International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

COST-OPTIMISED SERVERLESS ARCHITECTURE: DESIGN PATTERNS FOR EFFICIENT ENTERPRISE BACKEND WORKLOADS

Sai Viswa Teja Arumilli
Independent Researcher

Abstract

This research study creates an efficient serverless architecture of enterprise backend workload based on the Microsoft Azure Functions trace data in a cost-saving manner. The data on invocation counts, execution times and memory allocation collected was compiled, analyzed and converted into resource-efficiency indicators. Random Forest, Gradient Boosting and XGBoost classifiers forecasted like low and medium and high future cost. XGBoost performed the best with an accuracy of 90.73% and GB-second proved to be the winner. The results indicate that serverless cost reduction and efficiency can be facilitated by workload analytics, memory right-sizing, runtime optimization, load levelling and adaptive scaling.

Keywords: Serverless Computing, Cost Optimisation, XGBoost, Resource Efficiency, Azure Functions, Workload Analytics

I. INTRODUCTION

Background

The concept of Serverless computing has revolutionized the way backends of enterprises are developed by providing the benefit of automatic scaling, less infrastructure management and charging on usage basis. Nonetheless, a poor design of functions, high invocations, and long execution time and poorly configured resources, can greatly inflate operational expenses [1]. Performance, scalability, reliability and financial efficiency under complex enterprise workloads. This facilitates the balancing of performance, scalability, reliability and financial effectiveness in a cost-optimized architectural pattern across complexity enterprise workloads in the modern cloud-at-scale sustainably.

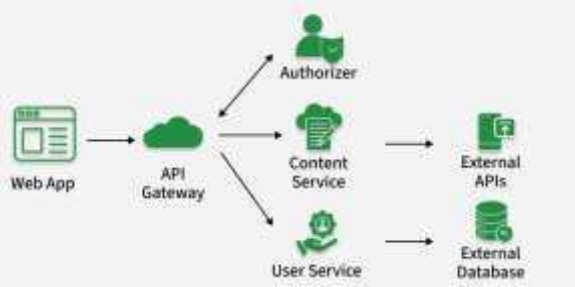

Fig. 1: Serverless Architecture

Research Problem

Although serverless computing is much easier to scale and operate differently, most businesses suffer because their costs are not predictable due to poor execution of functions, over allocation of resources, cold-start delays and fragmented architecture practices [2]. Present literature is not very specific on cost-conscious design patterns, which would collaboratively co-optimize financial economy, system performance, reliability and workload scalability demands on an enterprise.

Research Aim and Objectives Aim

This research aims to offer and test cost conscious patterns of serverless design, that can enhance effectiveness, performance, scaling and cost predictability of enterprise workloads on backends across clouds.

Objectives

- ***To examine cost drivers in backends serverless workloads in a variety of cloud environments.***
- ***To identify architectural patterns that reduce the waste of resources, execution time and operational cost.***
- ***To evaluate the patterns suggested in terms of performance, scalability, reliability and cost-efficiency metrics.***
- ***To develop practical guidelines that support the cost-conscious adoption of serverless and proper deployment of the enterprise-wide back-end.***

Novel Contribution

The research study suggests a comprehensive implementation of a cost conscious serverless design method, which relates architectural models with quantifiable cost and performance fundamental results [3]. It considers optimization techniques as a set, unlike the current methods that study them individually, resulting in evidence-based recommendations of efficient, reliable, predictable, and sustainable cloud backend operations in enterprises.

II.LITERATURE REVIEW

Cost Drivers in Serverless Enterprise Backend Workloads

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

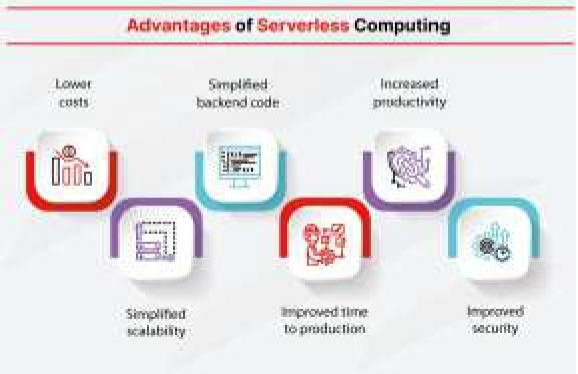

Fig. 2: Serverless Computing Architecture

Invocation frequency, execution duration, allocated data, data transfer and the storage as well as the outer service calls have consistently been found to be the key contributors to cost [4]. Research however does not agree on how much they should weigh as the pricing models of providers as well as their workload profiles vary greatly [5]. Experimental studies provide accurate measurements, and often use short lived synthetically generated workloads that are unlikely to reflect the complex enterprise traffic [6]. On the other hand, industrial case studies are real life and reflect the complexity of the operations though have low reproducibility and tend to hide the commercial information [7]. Cost is also largely analyzed with the majority of analyses considering cost as a standalone result of a result independent of both latency and reliability as well as scaling behavior [8]. Therefore, cross-cloud evidence cannot be said to be adequate in the explanation of the difference in costs of heterogeneous backend workloads [9].

