International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 3, Issue 2, 39-48, 2022
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V3I2P105

Original Article

Serverless PWAs: Reducing Backend Load with Cloud Functions and Edge Computing

Varun Kumar Chowdary Gorantla[1], Sarath Chandra Madala[2]
[1,2]Independent Researcher USA.

Abstract - *PWAs have been adopted widely due to their performance, the feel, and accessibility that come with web apps and the installation of web apps. However, their use expands; a backend sprawling appears insufficient to respond to consumer desire, productivity stutters, and the cost of operations rises. This paper focuses on incorporating serverless computation and edge computation to decrease the backend burden and improve the responsiveness of PWAs. Serverless computing in the form of Function as a Service is a system where developers can execute backend code without dealing with the concerns related to servers and infrastructures and additionally provides auto-scaling and cost-effectiveness through services such as AWS Lambda, Firebase Functions, and Azure Functions. When integrated with Edge Computing, according to which the computational process is also partial and exists as geographically dispersed nodes, the overall setup provides the necessary characteristics of low latency, high reliability, and scalability. This work identifies how cloud functions can carry the API logic and business rules and how the nodes control data caching, real-time analysis, and pre-processing. An implementation model is described on the Raspberry Pi edge cluster using OpenFaaS, AWS, and AZURE cloud services. The results of experiments with mixed workloads show lower response time, lower costs, and lower load on the origin server. Thus, the integration of serverless edges increases the scalability and performance of PWAs while solving the issues that can be met in traditional backend systems and architectures.*

Keywords - *Serverless computing, Edge computing, Progressive Web Applications, Backend load reduction, Cloud functions, OpenFaaS, AWS Lambda, Edge nodes.*

1. Introduction

PWAs are the new way users engage with and interact with websites, as they provide the benefits of the web together with the functionality of applications. Some of the trends of PWAs include offline capability, push notification, and home screen installation-supported devices, among others. However, the advancement of different styles of controlling web applications and real-time performance expectations have become major concerns of traditional web-based systems. [1-3] With increasing traffic and the user base reaching out to the world, the problem of maintaining homogeneity in service and availability becomes complex, further becoming more complex when back ends are unified and large. To address these limitations, the integration of serverless computing and edge computing into the PWA architecture was proposed as the prospective solution. By using the concept of cloud functions, serverless computing can be described as a back-end computing model in which computing resources are retrieved on-demand by executing instructions triggered by events and automatically managed by cloud services and software. This frees up resources of having to provision or manage servers and rein in expenses and complexity in operations. Concurrently, edge computing introduces the concepts of controlling and performing functions at micro data centers near the users globally.

This helps to enable lower latencies, improving the system's reliability so that users' requests are processed as quickly and effectively as possible. PWA is ideal for serverless and edge since it offers a scalable, efficient, and robust backend foundation for hosting the application. It is rather generous to state that such application architecture has been designed recently, which has been made possible by the availability of tools and services provided by different cloud platforms, including Amazon Web Services, Google Cloud, and Cloudflare, among others. For example, AWS Lambda@Edge, Google Cloud Functions, and Cloudflare Workers are developed to let developers run code near the users, enabling dynamic functionality without much strain on the backend. This paper also points out how these technologies can be utilized hand in hand to minimize the number of requests to the back end, enhance speed, and decrease app development for more complex PWAs in the future. Thus, based on the analysis of architectural patterns, real-world use cases, and performance characteristics, we can give a holistic picture of the advantages and disadvantages of using the serverless and edge-first approach in building PWA.

2. Background and Related Work

The combination of serverless computing and edge computing is fast becoming popular in web application development since it helps create easily scalable and low-latency solutions ideal for progressive web applications. Thus, as PWAs become more

complex and acquire more users, there is a demand for architectures that can accommodate different loads, be economically efficient, and provide the end-users with smooth experiences. [4-6]This section looks at the basis and literature review on serverless and edge computing and integrates them to improve the efficacy of PWAs.

2.1 Serverless Computing

Serverless computing has made a radical change in the backend development as a service, with the capability of developing and deploying applications without needing any server. Services, such as AWS Lambda, Google Cloud Functions, or Microsoft Azure Functions that use the Function-as-a-Service or FaaS model, lead to managing the function as an event-based but stateless service that scales with the application's needs. This model makes deployment easy and lowers operation costs and overheads, making charges only when functions are run. Serverless has also looked for the distribution of applications and computation to edge or near-user contexts to improve response time. The elements of serverless computing for applications involve flexibility and the ability to scale in real-time, which is some of the general features of modern progressive web applications. Some frameworks like OpenFaaS Kubeless were used when small serverless applications were needed for cloud and edge facilities. Further research has been undertaken to establish how the serverless architecture's dynamic resource management can relieve a lot of pressure on back-end service, especially during downtime, which is typical with web-based applications.

