Cosmos: A Cost Model for Serverless Workflows in the 3D Compute Continuum

Cynthia Marcelino
Distributed Systems Group
TU Wien, Vienna, Austria
c.marcelino@dsg.tuwien.ac.at

Sebastian Gollhofer-Berger
Distributed Systems Group
TU Wien, Vienna, Austria
e12024002@student.tuwien.ac.at

Thomas Pusztai
Distributed Systems Group
TU Wien, Vienna, Austria
t.pusztai@dsg.tuwien.ac.at

Stefan Nastic
Distributed Systems Group
TU Wien, Vienna, Austria
snastic@dsg.tuwien.ac.at

2025 IEEE International Conference on Smart Computing (SMARTCOMP) | 979-8-3315-8646-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/SMARTCOMP65954.2025.00072

***Abstract*—Due to the high scalability, infrastructure management, and pay-per-use pricing model, serverless computing has been adopted in a wide range of applications such as real-time data processing, IoT, and AI-related workflows. However, deploying serverless functions across dynamic and heterogeneous environments such as the 3D (Edge-Cloud-Space) Continuum introduces additional complexity. Each layer of the 3D Continuum shows different performance capabilities and costs according to workload characteristics. Cloud services alone often show significant differences in performance and pricing for similar functions, further complicating cost management. Additionally, serverless workflows consist of functions with diverse characteristics, requiring a granular understanding of performance and cost trade-offs across different infrastructure layers to be able to address them individually. In this paper, we present Cosmos, a cost- and a performance-cost-tradeoff model for serverless workflows that identifies key factors that affect cost changes across different workloads and cloud providers. We present a case study analyzing the main drivers that influence the costs of serverless workflows. We demonstrate how to classify the costs of serverless workflows in leading cloud providers AWS and GCP. Our results show that for data-intensive functions, data transfer and state management costs contribute to up to 75% of the costs in AWS and 52% in GCP. For compute-intensive functions such as AI inference, the cost results show that BaaS services are the largest cost driver, reaching up to 83% in AWS and 97% in GCP.**

***Index Terms*—serverless, cost, edge, space, cloud, continuum**

I. Introduction

Serverless computing offers high scalability and automatic infrastructure management with fine-grained resource utilization in a pay-per-use business model [1, 2, 3]. Due to its advantages, serverless computing has been widely adopted in different applications such as real-time data processing, IoT, and AI inference [4, 5, 6]. Small pieces of code are wrapped in short-lived functions managed by the platform. Typically, serverless functions are event-driven and stateless, which means they leverage external services, called Backend-as-a-Service (BaaS), to manage state and additional features, such as request routing, AI inference, and user authentication. Although functions are billed only for the execution time, the dependence on BaaS services leads to additional costs [7, 8, 9, 10].

Recently, the deployment of thousands of Low Earth Orbit (LEO) satellites with inter-satellite links (ISL) allows extending serverless computing beyond the edge and cloud into space, forming a 3D Compute Continuum. This continuum allows for dynamic and efficient execution of serverless workflows that leverage the advantages of each of its three layers: Edge computing for low-latency processing near data sources, the cloud for scalable, high-capacity computing, and LEO satellites for in-orbit processing and low-latency communication to reduce reliance on Earth-based data transfer [11, 12].

However, reliably predicting the costs of serverless workflows remains challenging. Cloud providers impose complex pricing models for different services, while edge and space layers of the 3D Continuum add further complexities, such as limited resources and energy constraints [13, 14]. Furthermore, functions in a workflow may vary from compute-intensive to data-intensive tasks, each with distinct resource and performance demands [13, 15]. Therefore, it is essential to model the costs of each function in a workflow to allow for finding an appropriate performance-cost tradeoff.

Common approaches for serverless cost estimation include: (a) *Predictions* [16, 17, 18] use models, such as ML and math models to estimate costs based on historical execution data. This enables the estimation and analysis of costs without executing or even deploying a workflow. However, these high-level predictions often fail to provide detailed cost breakdowns or to identify the main drivers of higher expenses. (b) *Simulations* [19, 20, 21] enable users to explore how costs behave under different parameter configurations. They offer valuable insights into performance and expenses across various workload patterns, highlighting important trade-offs. However, existing simulation tools often lack fine-grained parameters to identify which aspects contribute to higher costs.

Since current cost models are not detailed enough for precise performance-cost tradeoff decisions, users often err on the side of caution and incur higher costs to ensure performance. To address this gap, we present a classification of serverless costs, focusing on isolating and identifying the main cost drivers of workflows. The Cosmos cost model enables the building of intelligent frameworks to optimize serverless costs and maximize performance. Our main contributions include:

- *Cosmos: A cost and a performance-cost tradeoff model for serverless workflows* that incorporates the heterogeneity and dynamic characteristics of the 3D Continuum. Cosmos isolates the main cost drivers while accounting for their interdependencies, providing an understanding of how different factors impact execution and cost, e.g., resource

2693-8340/25/$31.00 ©2025 IEEE
DOI 10.1109/SMARTCOMP65954.2025.00072

constraints, workload characteristics, communication overhead, and dynamic pricing.