Cost-Aware Architectural Patterns for Serverless Resource Optimisation

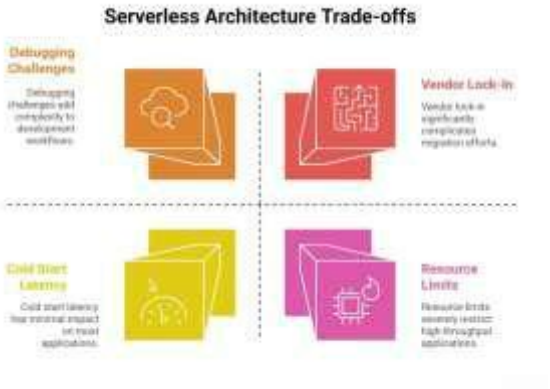

Fig. 3: Serverless Architecture Trade offs

The current literature suggests right-sizing, asynchronous messaging, the fusion of functions, caching, batching, and function concurrency, and function event filtering to minimize waste [10]. Right-sizing reduces overprovisioning, whereas fusion reduces the invocation overhead, however, both could cause a rise in latency, coupling, or deployment complexity [11]. Caching and batching can be used to make workloads efficient on repeated workloads, but is less effective with irregular traffic or strict freshness conditions [12]. Provider-specific optimization is a more preferred method of some researchers due to its exploitative nature on the native services whereas portable patterns is the preferred method on others due to absence of vendor dependence [13]. The former shows more immediate efficiency, whereas the latter provides flexibility in strategies [14]. The comparisons are still weak since the studies are conducted with varying architectures, periods of pricing and varying baselines and do not help in revealing the best patterns [15].

Performance Evaluation Metrics for Efficient Serverless Architectures

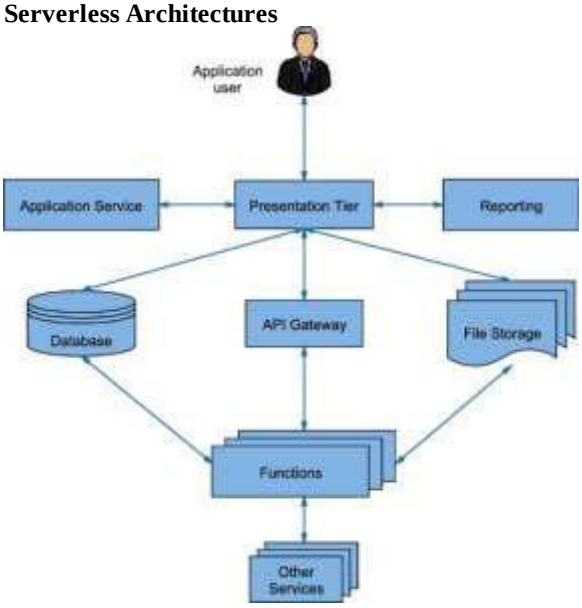

Fig. 4: Resource optimization in performance modeling for serverless application

Response time, throughput, frequency of cold-start, availability, resource utilization and the cost of money are typically evaluation metrics [16]. Despite the seemingly all-inclusive definition of these metrics, the researchers use different definitions, workload creators, monitoring timeframes, and clouds [17]. Benchmark experiments help to make a controlled comparison, but in most cases, failures are disregarded, traffic volatility, and long-term changes in pricing are ignored [18]. Production examinations enhance the ecological legitimacy yet seldom single out causal consequences of individual patterns [19]. Furthermore, tail delays can be hidden under average latency whereas costly workload peaks can be hidden

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

under aggregate cost [20]. Multi-objective methods are more accurate representations of the enterprise trade off, and often they weight subjectively the performance and cost [21]. This means that, there is no standardized framework as yet that will enable fair and reproducible cross-cloud pattern evaluation [22].

Practical Guidelines for Sustainable Cost-Aware Serverless Adoption

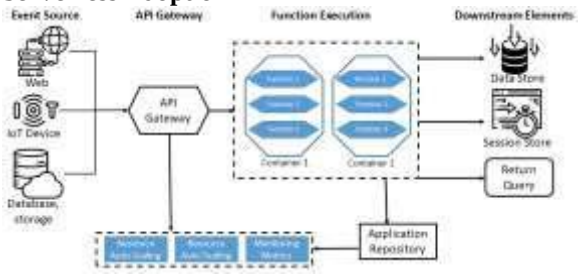

Fig. 5: Reducing Environmental Impact with Sustainable Serverless Computing

Governance, observability, financial accountability, skills, security, and sustainability are some of the focus areas highlighted in the adoption literature in addition to technical optimization [23]. Recommendations Practitioner frameworks contain actionable recommendations, and many of them are created by vendors of clouds, and are possibly biased towards proprietary solutions [24]. Academic models have fewer conceptual dependencies, albeit often too abstract to be operationalized to make a decision [25]. Research FinOps boosts cost ownership by tagging, budgets and monitoring, but tends to respond to spending once it has gone live, as opposed to influencing architecture in advance [26]. Sustainable studies also advance efficient utilization of the resources, however, hardly relate the carbon results with dependability and the business necessity [27]. The synthesis of these streams shows the necessity of provider-neutral guidelines which are able to connect the design choices, trade-offs, and governance [28].

TABLE 1: COMPARATIVE ANALYSIS OF COST-AWARE SERVERLESS DESIGN APPROACHES FOR ENTERPRISE BACKEND WORKLOADS

Approach	Model Equation	Parameters	Advantages	Limitations	Applicability	Effectiveness
Memory Right-Sizing	(C=I(TMP_c)+IP_i)	(I): invocations; (T): runtime; (M): memory; (P_c): compute price; (P_i): invocation price	Reduces overprovisioning and GB-second consumption	Insufficient memory may increase execution time and failures	Memory-intensive and stable workloads	High when memory and runtime are balanced
Function Batching	(I_b=\lceil I/b\rceil)	(I_b): batched invocations; (b): batch size	Reduces invocation frequency and processing overhead	May increase latency and processing complexity	High-volume, short-duration workloads	High for repetitive event processing
Function Fusion	(C_f=\sum C_j-O_r)	(C_j): function cost; (O_r): removed orchestration overhead	Minimises communication, invocation and orchestration costs	Increases coupling and reduces functional independence	Sequential microservice workflows	Moderate to high for stable workflows