2.2 Edge Computing

Edge Computing acts within the serverless architecture as the distributed computation that provides computation closer to the data production points and the end users. This reduces the latency and bandwidth and provides regular interaction, as it reduces the chances of having to relay every request through the centralized servers that are far away. While edge computing has been found particularly useful in areas with high latency requirements, such as IoT, online gaming, and augmented reality, PWAs are among the areas beginning to be associated with its use. Several works have come up to explain how edge computing aids serverless deployment through microservices and serverless functions that are deployed at the edge computing nodes. For instance, Wissen et al. have depicted that shifting computation tasks on microservers at the edge helps achieve higher performance along with decentralized reduction in data centre dependence. These have created a foundation for serverless edge computing, the relatively new paradigm of computing architecture that unites the flexibility of serverless computing and the speed of Edge computing.

Serverless Edge Computing is a solution that is a mixture of both a full cloud and a full edge solution because a full cloud solution has its drawbacks. It also makes it possible for you to achieve logical processing of events right at the network edge, making the processing faster, more reliable, and easily balanced. Explorations in the International Journal of Novel Research and Development (IJNRD) describe frameworks that incorporate serverless functions with edge clouds to enhance latency and system functioning. These systems always utilize geo-distributed Edge nodes and orchestrated microservices for Resource Management in a Distributed Network. However, these technologies are applied effectively in PWAs, as described below. The loosely defined concept of PWAs is that they are bang up to date, conform to standard web specifications, are device-agnostic, and cross-browser compatible. The combination with serverless edge computing in PWA architecture also opens up numerous improvements in terms of performance, such as distributed computing, context updates, and personalization, which can be done from the edges without high loads on the centralized backend. Main use cases include real-time analysis of information, secure data processing, content delivery based on geographical location, and offline services.

3. Proposed Approach: Serverless PWAs with Cloud Functions and Edge Computing

3.1 Architecture Overview

PWAs use cloud functions and edge computing to improve functionality, efficiency, capacity, and resilience. This shifts a number of backend computations to both serverless cloud functions and distributed edge networks, which makes for much less load on the central servers. [7-10] They include the following: User Device or Progressive Web App (PWA), Edge Computing Layer, Serverless Backend, and Monitoring & Security. At the User Device level, the Progressive Web Application (PWA) communicates with the user and can retrieve the content from the local cache and the backend. Service workers are important for managing cached data so that frequently accessed resources and data are available when accessed. This caching mechanism minimizes the usage of backend services and thus enhances the response time, especially when the internet connection is poor. As long as the current state and request information are available at the edge or serverless layers, depending on which is optimal, the PWA frontend will pull for content.

The edge computing layer enhances caching and functions that run on Workers from Cloudflare and Lambda@Edge from AWS, storing data nearer to clients. If a user demands an object, the edge cache looks for the latest copy of the object. If the necessary data resides in the cache, it shall be delivered as soon as possible, as it takes a couple of microseconds to respond.