- *A cost taxonomy that classifies the main cost drivers*, enabling their identification among invocation, compute, data transfer, state management, and BaaS. This provides insights into specific cost drivers for serverless workflows across the different layers of the 3D Continuum.
- *A case study on different commercial cloud and edge providers*, including AWS x86, AWS ARM, AWS Lambda@Edge, and GCP. We analyze the primary cost drivers associated with each platform. Since executing experiments in space is currently impractical, we use cloud and edge experiments to systematically evaluate each cost driver and extrapolate the insights to the 3D Continuum.

Our experiments show that data transfer and state management costs account for 75% of AWS costs and 52% of GCP costs for IO-intensive workloads. On the other hand, BaaS costs are the largest in compute-intensive functions, reaching up to 83% on AWS and 97% on GCP. Our performance-cost model highlights the options with the best tradeoff.

II. Illustrative Scenario & Research Questions

A. Illustrative Scenario

Fig. 1 shows an illustrative scenario, where the 3D Continuum enables a scalable serverless workflow for deforestation detection in remote areas, inspired by the DETER program in Brazil [22]. Drones collect environmental data, such as temperature, CO_2 levels, and high-resolution images. They transmit data to LEO satellites, which combine the edge-collected data with Earth observation (EO) imagery in a preprocessing step directly in orbit. Data volume reduction in space is more efficient than downlinking raw EO satellite data (1,5 TB per day [23, 24]) to Earth over a radio connection with an average speed of 300 Mbps [25]. ISLs do not suffer from interference from the Earth's atmosphere and can offer bandwidths up to 100 Gbps [26, 27]. Hence, the transfer time from EO satellites is reduced. The preprocessed data is downlinked to the cloud for performing deforestation pattern detection with compute-intensive ML models. By properly distributing tasks across the 3D Continuum, data can be preprocessed closer to the source, such that compute-intensive AI inference tasks in the cloud receive their inputs faster, resulting in an overall reduction in end-to-end execution time.

Optimizing data processing across all layers of the 3D Continuum is crucial to maximizing performance without depleting resource-constrained devices such as edge devices and LEO satellites. This requires identifying the performance and cost factors within each layer to address specific limitations and enable the 3D Continuum to process the serverless workflow effectively. By understanding these factors, frameworks can dynamically allocate workloads in the most suitable locations to minimize costs while maximizing performance on resource-limited devices and reducing overall execution time.

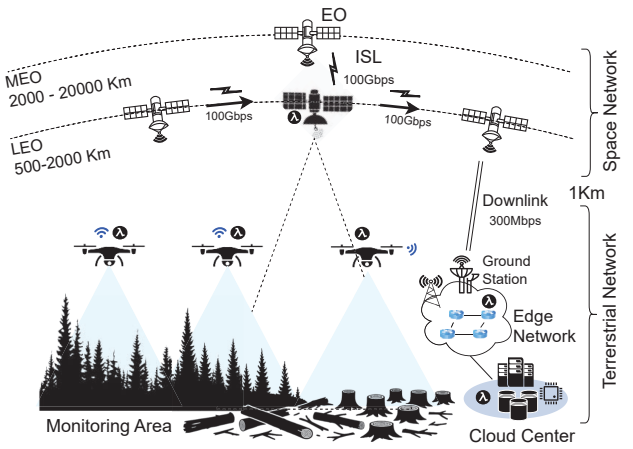


Fig. 1: Deforestation detection scenario with on-ground and in-orbit processing with serverless for the 3D Continuum.

B. Research Questions

We identify the following research questions in optimizing and classifying the costs across the 3D Continuum.

RQ-1: How can serverless workflows in the 3D Continuum be optimized according to their workload characteristics?

Optimizing serverless workflows across the 3D Continuum is challenging due to the varied characteristics of each infrastructure layer. Edge devices have limited computational power but provide low latency. Cloud services offer high scalability, but have higher latency and dynamic pricing, which can impact cost-sensitive workflows. LEO satellites provide low-latency network connections around the globe, but they have limited computational resources, and their power supply and onboard heat generation depend on their current position relative to the sun [11, 12, 28]. To optimize serverless workflows, we need to identify the workload requirements and associated costs for functions, especially when workloads have different needs for computing, resource usage, data transfer, and BaaS services.

RQ-2: How can cost models accurately capture and predict serverless execution costs for heterogeneous environments?