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Caching Strategy	(C_t=(1-h)C_e+C_{cache})	(h): cache-hit rate; (C_e): execution cost; (C_{cache}): caching cost	Reduces repeated computation and response time	Creates consistency, storage and invalidation challenges	Read-heavy workloads with repeated requests	High when cache-hit rates remain strong
Predictive Scaling	(K=\lceil\hat{\lambda}/\mu\rceil)	(\hat{\lambda}): predicted demand; (\mu): processing capacity; (K): instances	Anticipates workload bursts and reduces latency	Forecasting errors may cause under-scaling or waste	Variable and burst-oriented enterprise workloads	High with accurate demand forecasting
Machine Learning Recommendation	(s^*=\arg\min_s(\alpha C_s+\beta L_s+\gamma R_s))	(C_s): cost; (L_s): latency; (R_s): reliability risk; (\alpha,\beta,\gamma): weights	Selects patterns using multiple workload characteristics	Requires representative data, tuning and explainability	Complex enterprise serverless environments	High when validated across diverse workloads
Cross-Cloud Cost Evaluation	(CR=\frac{C_b-C_p}{C_b}\times100)	(C_b): baseline cost; (C_p): pattern cost; (CR): cost reduction	Enables comparable assessment across providers and patterns	Pricing differences may restrict direct comparisons	Multi-cloud deployment and migration decisions	Strong for provider-neutral evaluation

Literature Gap

Existing research investigate serverless cost drivers and disaggregated optimization trends but not a cohesive, provider-neutral workloads platform [30]. There are still evidence gaps and gaps in the synthetic benchmarks, discrepant metrics and single-cloud research [31]. There is therefore little whole course assessment between cost, performance, scalability, reliability, sustainability and governance to offer valuable cross cloud guidance on the decision on deploying clouds to an enterprise.

III. METHODOLOGY

Research Design

The research took the quantitative design with secondary-data analytics to assess the patterns of cost-optimized serverless architecture to enterprise back-end workloads. Analysis was concentrated on operation work load behavior, performance efficiency, memory and resource utilization and the future classifications of costs [32]. The reason behind the choice of secondary data has to do with their ability to deliver mass real-world traces, without having to conduct primary experiments.

Data Collection and Preparation

Data source link: ***https://github.com/Azure/AzurePublicDataset***

The data is acquired in the open source GitHub repository of Microsoft Azure, AzurePublicDataset. The Azure Functions Dataset 2019 is a dataset of anonymized production traces of Azure Functions in July 2019 and has numbers of calls to functions, individual trigger groups, distribution of execution times, and distributions of application level memory-allocation [33]. Applications, according to Microsoft, are units of resource allocation and the measurement of memory is at application level.

The CSVs are extracted and loaded into Python and unified on an application-day basis by use of

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

HashOwner, HashApp, and identification of days [34]. Traces of invocation were summarized into 1440 minute-level traces to get the total invocations, active minutes, the peak hour invocation and the burstiness of the workload. The aggregation of duration records has been done based on the number of executions and memory values have been transformed into gigabytes and megabytes respectively [35]. Where necessary missing and infinite values were removed or median-imputed. The analysis data ended up having 205,187 observations.

Exploratory Data Analysis

Invocation volume, counts of functions, execution time, memory usage, burstiness, resource use and efficiency have descriptive statistics performed on them. Pre-predictive modelling Distribution, time-based, resource consumption, and scatter plots and a heatmap looking at the skewness, variation, outliers and relationships between serverless cost drivers are used to identify and isolate skew, daily and outlier variation, and interrelations between the serverless cost drivers.

Spearman Correlation:

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$$

where r_s is the Spearman correlation coefficient, d_i is the difference between paired ranks and where n is the difference between paired ranks in the rank 1 and 2, and n is the total number of observations.

Feature Engineering

There is feature engineering that is used to convert raw traces into operational indicators that could be used to analyze serverless costs. To represent the compute, workload, and utilization behaviors, resource consumption, invocation intensity and resource efficiency were derived. Operation variables have been designed to model the efficiency of serverless workload. A GB-seconds proxy is used to estimate the consumption of the resources. GB-Seconds Resource Consumption:

In which GBS (GB-seconds) is resource usage on the GB, TCS is total seconds of compute and AMG is average memory allocated per GB are calculated values in gigabytes.

Invocation Intensity:

In which II is the intensity of invocation, TI is the total invocations and NF is the count of serverless functions.

Resource Efficiency:

RE can be used to represent resource efficiency, TI total invocations, GBS derived as GB-seconds resource-consumption proxy. A next-day cost target is then categorized as a low, medium and high-cost target (based on the 33rd percentile and 67th percentile training data).

Machine Learning Modelling

Three ensemble models which are used under the supervision were random forest, gradient boosting, and XGBoost. Random Forest is a combination of predictions of a number of decision trees. Random Forest Prediction:

$$\hat{Y} = \{h_1(X), h_2(X), \ldots, h_T(X)\}$$

where $\hat{Y}$ refers to the cost category forecasted, $h_t(X)$ is the forecast made by an individual decision tree, X is the input features and T is the number of trees.

Gradient Boosting made a series of progressively better forecasts by alleviating the errors made by an earlier model.

Gradient Boosting Update:

$$F_m(X) = F_{m-1}(X) + h_m(X)$$

$F_m(X)$ is the new model, $F_{m-1}(X)$ is the last model, η is the learning rate and $h_m(X)$ is the new decision tree that is added.