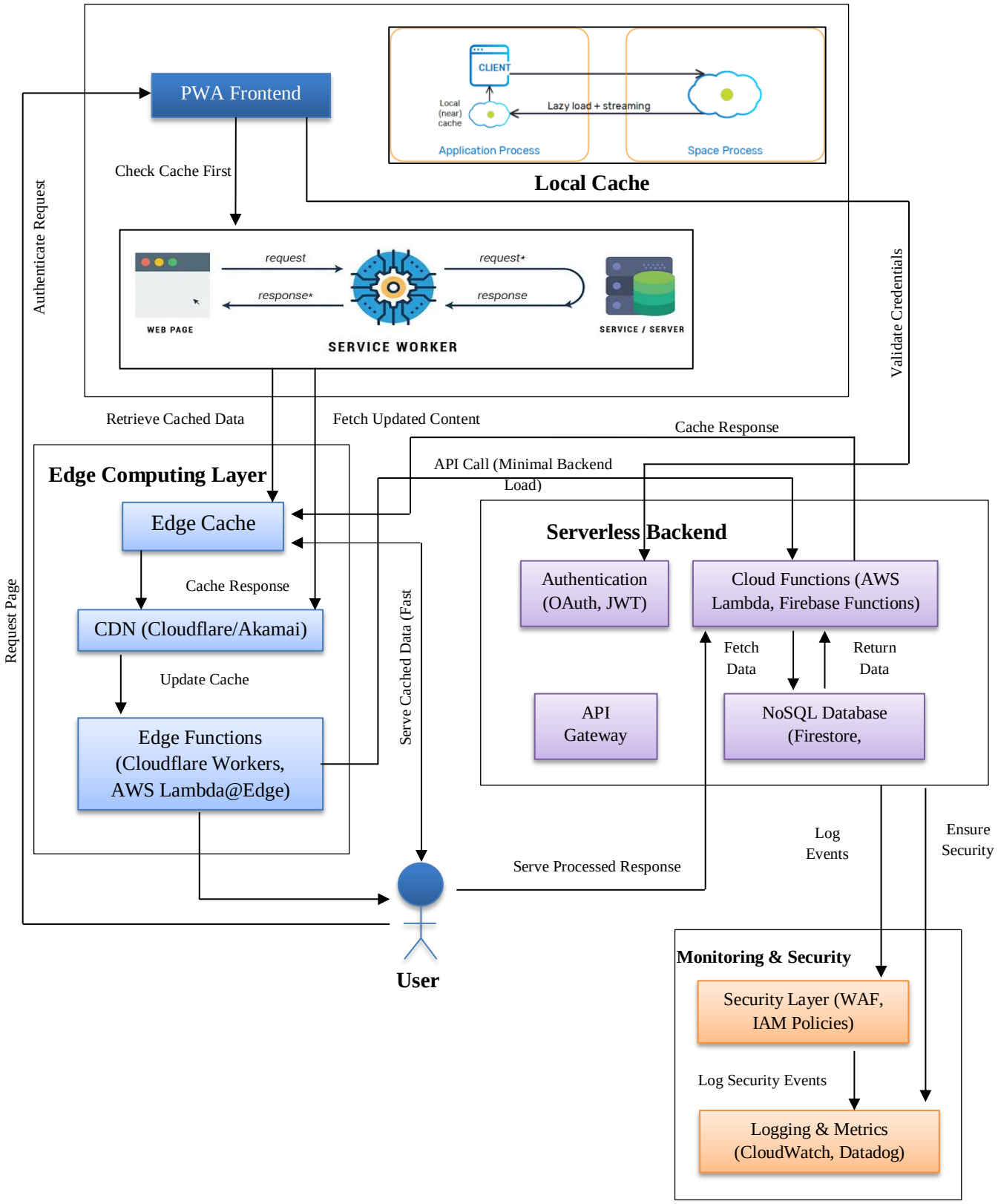

Fig 1: Serverless PWA Architecture with Cloud Functions and Edge Computing

Otherwise, the request is passed to the subsequent level for further processing or other required operations. Also, the edge functions can perform simple computations closer to the user, enhancing efficiency and responsiveness, especially for dynamic web-based applications. The serverless backend is a core data processing component of the application with authentication and computing logic functions. This includes API Gateways that act to validate requests, cloud functions (AWS Lambda, Firebase Functions) that handle data, and NoSQL databases such as Firestore and DynamoDB for quick data retrieval.

By utilizing the fully serverless solution, the backend is fully elastic and does not require providing resources with specific characteristics. This leads to a reduction of considerable costs while also greatly improving availability and fault tolerance. Monitoring & Security is another feature of the solution that offers proper protection based on security measures and observation instruments. Web Application Firewall (WAF) and Identity and Access Management (IAM) based access control safeguards the system against threats and hackers from gaining unauthorized access. Monitoring and metrics run through services like CloudWatch and Datadog allow real-time monitoring of the performance, security, and health of services crucial to the architecture's correct functioning.

3.2 Backend Load Reduction Strategy

3.2.1 How Cloud Functions Handle API Requests

Cloud functions are the key processing elements in serverless infrastructure that accompany a workload. They are triggered based on certain events, including HTTP calls, changes in the database, or any authentication events, and they run within their temporary environments. In PWAs, every time a particular API request, for example, a user login or a data retrieval, a cloud function is created to contain the logic pertaining to that particular request. These functions are stateless and run only if summoned, which makes it easy to do away with constant server operations. Also, the AWS Lambda, Firebase Cloud Function, and Azure Functions manage the horizontal scalability by themselves and with the resources needed for the current loads. This architecture reduces the burden of high volume traffic on the backend proprieties while at the same time avoiding the expense of idle servers.

Software density is also reduced with the help of cloud functions due to the separation of application and infrastructure, which lowers the complexity and makes it more modular. This is due to the fact that each function is usually executed concerning a specific task, such as validating an input, authenticating a user, or querying databases, thus making the majority of these maintainable and testable. In addition, cloud functions can work with other serverless services such as NoSQL databases, including DynamoDB and Firestore, and storage, including S3 and Firebase Storage, making request and response patterns more efficient without overloading central servers.

3.2.2 Data Caching and Processing at the Edge

Caching and computation at the edge are regarded as one of the most effective ways to reduce backend load. This is a network of distributed servers implemented through what is commonly known as the Content Delivery Networks (CDNs) like Cloudflare and Akamai. In this given architecture layout, the objects retrieved frequently, including HTML templates, configuration files for software applications, and API responses, are stored at the edge nodes. If any request is made, it first looks at the edge cache, which provides the data without involving the other parts of the system. It greatly reduces the response time and the number of requests that get through to the cloud functions and databases. Static caching, Cloudflare Workers, and AWS Lambda@Edge mostly deal with serving lightweight compute workloads at the edge. Such functions can prepare data input, implement personalization logic, validate data, and/or even receive and process calls from identified API scopes without invoking the original server. This form of distributed processing is made in a way that it delivers a low-latency experience and reduces the load to the central backends, especially events that would require high concurrency, such as live feeds, high-end user interactions, or global releases.

3.3 Implementation Model

3.3.1 Components Used (Firebase, AWS Lambda, Cloudflare Workers, etc.)

The serverless PWA model being discussed here is built on interconnected standards that are specifically cloud-based to ensure all the following characteristics: performance, scalability, and availability. [11-13] Firebase is used for hosting, authentication, and data storage by Firestore, a NoSQL cloud database. Firebase Cloud Functions are backend functions associated with its services and can scale automatically in response to events and real-time use. AWS Lambda is used for more general or computing-intensive functions or for those where integration with other AWS services can be incorporated. For instance, Lambda can execute operations that may include but are not limited to image processing, data summarization, or other custom login credentials methods.