Accurate prediction of serverless costs across diverse infrastructure layers requires accounting for the different characteristics, such as execution time, pricing, and operational constraints. It is essential to identify and integrate cost drivers from the dynamic characteristics of workload and infrastructure into a unified cost model. State-of-the-art serverless cost models [20, 29] do not provide fine-grained cost drivers such as fixed and dynamic prices to model the total cost of serverless functions. Failing to identify and integrate the cost drivers of dynamic environments, such as the 3D Continuum, can lead to inaccurate cost estimates, resulting in inefficient resource allocation, reduced performance, and increased expenses.

RQ-3: How to evaluate and benchmark cost drivers for serverless functions in the heterogeneous 3D Continuum? Validating cost models and workloads for serverless functions requires benchmarking across different infrastructures [18, 30].

However, replicating serverless workflows across edge, cloud, and space is challenging, mainly due to cross-layer interactions such as edge-to-cloud or cloud-to-space data transfers, complicating performance and cost evaluations. Therefore, isolating the impact of specific cost drivers in such heterogeneous environments is complex and expensive.

III. Serverless Main Cost Drivers in the 3D Continuum and Cosmos Cost and Performance-Cost Tradeoff Model

The serverless computing pricing model allows serverless functions to be billed only for execution time, avoiding costs with idle computing resources. Typically, a serverless workflow (Fig. 2) is composed of multiple functions distributed across the 3D Continuum that generate costs across several distinct factors. Each cost driver represents a specific characteristic of the task executed and services consumed during serverless workflow execution.

A. Serverless Main Cost Drivers in the 3D Continuum

Fig. 3 presents a taxonomy of the main cost drivers associated with serverless workflows, highlighting the focus of this analysis: Invocation, Compute, Data Transfer, and State Management. The main cost drivers are directly tied to the execution and performance of serverless functions, representing the most variable and impactful cost components in typical serverless workflows. Unlike some fixed costs, such as subscriptions and provisioned resources, which remain constant regardless of usage, the underlined drivers exhibit cost fluctuations based on function activity, data flows, and resource consumption.

a) Invocation: Invocation is the cost incurred each time a serverless function is triggered. This cost is calculated on a per-request basis and remains constant, regardless of the size of the request payload or the execution time of the function. The number of invocations or requests can influence the serverless platform's decision on scaling the function up or down.

b) Compute: Compute costs are determined by the execution time of the function and the allocated computational resources. It includes pricing based on the execution time in seconds and the memory allocated to the function, often represented in GB-seconds. The cost is directly proportional to the intensity of computation and the duration for which resources are consumed during each invocation.

c) State Management: Serverless functions are by design stateless, which means they leverage external services to store and manage state. As shown in Fig. 3, State Management is also a BaaS. However, as detailed in Section V, state management constitutes a significant portion of the overall function cost. It involves the persistence and handling of required data for executing serverless functions. These costs arise from storage retention, which may be fixed (e.g., monthly storage) or dynamic (e.g., per-operation costs). Typically, serverless functions leverage many state management services, such as object storage, key-value store (KVS), message brokers, and databases.

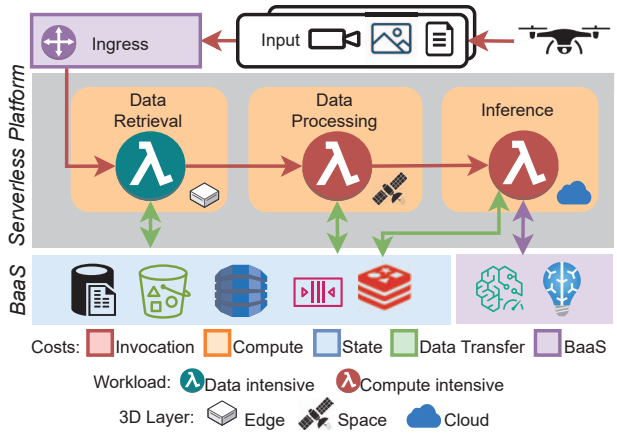


Fig. 2: Simplified serverless workflow for deforestation with main cost drivers along and workload characteristics of each function.

d) Data Transfer: Data transfer refers to the data movement between serverless functions and external systems. This includes both inbound and outbound data traffic, which may involve transferring data among serverless functions, external databases, and clients. Typically, the pricing model for data transfer is based on the volume of data transferred, measured in gigabytes (GB), and the network path used, such as intra-region and inter-region transfers.

e) BaaS: BaaS costs refer to the charges associated with additional services that support serverless functions and workflows. These services include managed APIs, event gateways, data processing frameworks (such as Glue DPU), and AI platforms such as AWS SageMaker and Vertex AI. The costs for these additional services can be divided into two categories: fixed costs, associated with hourly or monthly fees for service availability, and dynamic costs, which change based on the number of requests, the amount of data processed, or specific operations performed.