XGBoost reduced the error of prediction and regularized.

XGBoost Objective Function:

$$= (\quad) + (\quad)$$

where L denotes prediction loss and y_i denotes actual classes, $\hat{y}_i$ denotes prediction classes, and $\Omega(f_k)$ denotes regularization of model-complexity.

Model Training and Evaluation

The training and test split is split based on a time-based analysis of 70:30 where the previously observed workloads are used to predict the following categories of costs. Accuracy, precision, recall, F1-score, balanced accuracy and confusion matrices were used to test the models.

$$= \frac{}{+}$$

TP denotes True Positive, TN is True Negative, FP is False Positive, FN is False Negative. Selecting feature-importance analysis is also used as a measure to determine workload, execution-time, memory and resource efficiency variables with the highest contribution to the future classification of serverless costs.

Architecture Diagram

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

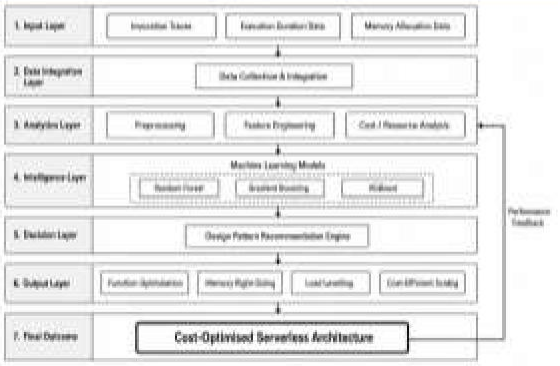

Fig. 6: Architecture Diagram

The architectural framework combines workload traces, preprocessing, feature engineering, cost analysis, machine-learning models and recommendation mechanisms to facilitate memory right-sizing, load levelling, workload function optimization, scalable execution and performance feedback loops.

Flowchart Diagram

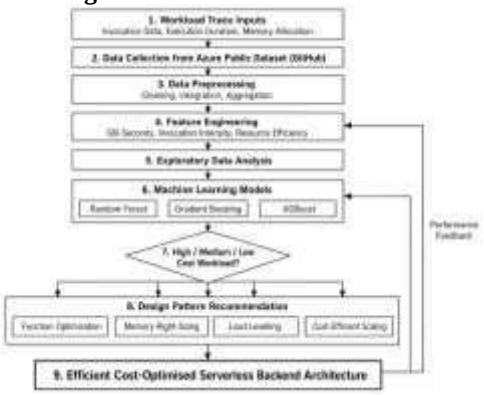

Fig. 7: Flowchart diagram

The flowchart shows a seamless process of the Azure workload collection and preprocessing, all the way to feature engineering, exploratory analysis, machine-learning classification, cost categorization, design-pattern recommendation, and a feedback loop to ensure efficient serverless optimization.

Pseudocode

```
CODE
Program to optimise cost-efficient serverless backend workloads
    Initialise
        Inputs
            Invocation data
            Execution duration
            Memory allocation
        Variables
            GB_Seconds
            Invocation_Intensity
            Resource_Efficiency
            Cost_Class
    Start process
        Clean and integrate workload data
        GB_Seconds = Compute_Time * Memory_GB
        Invocation_Intensity = Total_Invocations / (Functions + 1)
        Resource_Efficiency = Total_Invocations / (GB_Seconds + 1)
        Train Random Forest
        Train Gradient Boosting
        Train XGBoost
        Select best performing model
        Predict Cost_Class
        IF Cost_Class = High THEN
            Apply Function Optimisation
            Apply Memory Right-Sizing
        IF Workload = Bursty THEN
            Apply Load Levelling
        IF Cost_Class = Medium THEN
            Apply Cost-Efficient Scaling
        IF Cost_Class = Low THEN
            Maintain Standard Serverless Pattern
        Evaluate Accuracy, Precision, Recall and F1
        Update optimisation strategy using performance feedback
    End process
```

Fig. 8: Pseudocode

The pseudocode describes various stages of data preparation, feature engineering, training machine-learners, and cost allocation and guidance on the architecture, allowing the serverless workloads to be optimized adaptively by provisioning the memory size, load balancing, scaling, and performance feedback.

IV. RESULT AND DISCUSSION

Fig. 9: Integrated Azure Functions Dataset Preview

A preview of the dataset indicates that there was indeed effective integration of 25 variables between records of application-days. In the original application, the total number of invocations per day 1-5 differ with a mean of between 6 and 20 and same number of functions per day numbering between 1 and 6, indicating a significant workload change over time.

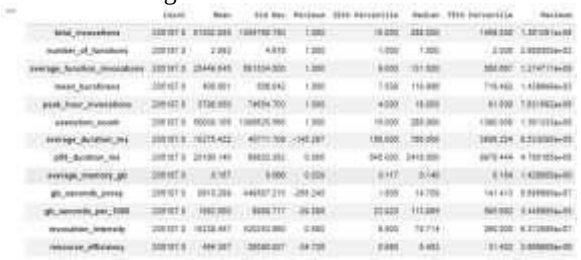

Fig. 10: Summary Statistics of Serverless Workload Variables

In 205,187 observations, the total invocations average 51,002 but have a median only 288 with the maximum above 136 million, which proves the extreme skewness. The average memory is 0.157 GB, a median resource usage of 14.709 GB-seconds and substantiates powerful transformation in the process of modelling.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

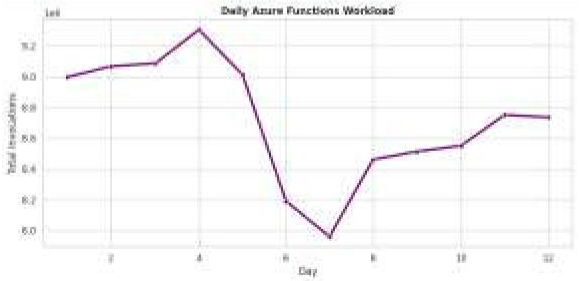

Fig. 11: Daily Azure Functions Workload

Daily workload is constant but fluctuates daily over a course of twelve days. Experts increase to an approximate of 899million on day 1 to a peak of around 930 million on day 4, after which it decreases drastically to an approximate of 796million on day 7 and thereafter.