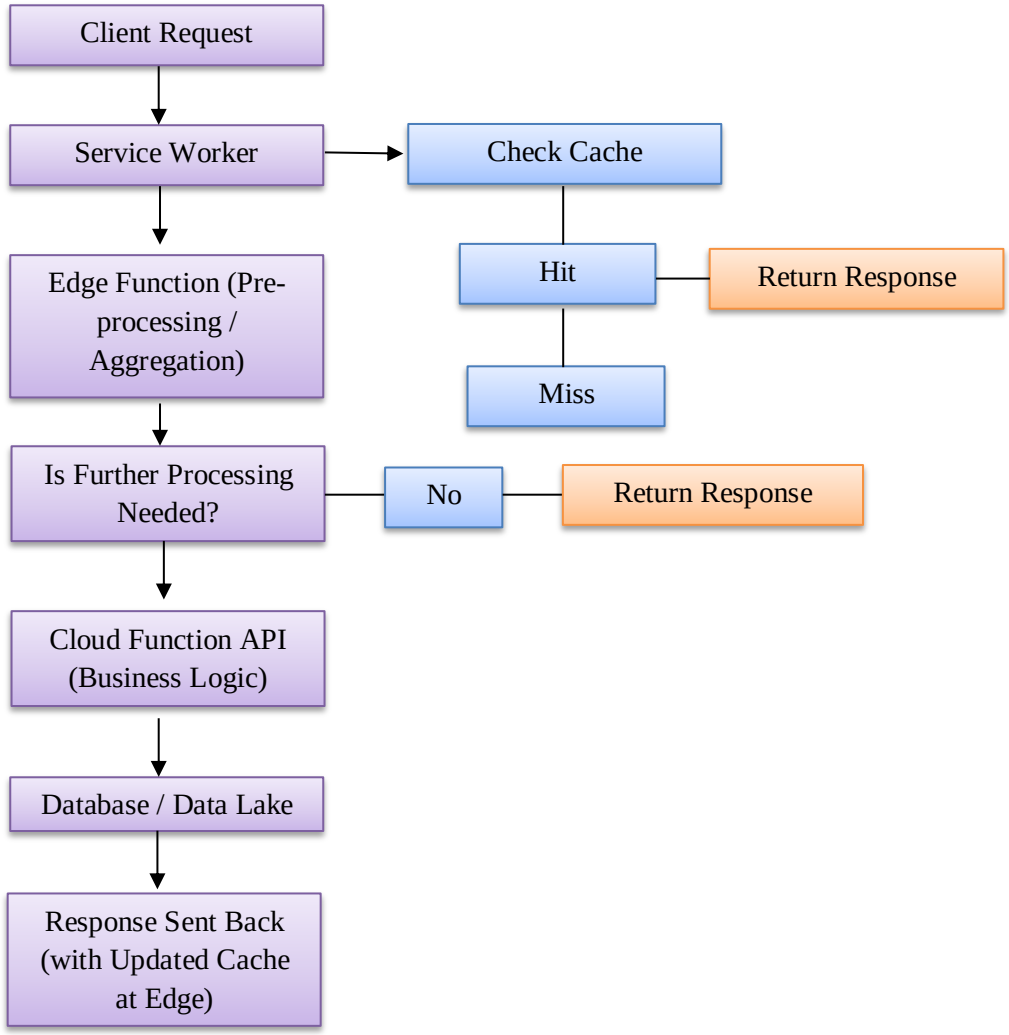

Fig 2: Serverless Strategy for Reducing Backend Load

Cloudflare Workers are crucial for running custom code at the edges, thus accelerating content delivery and making appropriate routing decisions. These workers intercept and modify HTTP requests and responses, provide a cached version of an API, and act as a security filter. For deeper computing tasks on the edge, AWS Lambda@Edge is applied with Amazon CloudFront to deploy functions globally at such locations. The utilisation of these distributed computing platforms guarantees such incongruities to be as low as possible. It makes the system extremely cortisone-sensitive without excessively putting pressure on a central processor.

3.3.2 Deployment Workflow

The serverless PWA architecture's deployment flow is designed similarly to a microservices-based architecture that combines CI/CD methods, in which the front and back ends are separated. PWA frontend in HTML, CSS, JS, and with added Service Workers, could be deployed on Firebase Hosting or other static web hosts. A service worker is installed on the client device to handle caching, offline, and background synchronization. This frontend layer communicates with cloud functions and edge services through HTTP and HTTPS requests. Cloud functions are installed in the cloud by command-line interfaces developed by Firebase and AWS. While application developers write functions such as HTTP or database writes in the code, PaaS deployment manages runtime. Likewise, edge functions, Cloudflare Workers, or AWS Lambda@Edge functions are uploaded to the regional edge node using CLI or graphical user interface, with routing configurations described in files. Access control of an application is applied through IAM roles and WAF rules when the application is being deployed. Due to tracking and logging, the system is visible after they implement AWS CloudWatch, Firebase Analytics, and Datadog to track the system performance, the

rates of errors, and other usage metrics. In combination, this deployment model allows for shortened iteration times, high environmental availability, minimal operational overheads, and low impact on the infrastructure.