B. Cosmos Cost Model

The total cost of a serverless workflow is represented as the sum of each cost driver: invocation, computation, state management, data transfer, and BaaS [10, 30, 31]. Cosmos proposes to isolate the main cost drivers of serverless workflows to better understand their impact, even though these costs are interconnected. For example, while data transfer costs can affect execution time for large inputs, our model distinguishes between these factors to determine whether variations in compute time are due to the complexity of the workload or the overhead related to data movement.

a) Function Invocation Cost: The invocation cost C^{inv} for function i accounts for the fixed price incurred for each request handled by the function, where n_i is the number of requests handled by a function i, and $p_{\text{inv},i}$, the price per invocation for the function i. Thus, the invocation cost can be expressed as:

$$C_i^{\text{inv}} = n_i \cdot p_{\text{inv},i} \tag{1}$$

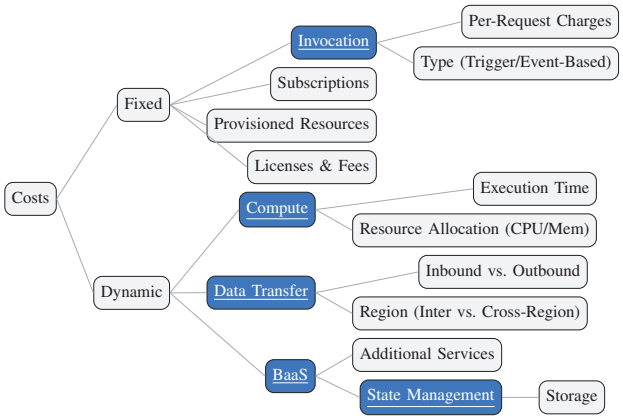


Fig. 3: Serverless workflow costs drivers, highlighting key cost drivers: Invocation, Compute, Data Transfer, and State Management (partial view).

b) Compute Cost: The compute cost C^{exec} for function i depends on the execution time and computational resources consumed, where n_i is the number of requests received by function i, t_i, the compute time per request, and $p_{\text{exec},i}$, the price per GB-second for execution. The data transfer for each request is calculated in Eq. (4). Thus, the compute cost for a specific compute duration can be expressed as:

$$C_i^{\text{exec}} = n_i \cdot t_i \cdot p_{\text{exec},i} \tag{2}$$

c) State Management Costs: The state management cost C^{state} for function i relates to additional services to store the function state such as storage systems. Typically, state management for each function i includes d_i, the amount of data stored (typically in GB), $p_{\text{state\_fixed},i}$, the price per GB of fixed storage costs in a certain amount of time (e.g., monthly charges). Therefore, the storage cost can be expressed as:

$$C_i^{\text{state}} = d_i \cdot p_{\text{state\_fixed},i} \tag{3}$$

d) Data Transfer Cost: The data transfer cost C^{transfer} for a function accounts for the cost of transferring data both into and out of the function. It depends on the number of requests n_i, handled by function i, $r_{\text{in},i}$ and $r_{\text{out},i}$, the total input and output data size transferred per request, respectively, and $p_{\text{t\_in},i}$ and $p_{\text{t\_out},i}$, the respective prices per GB for input and output data transfer. The total data transfer cost can be expressed as:

$$C_i^{\text{transfer}} = n_i \cdot (r_{\text{in},i} \cdot p_{\text{t\_in},i} + r_{\text{out},i} \cdot p_{\text{t\_out},i}) \tag{4}$$

e) BaaS Costs: The BaaS cost C^{baas} for each function i includes $t_{\text{fixed},i}$, the duration of fixed-cost services; $p_{\text{fixed},i}$, the price per unit time for fixed costs; n_i, the number of requests handled; r_i, the data processed per request (in GB); and $p_{\text{dynamic},i}$, the price per unit for dynamically priced services. Therefore, the BaaS cost can be expressed as:

$$C_i^{\text{baas}} = t_{\text{fixed},i} \cdot p_{\text{fixed},i} + n_i \cdot r_i \cdot p_{\text{dynamic},i} \tag{5}$$

f) Total Compute Layer-Specific Cost: Different layers might introduce different pricing models. For instance, LEO-based processing introduces unique computation costs for satellites, due to their high launch costs. Therefore, the total cost $C_{i,L}$ of a workflow aggregates the invocation, compute, state, data transfer, and BaaS costs for all functions in the workflow across every layer of the 3D Continuum. It is expressed as the sum of individual costs for all functions i in F, executed in layer $L \in \{e, c, s\}$, where e edge, c cloud, and s space:

$$C_{i,L} = C_{i,L}^{\text{inv}} + C_{i,L}^{\text{exec}} + C_{i,L}^{\text{state}} + C_{i,L}^{\text{transfer}} + C_{i,L}^{\text{baas}} \tag{6}$$

C. Cosmos Performance-Cost Trade-off Model

Optimizing serverless workflow execution in the 3D Compute Continuum requires balancing two competing objectives: minimizing costs and minimizing execution time. Processing functions closer to the data source (e.g., edge or space) reduces execution time but incurs higher costs, while cloud resources are cost-effective but introduce latency. Therefore, we define an optimization model that dynamically determines function execution while respecting a given budget and latency SLO constraints.