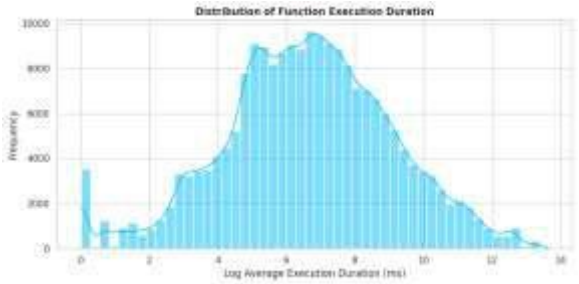

Fig. 12: Distribution of Function Execution Duration

The execution time has overall, right-tailed distribution when transformed logarithmically. Its maximum frequency of about 9,500 observations can be found between log-duration of 6.5-7.0 and the values spread to nearly 0 to nearly 14, indicating a large amount of heterogeneity in the execution of all functions which should be optimized using cost-conscious functions optimization methods and prediction.

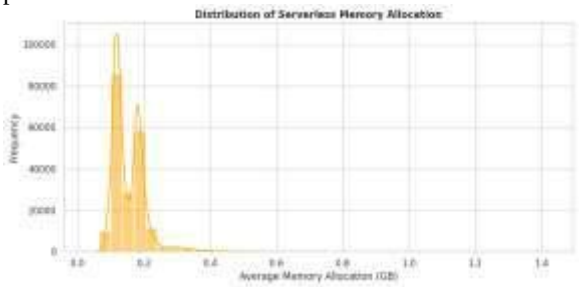

Fig. 13: Distribution of Services Memory Allocation

The distribution of memory has a strong concentration on low levels with prominent peaks of 0.12 GB, 0.18 GB. The majority of loads are under 0.30 GB, with the occasional application nearly hitting 1.4 GB, which is sufficient to right-size memory consumption since the atypical workloads with high memory consumption can skew consumption variation.

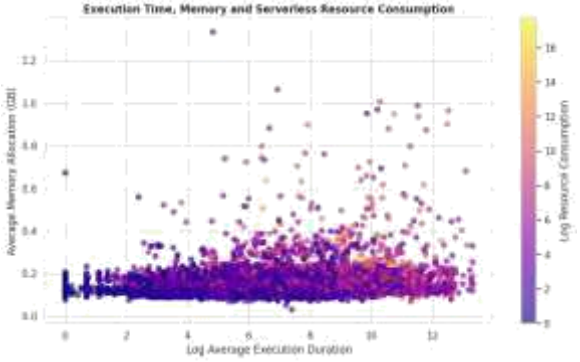

Fig. 14: Execution Time, Memory and Serverless Resource Consumption

Most of the applications are concentrated round a value of 0.10-0.30 GB memory and 3-11 log execution times as indicated by the scatterplot. Colors that require higher consumption of resources, up to about 12 -17 log units, are predominantly presented at longer run times and higher memory, which are useful to make joint runtime-memory optimization decisions on workloads.

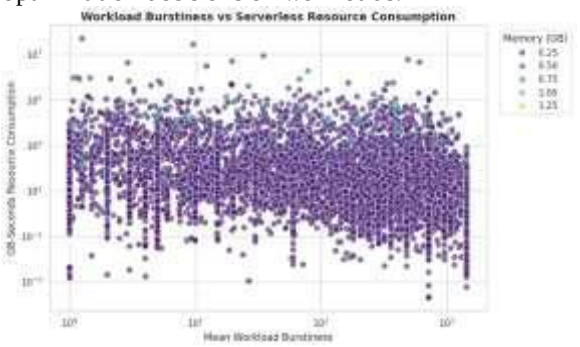

Fig. 15: Workload Burstiness vs Serverless Resource Consumption

Workload burstiness spans roughly 1 to above 1,000, while resource consumption ranges from below 10^-3 to above 10^7 GB-seconds. The dispersed pattern indicates no simple positive relationship; extreme resource values occur across burstiness levels, supporting multidimensional rather than burstiness-only optimization.

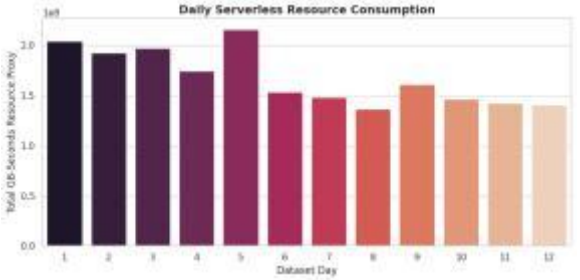

Fig. 16: Daily Serverless Resource Consumption

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Day 5 has the highest levels of resource consumption at about 2.15×10^8 GB-seconds as compared to 2.03×10^8 on day 1. It decreases to an approximate of 1.36×10^8 on day 8, however it reverses to a degree showing that the costs change significantly with time and thus, adaptive optimization is necessary over the days seen.