4. Performance Analysis and Evaluation

4.1 Experimental Setup

An account was created to evaluate the performance of serverless PWAs that rely on cloud functions and edge computing. To achieve these goals, the controlled experiment environment was created to resemble real-world conditions as closely as possible. They aimed to test the proposed architecture's impact in that it minimizes backend load, improves response time, and scales up dynamically as the workload increases. [14-16] An edge computing integrated with cloud serverless solutions was used to implement the solution and provide a clear evaluation of latency, throughput, and system performance. Components included an edge tier based on Raspberry Pi OpenFaaS cluster, cloud-based serverless AWS, Microsoft Azure solutions, and tools for industry-standard performance metrics. The edge infrastructure involved a group of Raspberry Pi computers to implement serverless computing by OpenFaaS. OpenFaaS was selected because of its lightweight architecture and integration with the Kubernetes cluster. The Raspberry Pi cluster replica and edge network topology help execute functions near the consumer. This allowed for the direct comparison of edge processing and cloud serverless mode of operation by comparing the number of round turns and the shifts in load towards the backend.

Edge functions included actions like caching API results, handling customer queries, and running simplified artificial intelligence operations, such as picture identification. AWS Lambda and Azure Functions were used for the traditional serverless model to test the centralized cloud environment. These platforms gave a reference point to gauge the benefits of distributed computing on the edge. Both cloud services were set in the auto-scaling strategy to what was seen in the number of requests occurring in the cloud. To compare Lambda and Azure Functions to edge computing, backend workloads such as database queries, authentication processing, and computationally intensive AI jobs like object recognition were performed on them, and the time, cost, and scalability were recorded. Tools for performance evaluation were incorporated into the experiment to increase the reliability and reproducibility of the tests Kubernetes used to manage the distribution of functions and also took responsibility for monitoring the cluster's health on the Raspberry Pi devices. Load testing is a tool that helps to emulate the users at a particular time and then bombard the PWA, the edge, and the cloud functions with HTTP requests at the determined amount of traffic.

It also made it possible to perform tests of request latency, error rates, and the system’s handling ability under pressure. Moreover, Prometheus, an open-source monitoring system, gathered such valuable data as the time required for function execution, CPU and memory consumption, and the number of requests per second. The other tool employed was Prometheus, which works in conjunction with Grafana to offer information and trends about the system's performance over some time. The affected workloads during testing were meant to model the real-life backend of a PWA, which includes AI-based and data-oriented tasks. Some workloads have been identified to include image recognition through deep learning models, data aggregation from various sources, and caching of commonly accessed API responses. Some of these workloads involved testing the computational workload we would normally run on centralized cloud environments to determine the efficacy of serverless edge computing. The results from these experiments were that there are specific ratios of latency and backend load by which the serverless PWA architecture benefits from a caching layer, depending on the system's scalability.

4.2 Performance Metrics

Several assessment criteria were used to determine if serverless edge computing is an appropriate solution strategy to decrease backend pressure and enhance efficiency. These comparisons were made on three architectures: OpenFaaS, AWS Lambda, and the back-end servers. [17-20] The performance measures evaluated were average response time, throughput, frequency of cold start, cost, number of successful transactions per successful backup, and reduction in backend load. This analysis is based on real-world experiments, which explain how this class of PWAs behaves depending on the infrastructure settings.

Table 1: Performance Comparison Table

Metric	Serverless Edge (OpenFaaS)	Cloud Serverless (AWS Lambda)	Traditional Backend
Avg. Response Time	128 ms	210 ms	450 ms
Throughput (req/s)	1,200	950	600
Cold Start Frequency	12%	8%	N/A
Cost Efficiency ($/GB)	$0.00023	$0.00032	$0.0015
Success Rate	89%	98%	95%
Backend Load Reduction	62% offloaded	38% offloaded	N/A

4.2.1 Response Time

One of the major benefits of serverless edge computing was its response time. Similarly, edge-based serverless functions, with sample applications created using OpenFaaS on Raspberry Pi cluster, it was realized that the average response time was 128ms, which is 39% less compared with cloud serverless (210ms) and 71% less compared with traditional backend (450 ms). This improvement was mainly because edge nodes were closer to users regarding network distances and, hence, shorter time to get and process the requests. Further, Cloudflare Workers got 33% better than AWS Lambda@Edge; this proves our earlier point that edge processing beats latency-accentuated applications.

4.2.2 Scalability

Scalability was another important aspect because PWAs could have varying workloads depending on the number of users. Both serverless edge and cloud serverless infrastructure offered better backend solutions than traditional systems as they adjusted to each load. The current OpenFaaS-based edge computing setup was about 1,200 req/s, and it outperformed AWS Lambda with 950 req/s and was significantly higher than traditional backends at a rate of 600 req/s. The enhanced scalability at the edges was due to a decentralized model in terms of resources whereby computation was a distributed function of all the edge nodes instead of depending on the cloud data centers.