We define the total execution time of function i on layer L as $T_{i,L}$ Our goal is to minimize both total cost and total execution time, where α and β are weighting factors that dynamically adjust the relative importance of cost and execution time. Our performance-cost model can be defined as:

$$\begin{aligned} \min \quad & \sum_{i \in F} \sum_{L} \left(\alpha C_{i,L} + \beta T_{i,L} \right) \\ \text{s.t.} \quad & \sum_{i \in F} \sum_{L} C_{i,L} \leq B, \quad \text{(Budget constraint)} \\ & \sum_{i \in F} \sum_{L} T_{i,L} \leq L_{\max}, \text{(Latency SLO constraint)} \end{aligned} \tag{7}$$

Instead of manually selecting α and β, we employ a Pareto front approach [32, 33] to dynamically balance cost and execution time. We solve two separate optimization problems to determine the best-case scenarios for cost and execution time. We first minimize cost without considering execution time. Let the cost minimization from this be C^*:

$$C^* = \min \sum_{i \in F} \sum_{L} C_{i,L} \quad \Rightarrow \quad \alpha = \frac{1}{C^*} \tag{8}$$

Next, we minimize execution time and let this minimization be T^*:

$$T^* = \min \sum_{i \in F} \sum_{L} T_{i,L} \quad \Rightarrow \quad \beta = \frac{1}{T^*} \tag{9}$$

Cosmos performance-cost model trade-off ensures an optimal balance in which cost and performance are equally prioritized by dynamically adjusting weighting α and β based on the best achievable budget and latency SLOs, making it well-suited for optimizing serverless workflows in 3D Compute Continuum.

IV. Case Study Implementation

We implemented a simplified serverless workflow as described in Section II-A. We identified the cost drivers using AWS (x86, ARM, and Lambda@Edge) and GCP, which offer detailed billing metrics and in-depth insights. Our case study has three serverless functions: data retrieval, data processing, and AI inference. Each function is implemented in Python with the required BaaS services to emulate real-world serverless workflows and measure their cost drivers. Our case study implementation is published as an open-source framework part of the Polaris SLO Cloud project and available on Github[1].

a) Data Retrieval: We use AWS Lambda for compute, API Gateway for HTTP requests, and retrieve data from DynamoDB or S3. In GCP, we use Cloud Functions to interact with Firestore for queries and Cloud Storage for retrieval, responding directly to HTTP requests.

b) Data Processing: In AWS, we utilize Lambda for computing, API Gateway for request handling, and AWS Glue to execute ETL tasks, with data written to S3 or DynamoDB. GCP implements Cloud Functions for orchestration and Dataflow for ETL processing, leveraging its pay-as-you-go model for CPU, memory, and data transfer.

c) AI Inference: We use AWS Lambda for preprocessing, API Gateway for request handling, and SageMaker Serverless for inference, while in GCP, Cloud Functions route requests to Vertex AI for model execution. Invocation, execution, storage, and data transfer costs are logged via AWS CloudWatch and GCP Cloud Monitoring to validate our cost model.

V. Evaluation

To validate our cost model, we implement and evaluate a serverless workflow containing typical tasks in image processing as described in our illustrative scenario in Fig. 1. We executed the implemented workflow in two major leading cloud serverless platforms, AWS Lambda [34] and GCP Google Cloud Functions (GCF) [35].

a) Metrics: Performance-Cost Trade-off shows the trade-offs between lower latency and lower costs. *Latency* shows the mean execution time for each function from the HTTP client. *Cost* evaluates the financial costs of the functions by analyzing function invocation, execution costs, and data transfers, as well as the costs associated with BaaS services. The costs are calculated in USD per million requests.

b) Baselines: In our experiments, we validate our cost model and compare the designed metrics for our serverless functions between two leading cloud providers, AWS and GCP.