Fig. 17: Correlation of Serverless Cost and Workload Drivers

The heatmap shows that there are strong correlations between costs, total invocations are 0.63 with GB-seconds, average duration is 0.99 with GB-seconds per 1000 executions, resource efficiency is -0.93 with the result. These scales vigorously support cost optimization and predictive modeling strategies that are based on runtime.

Fig. 18: Random Forest Classifier Configuration

The random forest model is set to 500 trees, depth 18, identification 4 and the leaf size 2. Equal weighting of classes is used to solve unequal cost classes whereas parallel processing and random state 42 enhance efficiency and reproducibility; performance needs test measurements.

Fig. 19: Gradient Boosting Classifier Configuration

The Gradient Boosting classifier has the following settings of estimators' number of 300, learning rate of 0.05, max depth of 4, random state of 42. Such settings have a good balance between learning rate and model complexity that facilitates steady sequential error-correcting information as well as overfitting control by predicting serverless cost-category in practice reliably.

Fig. 20: XGBoost Classifier Configuration

The XGBoost classifier is trained using 600 estimators, a learning rate of 0.03, a maximum depth of 7, the minimum child weight of 3, a subsample of 0.85 and column sampling of 0.85. This regularized setup enhances the learning of nonlinear patterns and minimizes over-fitting among serverless workload characteristics effectively when making predictions as reliable as possible.

	Model	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy
0	Random Forest	0.905928	0.906022	0.905928	0.905963	0.906296
1	Gradient Boosting	0.906897	0.906910	0.906897	0.906904	0.907164
2	XGBoost	0.907291	0.907381	0.907291	0.907323	0.907565

Fig. 21: Machine Learning Model Performance Comparison

Comparison on models indicates that its performance is always high over 90%. XGBoost has the highest accuracy of 90.73%, equal accuracy of 90.76% and has an F1-score of 90.73% at the expense of Gradient Boosting with 90.69% accuracy and the Random Forest with 90.59% accuracy, confirming XGBoost use to draw deployment.

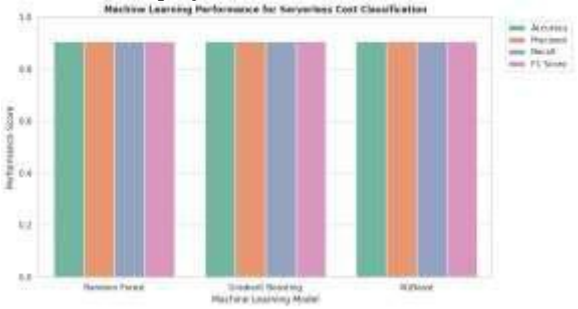

Fig. 22: Machine Learning Performance Comparison for Serverless Cost Classification

The chart demonstrates that there is high classification performance in all the three ensemble models. XGBoost has a slight lead at the 90.73% accuracy and F1-score and Gradient Boosting and random forest at 90.69% and 90.59%, respectively with confirmatory results that the serverless cost classification has good overall performance with stable and reliable results.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

	precision	recall	f1-score	support
Low Cost	0.9121	0.9207	0.9164	21657
Medium Cost	0.8688	0.8688	0.8688	22370
High Cost	0.9419	0.9332	0.9375	22029
accuracy			0.9073	66056
macro avg	0.9076	0.9076	0.9076	66056
weighted avg	0.9074	0.9073	0.9073	66056

Fig. 23: XGBoost Classification Report

The classification report establishes the strength of class-levels performance in 66,056 observations in tests. Low Cost achieves 91.64% F1, Medium Cost 86.88%, and High Cost 93.75%. The overall accuracy is 90.73% and weighted precision, recall, and F1 are about 90.7%, which indicates good consistency in the different categories.

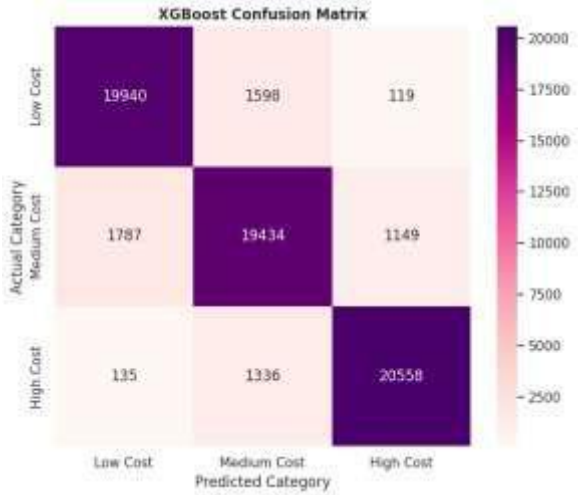

Fig. 24:XGBoost Confusion Matrix

The confusion matrix of XGBoost reveals that 19, 940 Low, 19, 434 Medium and 20, 558 High Cost cases were correctly classified. There is maximum misclassification between Medium and Low which is 1,787 and 1,598 respectively which point towards adjacent-class overlap prediction.

	Feature	Importance
12	gb_seconds_proxy	0.647108
8	total_compute_seconds	0.150573
0	total_invocations	0.033496
7	execution_count	0.031907
13	gb_seconds_per_1000	0.018971
15	resource_efficiency	0.014652
5	mean_burstiness	0.013294
10	p99_duration_ms	0.012955
4	average_active_minutes	0.012250
6	peak_hour_invocations	0.011875

Fig. 25: Serverless Cost Feature Importance

The GB-seconds proxy is the most significant predictor with a value of 0.6471 with total compute seconds coming second with a value of 0.1506. In invocation, the total invocation has a contribution of 0.0335 and the number of executions 0.0319, and the rest of the variables contribute less than 0.019 and hence the consumption of resources is a strong indicator of the future cost classification.