4.2.3 Cost Efficiency

The temporal performance was estimated by analyzing the computing resources consumed per Gigabyte (GB) data. Edge computing was the most affordable, with $0.00023 per GB, while AWS Lambda cost $0.00032 per GB, and the normal back-end servers cost $0.0015 per GB. The factors supporting the cost benefits of edge computing included less data moving around for processing and the amount of money required to process this information. On the other hand, the structure of cloud serverless caused extra costs, especially related to the data transfer between the regions and centralized cloud services.

4.2.4 Backend Load Reduction

The proposed architectural approach was to decrease server-side computing and implement an efficient client-side caching strategy. Thus, the test results showed that edge computing limited the number of requests in the origin server by 62%, which lessened backend pressure. Service workers also cached the most frequently accessed assets to minimize the need to retrieve assets through APIs and, therefore, decreased the system's workload. Self-served solutions, while effective, resulted in a lesser improvement of 38 % in backend demands due to their centralized architecture.

5. Results and Discussion

Thus, the performance results clearly describe the differences and trade-offs between serverless edge computing and cloud-based serverless solutions. All of them have their strengths and weaknesses based on the goals of PWAs, including time to first byte, availability, costs, and decrease in backend traffic. In order to use these aspects, it is possible to evaluate how serverless edge computing helps improve PWA performance and identify the fields where cloud-based solutions are still more effective.

5.1 Edge vs. Cloud Trade-offs

Lower latency of OpenFaaS on the Raspberry Pi cluster with 128 ms compared to 210 ms for AWS Lambda. This enhancement was mainly a result of the accessibility to the clients, thereby avoiding several network jumps and dependence on centralized cloud-based computing. However, cloud platforms like AWS Lambda offered higher reliability, which was determined to be 98%, and in OpenFaaS, the reliability was 89 percent. This was mainly because the edge nodes here are comparatively smaller in scale than the large cloud structures, and at times, the availability of resources and failures in these edge nodes impact performance. The trade-off was cold start frequency. Cold starts were higher in edge nodes at 12% compared to AWS Lambda at 8 % and thus impacted the iat response times under low usage frequency. This issue emerged due to fewer warm instances in edge nodes: most nodes are characterized by restricted resources compared to auto-scaled cloud systems. Cloud providers use provisioned concurrency to cut down cold start times; however, with edge solutions, there is no equivalent level of optimization, and efforts like function preloading or container reuse are required to reduce it.

5.2 Cost Implications

The cost optimization analysis showed a considerable reduction in operation cost for mobile applications implemented using edge computing compared to those implemented using cloud serverless or traditional backends. Traditional system design running on this server had a problem of high capital investment on the infrastructure since the servers had to be on even when the business was receiving few clients. On the other hand, the serverless models work on an on-demand basis, which is more efficient when assuming irregular usage patterns in workloads. Edge computing also helped to continue cutting costs by handling data locally, which means that less data needed to be moved to and from the cloud to cause inordinate amounts of egress fees. For example, AWS Lambda incurs charges for outbound data transfer when using multiple regions. In contrast, edge jobs, including

Cloudflare Workers or OpenFaaS processes, operate computations nearer to the consumer, removing the need for additional transactions to the cloud. It improves the localization of processing and, when combined with serverless systems based at the edge, edges out other options for latency-bound applications concerning cost.

6. Security and Scalability Considerations

Concerning the establishment of serverless as one of the main architectural aspects and the importance of edge computing for Progressive Web Applications (PWAs), there are essential questions of security and scalability. While reducing backend load and increasing performance is great, secure and reliable data handling and the performance in different loads must be maintained to be trusted and make the users happy.

6.1 Security Challenges

6.1.1 Authentication and Data Privacy in Serverless PWAs

Security in serverless PWAs, but the most critical of them are related to authentication and data privacy. Currently, traditional system administrators separate security by perimeter because the architectures of serverless solutions are naturally event-based and mostly distributed. First, the serverless functions are directly connected to the users, their data, APIs, and databases, which means there is a potential for multiple exposures. Furthermore, edge-based functions exacerbate this problem because these functions execute outside of main security infrastructures in proximity to the user, which may elicit issues of data tampering, non-compliance, or intrusion. Anywhere the component is situated, there is a requirement for authentication, and this is quite complex, bearing in mind that the nodes can be distributed. Besides, processing and storage functions hosted at the edge level could contain or analyze a user's specific information, meaning that this transmission must be encrypted end to end and only permitted to an authorized set of users. A high probability of being attacked exists, such as an injection attack, API broken point, and privileges escalation, particularly in a multi-tenant cloud model.

6.1.2 Mitigation Strategies

A zero-trust security model is commonly advocated for in such a context. Thus, identity verification and access control are embedded into the application flow for every step instead of using the internal secure network as a protection factor. The serverless PWAs must utilize token-based authentication services (like OAuth 2.0 or JWTs) and API gateways with throttling, input validation, and logging features. There are scalable solutions for identity management, such as the cloud-native S3, IAM role, Azure AD, and Firebase Auth. Key features of secure edge computing include using secure containers, encrypted communication channels for end devices, and mechanisms for creating sandboxes and executing them at the network edge. Logging and monitoring can also be done; programs such as Prometheus or Loki can be used to flag any abnormal usage, or if there is any non-compliance or deviation from the defined strict security standards, runtime policies that can be examples are Open Policy Agents can be used to enforce it.