A. Experimental Setup

We executed the workflow presented in Fig. 2 using services from AWS and GCP. For each function, we selected similar services, such as Cloud Storage and AWS S3. We deployed AWS functions with 128MB RAM in the `eu-central-1 (Frankfurt)` region, and GCP functions

[1]https://github.com/polaris-slo-cloud/cosmos

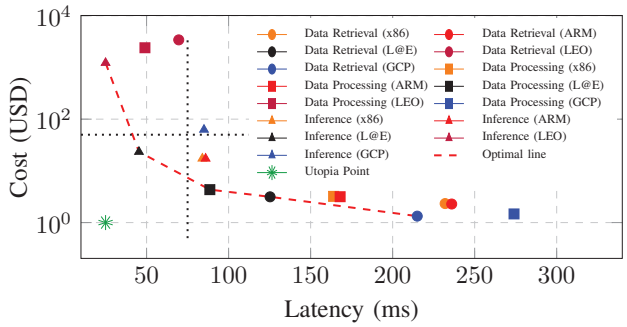


Fig. 4: Cosmos Performance Cost Tradeoff Model highlighting the optimal line between latency and costs, with theoretical lowest cost and latency, and SLO constraints of 50 USD and 75ms (pointed line).

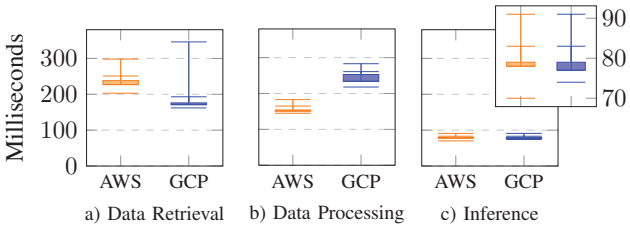
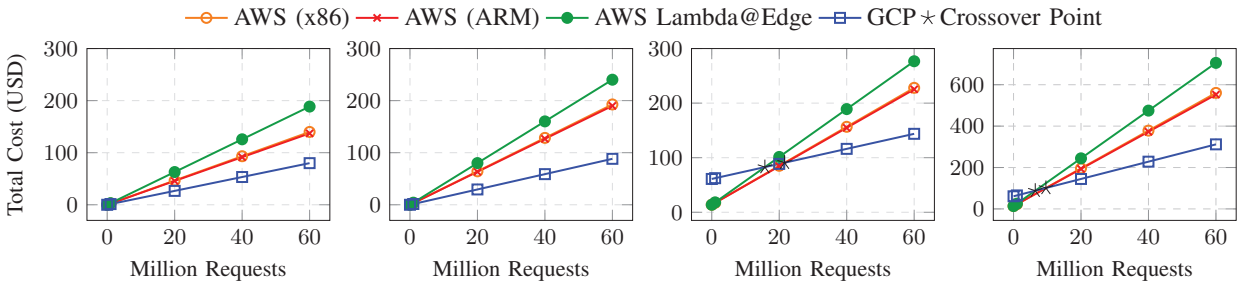


Fig. 5: End-to-end latency for serverless workflows for AWS(x86) and GCP

in the `europe-west3-a (Frankfurt)` region. We executed HTTP requests using the Postman REST client on a MacBook Pro 2020 with an i7 processor. To ensure consistency, for each experiment presented, we performed the tests five times sequentially and at similar times for both providers and calculated the mean results to analyze performance and costs under varying workload scenarios.

B. Performance-Cost Trade-off Results

Fig. 4 depicts the trade-off between latency and cost for different layers and platforms in the 3D continuum, including AWS (x86, ARM, Lambda@Edge), GCP, and a hypothetical LEO. To the best of our knowledge, there is no existing pay-per-use pricing model for LEO computing yet; therefore, we assume that LEO execution price of 49USD for 1M requests per ms executed based on [36], while achieving lower latency than L@E [26, 27], and L@E offers nearly 2x lower latency over AWS Lambda [37]. Further, to better understand the trade-offs and capture layer-specific impact, we assume that functions run entirely on a single layer, each offering a similar resource capacity. In Fig. 4, the dashed red line represents the optimal cost-latency trade-off. The points below are infeasible, as no deployment can achieve both lower cost and latency, illustrated by the utopia point, which represents the lowest cost and latency but is unreachable. The points above the optimal line are feasible but always involve trade-offs of either higher cost or latency. Inference (x86, ARM, and GCP) functions do not appear on the optimal line due to their high costs, which are driven by BaaS costs, making them less cost-effective compared to Data Retrieval and Data Processing. Although



(a) Data Retrieval (b) Data Processing (c) AI Inference (d) Total Workflow Cost

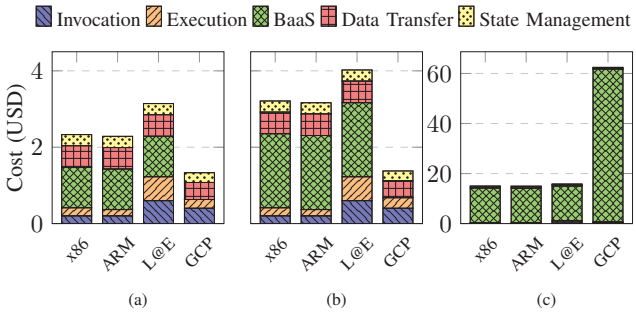
Fig. 6: Cost comparison of serverless workflows across AWS(x86, ARM, and Lambda@Edge) and GCP for different workloads, showing the total costs as a function of request volume for each function and the total workflow.