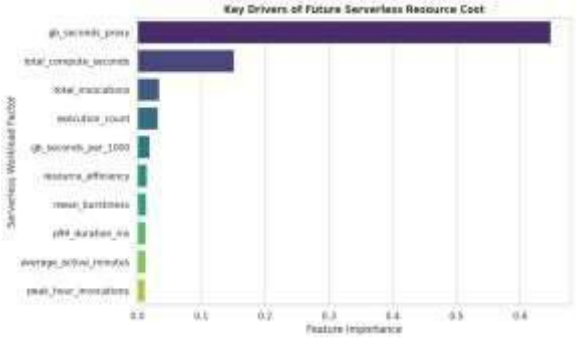

Fig. 26: Key Drivers of Future Serverless Resource Cost

The feature-importance chart has been used to visually validate the dominance of GB-seconds proxy in predictors at about 64.7% which is by far more than the total compute seconds at 15.1%. The contribution of invocation volume and the execution count is about 3.3% and 3.2%, respectively, which is why resource-consumption optimization is the key architectural focus in general.

TABLE 2: SUMMARY OF KEY RESULTS

Result Area	Key Finding

Random Forest	90.59% accuracy
Gradient Boosting	90.69% accuracy
XGBoost	**90.73% accuracy**
Best Model	XGBoost
Balanced Accuracy	90.76%
Low-Cost F1	91.64%
Medium-Cost F1	86.88%
High-Cost F1	93.75%
Top Predictor	GB-seconds proxy
Feature Importance	64.71%
Second Predictor	Compute seconds
Second Importance	15.06%
Main Insight	Resource use drives cost
Recommended Patterns	Right-sizing, scaling, optimization

Discussion

The results portray that the compute consumption, execution duration and invocation intensity are the key factors that determine serverless cost behavior. XGBoost has the highest predictive performance of 90.73% accuracy and 90.76% balanced accuracy which is a bit higher than Gradient Boosting and Random Forest. This consistency makes it clear that ensemble learning can be effective to classify cost categories of workloads. The feature importance also established GB-seconds proxy as the most important predictor (64.71%) then total compute seconds at 15.06% as the major predictor in terms of central tendency in resource consumption to cost optimization. The findings thus support memory right-sizing, runtime optimization and adaptive scaling as viable architecture design patterns to derive an efficient enterprise serverless architecture.

V. CONCLUSION AND FUTURE DIRECTION

Conclusion

The research finds that workload analytics, engineered resource metrics, and machine-learning ensemble make a good fit to support cost-optimized serverless architecture. XGBoost provided the best classification and GB-seconds proved to be the cost predictor. These results validate the idea that memory right-sizing, the ability to run applications on any platform, load balancing, and adaptive scaling offer useful design patterns to enhance the efficiency of serverless services and the ability to regulate resource usage.

Future Direction

Future studies are advised to expand the framework based on more up-to-date multi-cloud serverless trace data, real-time billing information, and the workload produced by telemetry. The dynamic optimization of deep learning, online learning, and reinforcement learning should be assessed in studies. The remaining areas addressed should be tested with cold-start effects, carbon-conscious scheduling, security policies, and cross-provider portability, and recommendation patterns should be validated by testing them in the context of longitudinal enterprise deployments to test scalability, robustness, and long-term cost reduction.

VI. REFERENCES

[1]Nawrocki, P. and Reszelewski, W., 2017. Resource usage optimization in mobile cloud computing. *Computer Communications*, *99*, pp.1-12.

[2]Adhikari, M., Amgoth, T. and Srirama, S.N., 2019. A survey on scheduling strategies for workflows in

