, enabling optimal utilization of available resources and protecting the integrity of the system (Ogungbola, A. 2025). Decoupling the application logic from the storage layer is one of the most critical factors in ensuring that the system remains scalable. Kubernetes facilitates volume abstraction through Persistent Volume Claims (PVCs), allowing pods to scale without regard to the storage media involved.

The decoupling is more flexible and makes horizontal autoscaling of even stateful components possible. Besides, the containerization approach applied by Docker supports simplified dependency management, making it easy to deploy applications across various environments. With Docker deployed alongside Kubernetes, it is possible to implement changes easily and roll back instances of improperly deployed applications without affecting the application's status, thereby ensuring zero downtime (Ogungbola, A. 2025).

SERVERLESS INTEGRATION AND RESOURCE OPTIMIZATION

The infrastructure of stateful applications is increasingly being replaced by containers and VMs; however, serverless computing is also offering integration possibilities. Although stateless functionality is a traditional foundation of serverless paradigms, persistent event storage and service orchestration layers are gradually transforming them into more stateful models. The on-demand allocation mechanism implements serverless resource optimization models such that functions are consumed only when required.

This model reduces idle resource utilization and cost; however, state remains difficult to maintain. Nevertheless, recent literature identifies emerging trends in which serverless systems are coupled with persistent state stores or durable function containers, bridging short-lived and durable models of execution (Adeola, O. V. 2025).

These techniques are applicable in learning and business systems that demand low-latency responses and user session data. As an example, through serverless backends and containerized frontends, hybrid systems may be designed whereby event triggers can scale stateless tasks within a few seconds, while user continuity is guaranteed by container-hosted stateful services (Adeola, O. V. 2025).

Interoperability is very important in these architectures. Knative or OpenFaaS are Kubernetes-native serverless systems, which include features that combine serverless functions with long-running pods to permit dynamic scaling—at the cost of state transitions, which are managed effectively.

SYNERGY BETWEEN CONTAINERS AND SERVERLESS MODELS

As applications grow more complex, containers combined with serverless infrastructures will become a viable solution for the ongoing improvement of scalability and operational efficiency. Due to their enduring context and lifecycle, containers are better suited for stateful services. Serverless functions are useful when responding to intermittent workloads and asynchronous processes. A hybrid of the two paradigms can move some workloads—such as data processing, caching, and logging—to serverless functions, while keeping the majority of the business logic within containerized services. The granular autoscaling of functions based on the type of functions and short-term resource profiles can be achieved through this architectural design, enabling efficient allocation and providing elasticity (Vora, V. A. 2025).

Publisher: SARC Publisher

One such application can be a container-based system that serves a machine learning model and offloads feature extraction and data ingestion operations to serverless functions. As traffic flows, the secondary services can be automatically scaled without impacting the model-serving container, thereby avoiding resource constraints and increased response times (Vora, V. A. 2025).

The integration should also bring about uniformity in the sharing of states and the propagation of identities among function calls. State can be shared between stateless functions through shared databases or maintained caches, supporting continuity between asynchronous calls. This architectural combination creates opportunities for scalability strategies that are not restrained and that balance cost-effectiveness with system responsiveness (Rubinstein, 2025)

Live Migration Techniques and Edge Integration

Edge-cloud hybrid environments are particularly complex with respect to the autoscaling of stateful workloads. Latency sensitivity and connection issues complicate scaling decisions and make them more difficult to implement. One approach that is on the rise is live container migration, which involves moving containers between edge and cloud nodes without causing any interruption to running sessions or disruption to information streams.

Live migration of stateful services involves persistent state within a container, which is migrated to other nodes and then restored on the destination. Effective checkpointing and robust communication synchrony across storage systems and high-speed communication channels are necessary to achieve this (Al-Bayram, R. B., & Qasha, R. P. 2025).

These techniques are vital in online games, diagnostic medicine, or financial market trading, where uptime and low latency are of utmost priority. However, session affinity, data consistency, and the reduction of cold restarts remain challenging to support. Checkpointing and pre-copy migration methods are also applied to reduce downtime, with the trade-off being increased system complexity. In addition, the resource provisioning process in heterogeneous edge environments necessitates the use of precise modeling to forecast demand-based scaling as well as scale localities (Al-Bayram, R. B., & Qasha, R. P. 2025).

Figure 2 below illustrates the latency improvement observed in a sample edge-cloud migration scenario using live container migration.