TABLE I: Detailed cost comparison across serverless functions for AWS (x86, ARM, Lambda@Edge (L@E)) and GCP in USD for 1M requests.

Workflow	Component	AWS (x86)	AWS (ARM)	L@E	GCP	Cost Unit
Data Retrieval	Function Invocation	0.20	0.20	0.60	0.40	1M requests
	Function Execution	0.213	0.1667	0.6251	0.2304	GB-second
	API Gateway	1.06	1.06	1.06	-	1M requests
	DynamoDB Reads	0.1345	0.1345	0.1345	-	1M requests
	DynamoDB Storage	0.269	0.269	0.269	-	GB-month
	Firestore Reads	-	-	-	0.046	1M requests
	Firestore Storage	-	-	-	0.231	GB-month
	Total DynamoDB/Firestore	0.4035	0.4035	0.4035	0.277	Read/Store
	AWS S3 Retrieval	0.43	0.43	0.43	-	1M requests
	AWS S3 Storage	0.0245	0.0245	0.0245	-	GB-month
	Cloud Storage Retrieval	-	-	-	0.4	1M requests
	Cloud Storage	-	-	-	0.025	GB-month
	Total AWS S3/Cloud Storage	0.4545	0.4545	0.4545	0.425	Read/Store
	Total	**2.331**	**2.2847**	**3.1431**	**1.3324**	-
Data Processing	Function Invocation	0.20	0.20	0.60	0.40	1M requests
	Function Execution	0.213	0.1667	0.6251	0.276	GB-second
	API Gateway	1.06	1.06	1.06	-	1M requests
	DynamoDB Reads	0.1345	0.1345	0.1345	-	1M requests
	DynamoDB Storage	0.269	0.269	0.269	-	GB-month
	Firestore Reads	-	-	-	0.046	1M requests
	Firestore Storage	-	-	-	0.231	GB-month
	Total DynamoDB/Firestore	0.4035	0.4035	0.4035	0.277	Read/Store
	AWS S3 Retrieval	0.43	0.43	0.43	-	1M requests
	AWS S3 Storage	0.0245	0.0245	0.0245	-	GB-month
	Cloud Storage Retrieval	-	-	-	0.4	1M requests
	Cloud Storage	-	-	-	0.025	GB-month
	Total AWS S3/Cloud Storage	0.4545	0.4545	0.4545	0.425	Read/Store
	Glue DPU (2-hour ETL)	0.88	0.88	0.88	-	Processing
	Dataflow CPU	-	-	-	0.07325	CPU-hour
	Dataflow Memory	-	-	-	0.00465	GB-hour
	Dataflow Processed Data	-	-	-	0.01439	GB processed
	Total ETL Costs	0.88	0.88	0.88	0.09229	DPU/Dataflow
	Total	**3.211**	**3.1647**	**4.0031**	**1.47029**	-
AI Inference	Function Invocation	0.20	0.20	0.60	0.40	1M requests
	Function Execution	0.213	0.1667	0.6251	0.2304	GB-second
	API Gateway	1.06	1.06	1.06	-	1M requests
	DynamoDB Reads	0.1345	0.1345	0.1345	-	1M requests
	DynamoDB Storage	0.269	0.269	0.269	-	GB-month
	Firestore Reads	-	-	-	0.046	1M requests
	Firestore Storage	-	-	-	0.231	GB-month
	Total DynamoDB/Firestore	0.4035	0.4035	0.4035	0.277	Read/Store
	AWS S3 Retrieval	0.43	0.43	0.43	-	1M requests
	AWS S3 Storage	0.0245	0.0245	0.0245	-	GB-month
	Cloud Storage Retrieval	-	-	-	0.4	1M requests
	Cloud Storage	-	-	-	0.025	GB-month
	Total AWS S3/Cloud Storage	0.4545	0.4545	0.4545	0.425	Read/Store
	SageMaker Provisioning	13.7376	13.7376	13.7376	-	Fixed monthly
	Vertex AI Provisioning	-	-	-	61.056	Fixed monthly
	SageMaker Inference	1.24	1.24	1.24	-	request
	Vertex AI Inference	-	-	-	0.20	request
	Total Fixed + Variable Costs	**17.3086**	**17.2623**	**18.7807**	**62.5884**	-

LEO offers the lowest latency, it significantly increases costs. Inference (GCP), Data Processing (LEO), and Data Retrieval (LEO) significantly exceed the cost and latency SLOs, making it the least among the options evaluated. Latency and cost results are further discussed in Section V-C and Section V-D, respectively.