cloud environment and emerging trends. *ACM Computing Surveys (CSUR)*, *52*(4), pp.1-36.
[3]Buyya, R., Srirama, S.N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L.M., Netto, M.A. and Toosi, A.N., 2018. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM computing surveys (CSUR)*, *51*(5), pp.1-38.
[4]Yussupov, V., Breitenbücher, U., Hahn, M. and Leymann, F., 2019, October. Serverless parachutes: Preparing chosen functionalities for exceptional workloads. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 226-235). IEEE.
[5]Gill, S.S., Tuli, S., Xu, M., Singh, I., Singh, K.V., Lindsay, D. and Pervaiz, H., 2019. Internet of things. *Internet of Things*, *8*(100118), pp.1-30.
[6]Tao, Z., Xia, Q., Hao, Z., Li, C., Ma, L., Yi, S. and Li, Q., 2019. A survey of virtual machine management in edge computing. *Proceedings of the IEEE*, *107*(8), pp.1482-1499.
[7]Kratzke, N., 2018. A brief history of cloud application architectures. *Applied Sciences*, *8*(8), p.1368.
[8]Gadepalli, P.K., Peach, G., Cherkasova, L., Aitken, R. and 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.
[9]Chandra Paul, P., Loane, J., McCaffery, F. and Regan, G., A Serverless Architecture for Wireless Body Area Network Application. *IMBSA 2019: Model-Based Safety and Assessment*, *11842*, pp.239-254.
[10]Gunasekaran, J.R., Thinakaran, P., Kandemir, M.T., Urgaonkar, B., Kesidis, G. and Das, C., 2019, July. Spock: Exploiting serverless functions for slo and cost aware resource procurement in public cloud. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (pp. 199-208). IEEE.
[11]Zhang, C., Yu, M., Wang, W. and Yan, F., 2019. {MArk}: Exploiting cloud services for {Cost-Effective},{SLO-Aware} machine learning inference serving. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)* (pp. 1049-1062).
[12]Diallo, S. and Ndlovu, G., 2019. Energy-Efficient ML Inference Pipelines for IoT Data Using Serverless Cloud Functions. *International Journal of Artificial Intelligence & Digital Transformation*, *2*(2), pp.01-19. [13]Agarwal, P. and Lakshmi, J., 2019, September. Cost aware resource sizing and scaling of microservices. In *Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things* (pp. 66-74).
[14]Alvarez, F., Breitgand, D., Griffin, D., Andriani, P., Rizou, S., Zioulis, N., Moscatelli, F., Serrano, J., Keltsch, M., Trakadas, P. and Phan, T.K., 2019. An edge-to-cloud virtualized multimedia service platform for 5G networks. *IEEE Transactions on Broadcasting*, *65*(2), pp.369-380.
[15]Rosati, P., Fowley, F., Pahl, C., Taibi, D. and Lynn, T., 2018, March. Right scaling for right pricing: A case study on total cost of ownership measurement for cloud migration. In *International Conference on Cloud Computing and Services Science* (pp. 190-214). Cham: Springer International Publishing.
[16]Mohanty, S.K., Premsankar, G. and Di Francesco, M., 2018, December. An evaluation of open source serverless computing frameworks. In *IEEE International Conference on Cloud Computing Technology and Science* (pp. 115-120). IEEE.
[17]Palade, A., Kazmi, A. and Clarke, S., 2019, July. An evaluation of open source serverless computing frameworks support at the edge. In *2019 IEEE world congress on services (SERVICES)* (Vol. 2642, pp. 206-211). IEEE.
[18]McGrath, G. and Brenner, P.R., 2017, June. Serverless computing: Design, implementation, and performance. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 405-410). IEEE.
[19]Li, J., Kulkarni, S.G., Ramakrishnan, K.K. and Li, D., 2019, December. Understanding open source serverless platforms: Design considerations and performance. In *Proceedings of the 5th international workshop on serverless computing* (pp. 37-42).
[20]Jackson, D. and Clynch, G., 2018, December. An investigation of the impact of language runtime on the performance and cost of serverless functions. In *2018 IEEE/ACM international conference on utility and cloud computing companion (UCC companion)* (pp. 154-160). IEEE.
[21]Wang, H., Niu, D. and Li, B., 2019, April. Distributed machine learning with a serverless architecture. In *IEEE INFOCOM 2019-IEEE conference on computer communications* (pp. 1288-1296). IEEE.
[22]Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L. and Pallickara, S., 2018, April. Serverless computing: An investigation of factors influencing microservice performance. In *2018 IEEE international conference on cloud engineering (IC2E)* (pp. 159-169). IEEE.
[23]Tak, B., Park, S. and Kudva, P., 2019. Priolog: Mining important logs via temporal analysis and prioritization. *Sustainability*, *11*(22), p.6306.
[24]Bravetti, M., Giallorenzo, S., Mauro, J., Talevi, I. and Zavattaro, G., 2019, April. Optimal and automated deployment for microservices. In *International*

Conference on Fundamental Approaches to Software Engineering (pp. 351-368). Cham: Springer International Publishing.

[25]Wilhelm, S., Förster, R. and Zimmermann, A.B., 2019. Implementing competence orientation: Towards constructively aligned education for sustainable development in university-level teaching-and-learning. *Sustainability*, *11*(7), p.1891.

[26]Grody, A.D., 2018. Rebuilding financial industry infrastructure. *Journal of Risk Management in Financial Institutions*, *11*(1), pp.34-46.

[27]Tcvetkov, P., Cherepovitsyn, A. and Fedoseev, S., 2019. The changing role of CO2 in the transition to a circular economy: Review of carbon sequestration projects. *Sustainability*, *11*(20), p.5834.

[28]Armstrong, R.C., Wolfram, C., De Jong, K.P., Gross, R., Lewis, N.S., Boardman, B., Ragauskas, A.J., Ehrhardt-Martinez, K., Crabtree, G. and Ramana, M.V., 2016. The frontiers of energy. *Nature Energy*, *1*(1), p.15020.

[29]García-López, P., Sánchez-Artigas, M., Shillaker, S., Pietzuch, P., Breitgand, D., Vernik, G., Sutra, P., Tarrant, T. and Ferrer, A.J., 2019. Servermix: Tradeoffs and challenges of serverless data analytics. *arXiv preprint arXiv:1907.11465*.

[30]Pu, Q., Venkataraman, S. and Stoica, I., 2019. Shuffling, fast and slow: Scalable analytics on serverless infrastructure. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)* (pp. 193-206).

[31]Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N. and Gonzalez, J.E., 2019. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*.

[32]Kannan, R.S., Subramanian, L., Raju, A., Ahn, J., Mars, J. and Tang, L., 2019, March. Grandslam: Guaranteeing slas for jobs in microservices execution frameworks. In *Proceedings of the Fourteenth EuroSys Conference 2019* (pp. 1-16).

[33]Hermenier, F., Ramesh, A., Nagpal, A., Shukla, H. and Chandra, R., 2019, November. Hotspot mitigations for the masses. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 102-113).

[34] GitHub., (2020). *GitHub - Azure/AzurePublicDataset: Microsoft Azure Traces*. Available at: https://github.com/Azure/AzurePublicDataset

[35]Villalba, A., Berral, J.L. and Carrera, D., 2018. Constant-time sliding window framework with reduced memory footprint and efficient bulk evictions. *IEEE Transactions on Parallel and Distributed Systems*, *30*(3), pp.486-500.