6.2 Scalability Aspects

6.2.1 Handling High-Traffic Scenarios

Serverless computing is scalable and well-aligned with the needs of the PWAs that may receive unpredictable traffic bursts. During such periods of traffic surge, for example, when releasing a new product or sensation, increasing numbers is critical. In this case, traditional computers and servers with their load-balanced architecture suffer from such load when the underlying infrastructure is not pre-provisioned, which results in compromised performance or failure. Serverless and edge-based architectures meet this requirement because functions are scaled based on current executing instances, as well as the occurrences of events. For instance, AWS Lambda and Google Cloud Functions can scale horizontally within a millisecond and guarantee improved performance as more users are added. Similar to the Cloudflare Workers and OpenFaaS, at the edge, lightweight containers and event-driven invocation enable one to quickly supply localized traffic demands without overwhelming the main servers.

6.2.2 Auto-scaling Capabilities of Serverless and Edge Solutions

Auto-scaling is supported in cloud and edge serverless solutions; however, the implementation method differs. Functions are scaled almost linearly in the case of cloud environments, as there is virtually no limit (depending on the provider) for the amount of data centers with enough processing power. This enhances the suitability and uniformity of cloud serverless systems to handle global connections. Nonetheless, issues with the scalability of latency and cold starts may occur at unpredictable and short notice when loaded with significant traffic.

Edge-based solutions provide faster scaling, while the latency turns out to be lower, especially for geographically dispersed requests. Through such functions being provided closer to the client, edge nodes can act instantly, eliminating bottlenecks arising from the distances covered. Moreover, the geographical decentralization of edge computing helps to avoid centralization since computational tasks are settled in different nodes. Therefore, to make scaled-up even more manageable, there

are Horizontal Pod Autoscalers to scale pods, such as Kubernetes, Knative, and OpenFaaS Gateway. They are tools that check the usage of resources and the functioning of a certain function through the CPU load, number of requests, and others, automatically scaling if these predefined values are exceeded. These features of serverless PWAs with smart caching and CdNs make it possible to easily get high availability and performance even at cloudy moments in case of traffic bursts.

7. Future Trends

The role of serverless systems and edge technologies remains the possible future of Progressive Web Applications (PWAs) web applications' architecture. New trends depict a future scenario with closer relations between computing, intelligence, and user access for applications' built-in features of extreme response, security, and intelligence. Here are some important areas of development of the field.

7.1 AI at the Edge

AI at the Edge, which is considered one of the most prominent trends at the moment, refers to the usage of AI and ML models at the edge of the network. Frameworks such as TensorFlow Lite and ONNX ensure that deep learning scenarios such as image recognition, predictive analysis, and recommendation can now be extended easily to the edge nodes. This significantly decreases the usage of the cloud-based inference engine, which has the benefits of less power consumption and better user privacy because no data ever leaves the device. Such use cases include voice-based shopping assistants, content personalization in real-time, and smart camera interfaces, among others, that can be implemented on edge serverless functions. These edge scenarios also help decrease the load on the backend, as most of the calculations are performed right at the device and in the background.

7.2 Federated and Decentralized Architectures

Distributed computing federations where the computations are processed both on the user equipment and on cooperative edge nodes. This decentralization is more appropriate in privacy-preserving applications and systems and in scenarios where the end devices may periodically go offline. Web assembly and service mesh architectures like Istio or Linkerd allow the development of microservices that can run securely and at top performance on web browsers through the gateway on edge. Some decentralized serverless architectures, sometimes termed functional computing over peer-to-peer networks, make it possible for PWAs to work even in regional cloud failures. Such models could redefine how applications work in the offline mode, data synch, and how conflicts are resolved, especially in those new frontiers or developing countries.

7.3 Enhanced Observability and Developer Tooling

Serverless computing is also experiencing similar growth in observability and debugging, which are essential for monitoring the effectiveness of distributed systems. Modern approaches include monitoring and diagnostics that allow tracing, performance analysis, and the detection of deviation in distributed applications on edge and cloud. Frameworks such as OpenTelemetry, Grafana, and Jaeger are being included directly into the serverless frameworks to allow developers to monitor the requests' flows and the functions' behavior. At the same time, low-code and no-code environments are gradually gaining serverless and edge functionality possibilities. This means it becomes easier for developers to integrate such options and does not take much time. It will cause the development of many new PWAs that can self-scale, self-configure, and self-adapt to diverse environments.

8. Conclusion

Serverless computing technology with edge technologies is a new way of developing Progressive Web Applications with high performance, scalability, and cost-effectiveness. Whenever computation is shifted to serverless cloud functions and precise processing is performed at the proximity of a network link, the backend burden can be substantially lessened, latency can be decreased, and the end-user experience can be improved. The features such as dynamic scaling, event-based triggering, and the pay-as-you-go approach successfully implemented in AWS Lambda, Firebase Functions, and OpenFaaS make these architectures suitable for modern web applications with fluctuating traffic and users worldwide. Because of the architecture's flexibility, PWAs do not stop being responsive even when subjected to high loads of traffic or restricted connections. Though cold starts, data privacy and coordination of the necessary resources are a few significant issues that need to be addressed, and the various platforms for orchestration, monitoring, and security tools are emerging quickly. These and other developments of Edge AI and other future trends in computing, as well as the implementation of federated computing and intelligent caching, are going to pave the way for the adoption of serverless PWAs as the new frontier for fast, reliable and adaptive digital services in the global market.