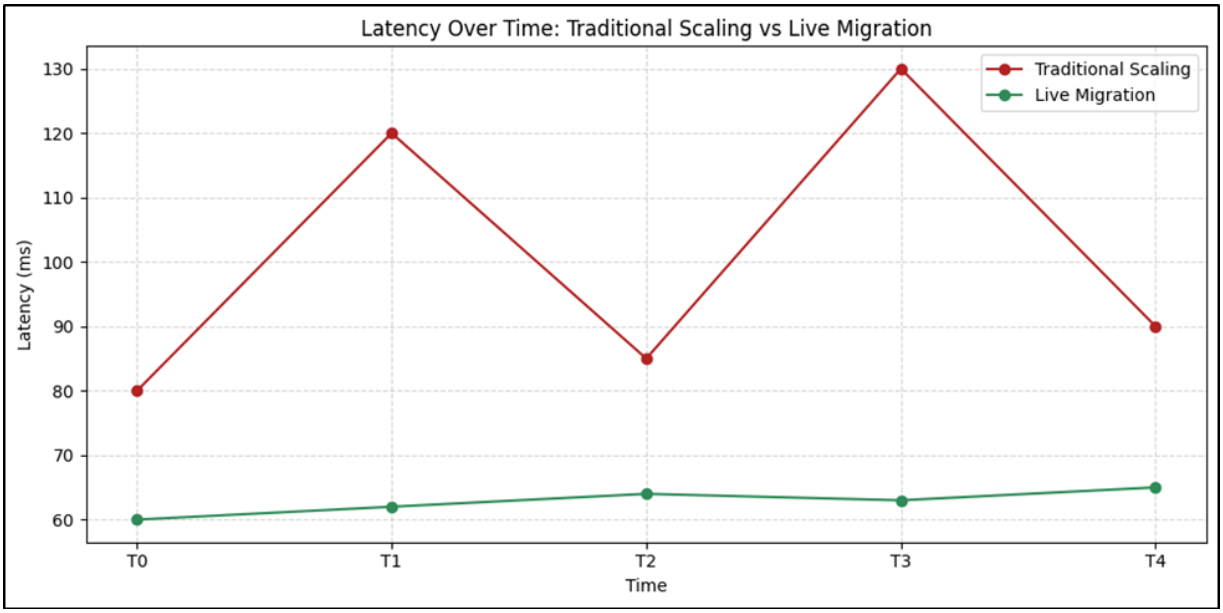

Figure 2: Latency Comparison: Traditional Scaling vs. Live Migration in Edge-Cloud Deployments
Source: Adapted based on techniques and performance observations discussed in (Mekki, M., & Ksentini, A. 2025) and (Al-Bayram, R. B., & Qasha, R. P. 2025)

The graph demonstrates that live migration significantly reduces latency spikes during scaling events compared to traditional stop-and-start strategies. This finding reinforces the importance of incorporating live migration into any reference architecture designed for real-time or mission-critical systems.

Publisher: SARC Publisher

CONCLUSION

The new dynamics of cloud-native computing and complex distributed systems have resulted in an urgent need for a robust reference architecture that supports stateful autoscaling of virtual machine and containerized environments. Unlike stateless services, stateful applications introduce new complexities related to session persistence, data integrity, service continuity, and orchestration compatibility.

The current study proposes a reference architecture founded on proactive scaling policy, SLA-sensitive strategy, lifecycle management, and hybrid cloud capability. It also incorporates vertical scaling, predictive analytics, and live migration to enhance the performance and high availability of various environments.

This architecture is founded on the synthesis of the best concepts from VMs, containers, and serverless paradigms, and is employed to support a diverse range of workloads, enhance operational efficiency, and align with existing DevOps ideologies. Such architectures will be further supplemented with machine learning enhancements, orchestration frameworks, and edge computing advancements for their components in the future.

REFERENCES

1. Xu, M., Wen, L., Liao, J., Wu, H., Ye, K., & Xu, C. "Auto-scaling Approaches for Cloud-native Applications: A Survey and Taxonomy." arXiv preprint arXiv:2507.17128 (2025).
2. Peri, A., Tsenos, M., & Kalogeraki, V. "Vertical Scaling Can Save Time: Optimizing Container Scheduling to Handle Sudden Bursts." *Proceedings of the 19th ACM International Conference on Distributed and Event-based Systems*. (2025).
3. Pozdniakova, O. "Research of service level agreement aware autoscaling algorithms for containerized cloud-native applications." (2025).
4. Hollander, P. "Proactive autoscaling for performance variable cloud infrastructure." *MS thesis*. (2025).
5. Moomaw, A. H. "Death by design: a biological approach to container security." (2025).
6. Mekki, M., & Ksentini, A. "Resiliency focused proactive lifecycle management for stateful microservices in multi-cluster containerized environments." *Computer Communications* 236 (2025): 108111.
7. Ogungbola, A. "Scalability and Stability in Cloud-Native Applications: Lessons from Docker and Kubernetes Deployments." *Knowledge Production and Management in Africa: Exploring Socioeconomics, Indigenous Knowledge, and Technological Integration Across a Continent. Cham: Springer Nature Switzerland*, (2025): 33-53.
8. Suthari, Y., Thakker, Z., Govindarajan, S., Krishnan, N., Raju, V., & Rao Katta, S. K. "Cost Optimization Techniques for Efficient Resource Allocation in Cloud Computing Environments." *Proceedings of the 2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*. IEEE, (2025). 793-797.
9. Adeola, O. V. "Resource Management in Serverless Computing: A Review and Perspective on Educational Integration." *Authorea Preprints* (2025).
10. Vora, V. A. "Serverless computing and container synergy in network architectures." *World Journal of Advanced Research and Reviews*, 26.1 (2025): 3429-3438.
11. Rubinstein, I. "Leading High-Impact Distribution Strategies: Publisher Partnerships and Monetization Innovation in Digital Platforms." *IPHO-Journal of Advance Research in Business Management and Accounting* 3.12 (2025): 48-55.
12. Al-Bayram, R. B., & Qasha, R. P. "Provisioning of live container migration in edge/cloud environments: techniques and challenges." *J. Appl. Eng. Technol. Sci* 6.2 (2025): 829-848.

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:
Singh, R. "A Reference Architecture for Stateful Autoscaling of Virtual Machines and Containers" *Sarcouncil Journal of Engineering and Computer Sciences* 5.1 (2026): pp 41-47.

Publisher: SARC Publisher