References

[1] Aslanpour, M. S., Toosi, A. N., Cicconetti, C., Javadi, B., Sbarski, P., Taibi, D., ... & Dustdar, S. (2021, February). Serverless edge computing: vision and challenges. In Proceedings of the 2021 Australasian Computer Science Week multiconference (pp. 1-10).

[2] Kjorveziroski, V., Filiposka, S., & Trajkovik, V. (2021). IoT serverless computing at the edge: A systematic mapping review. Computers, 10(10), 130.
[3] Gadepalli, P. K., Peach, G., Cherkasova, L., Aitken, R., & Parmer, G. (2019, October). Challenges and opportunities for efficient serverless computing at the edge. In 2019 38th Symposium on Reliable Distributed Systems (SRDS) (pp. 261-2615). IEEE.
[4] Chaudhry, S. R., Palade, A., Kazmi, A., & Clarke, S. (2020). Improved QoS at the edge using serverless computing to deploy virtual network functions. IEEE Internet of Things Journal, 7(10), 10673-10683.
[5] Baresi, L., & Quattrocchi, G. (2021). PAPS: A serverless platform for edge computing infrastructures. Frontiers in Sustainable Cities, 3, 690660.
[6] Javed, H., Toosi, A. N., & Aslanpour, M. S. (2022). Serverless platforms on the edge: a performance analysis. In New Frontiers in Cloud Computing and Internet of Things (pp. 165-184). Cham: Springer International Publishing.
[7] Sbarski, P., & Kroonenburg, S. (2017). Serverless architectures on AWS: with examples using Aws Lambda. Simon and Schuster.
[8] Rajan, A. P. (2020). A review on serverless architectures-function as a service (FaaS) in cloud computing. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(1), 530-537.
[9] Golec, M., Ozturac, R., Pooranian, Z., Gill, S. S., & Buyya, R. (2021). IFaaSBus: A security-and privacy-based lightweight framework for serverless computing using IoT and machine learning. IEEE Transactions on Industrial Informatics, 18(5), 3522-3529.
[10] Singh, P., Masud, M., Hossain, M. S., Kaur, A., Muhammad, G., & Ghoneim, A. (2021). Privacy-preserving serverless computing using federated learning for smart grids. IEEE Transactions on Industrial Informatics, 18(11), 7843-7852.
[11] Schuler, L., Jamil, S., & Kühl, N. (2021, May). AI-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. In 2021 IEEE/ACM 21st international symposium on cluster, cloud and internet computing (CCGrid) (pp. 804-811). IEEE.
[12] Costanzo, G. T., Zhu, G., Anjos, M. F., & Savard, G. (2012). A system architecture for autonomous demand side load management in smart buildings. IEEE transactions on smart grid, 3(4), 2157-2165.
[13] Hajian, M. (2019). Progressive web apps with angular: create responsive, fast, and reliable PWAs using angular. Apress.
[14] Cao, K., Hu, S., Shi, Y., Colombo, A. W., Karnouskos, S., & Li, X. (2021). A survey on edge and edge-cloud computing assisted cyber-physical systems. IEEE Transactions on Industrial Informatics, 17(11), 7806-7819.
[15] Ferrer, A. J., Marquès, J. M., & Jorba, J. (2019). Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. ACM Computing Surveys (CSUR), 51(6), 1-36.
[16] Bardsley, D., Ryan, L., & Howard, J. (2018, September). Serverless performance and optimization strategies. In 2018 IEEE International Conference on Smart Cloud (SmartCloud) (pp. 19-26). IEEE.
[17] Vahidinia, P., Farahani, B., & Aliee, F. S. (2020, August). Cold start in serverless computing: Current trends and mitigation strategies. In 2020 International Conference on Omni-layer Intelligent Systems (COINS) (pp. 1-7). IEEE.
[18] Bakshi, K. (2017, March). Microservices-based software architecture and approaches. In 2017 IEEE aerospace conference (pp. 1-8). IEEE.
[19] Aksakalli, I. K., Çelik, T., Can, A. B., & Tekinerdoğan, B. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. Journal of Systems and Software, 180, 111014.
[20] Silva, P., Costan, A., & Antoniu, G. (2019, December). Investigating edge vs. cloud computing trade-offs for stream processing. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 469-474). IEEE.
[21] Sengupta, A., Tandon, R., & Simeone, O. (2016, July). Cloud RAN and edge caching: Fundamental performance trade-offs. In 2016, IEEE 17th International workshop on signal processing advances in wireless communications (SPAWC) (pp. 1-5). IEEE.
[22] Yuan, D., Yang, Y., & Chen, J. (2012). Computation and Storage in the Cloud: Understanding the Trade-offs. Newnes.