Performance-Cost Takeaway: The optimal line in Fig. 4 highlights the best trade-offs, with any function above it requiring sacrifices in either cost or latency.



(a) (b) (c)

Fig. 7: Cost breakdowns for AWS (x86, ARM), AWS Lambda@Edge (L@E) and GCP across different functions in a serverless workflow for 1M requests. (a) Data retrieval, (b) Data processing and (c) AI inference.

C. Latency Results

Fig. 5 presents the end-to-end latency results for serverless functions across three workflows: data retrieval (Fig. 5a), data processing (Fig. 5b), and inference (Fig. 5c). The x axis represents the cloud providers (AWS and GCP), and the y axis indicates response latency in milliseconds. For data retrieval, AWS exhibits latencies ranging from 203 ms to 298 ms, while GCP shows a range from 162 ms to 346 ms. In data processing, AWS maintains a latency range of 145 ms to 183 ms, compared to GCP's range of 218 ms to 283 ms. For inference, AWS demonstrates a latency range of 70 ms to 92 ms, while GCP records latencies between 74 ms and 91 ms.

Latency Takeaway: GCP shows 13% lower latency for data-intensive functions, while AWS outperforms GCP by 36% in compute-intensive tasks. For AI inference, both perform similarly, with AWS slightly faster.

D. Cost Results

Fig. 6 compares serverless workflow costs across AWS (x86, ARM, Lambda@Edge) and GCP for data retrieval (Fig. 6a), processing (Fig. 6b), and AI inference (Fig. 6c). GCP consistently offers lower costs, especially at scale, while AWS x86 and ARM have lower fixed costs but higher per-request expenses. While AWS Lambda@Edge is deployed closer to end-users and data sources, potentially reducing latency, in all three

use cases, data retrieval (Fig. 6a), data processing (Fig. 6b), and AI inference (Fig. 6c), AWS Lambda@Edge incurs the highest per-request costs compared to AWS (x86 and ARM), and GCP. For AI inference, GCP's lower incremental costs make it more cost-efficient beyond the crossover point requests; despite its higher fixed cost, it remains more economical for sustained workloads due to its lower incremental cost. Fig. 6d combines the cost of all functions to estimate the total workflow costs. Overall, if one considers the total workflow, AWS is cheaper for low-volume requests, and after crossing the point of around 9 million requests, GCP becomes more cost-efficient.

Fig. 7 shows the cost breakdown for AWS (x86, ARM, Lambda@Edge) and GCP across three workflows: data retrieval (Section V-D), data processing (Section V-D), and AI inference (Section V-D). Costs are categorized into invocation, execution, backend-as-a-service (BaaS), data transfer, and state management. For data retrieval, data transfer dominates, accounting for 53% (AWS x86), 54% (AWS ARM), and 75% (Lambda@Edge), while GCP is at 52%. For data processing, BaaS costs account for 44% of AWS, and data transfer (37% AWS; 12% GCP) drives expenses. AWS Lambda@Edge remains the most expensive due to high execution costs. For AI inference, BaaS dominates, comprising 83% of AWS costs and 97% of GCP.

Cost Provider Takeaway: GCP has 30% lower costs for data-intensive and 57% for compute-intensive workloads than AWS x86. AWS x86 has 75% lower costs for AI inference at low workloads. AWS Lambda@Edge incurs 35% higher data retrieval and 25% higher data processing.

Cost Driver Takeaway: In data retrieval functions, data transfer and state management drive costs, making up 53% (AWS x86), 54% (AWS ARM), and 75% (AWS Lambda@Edge), compared to 52% on GCP. In data processing, BaaS is the primary cost driver, accounting for 60% of AWS x86 and ARM, and 48% of AWS Lambda@Edge, while data transfer contributes 37%, 38%, and 29%, respectively, versus 12% on GCP. In AI inference, BaaS dominates, comprising 83% of AWS costs and 97% of GCP.

E. Threats to Validity

Conducting experiments in space is still challenging due to the high cost, limited accessibility, and lack of publicly available pricing models for in-orbit processing. To the best of our knowledge, there is no standardized cost framework for LEO-based pay-per-use processing. Therefore, our evaluation focuses on edge-cloud-based serverless workflows, which provide a controlled environment that enables us to identify and characterize key cost drivers. For LEO, we make assumptions about pricing models to be able to show the performance trade-off between latency and costs based on Takeme2Space [36]. While our findings provide a foundation for developing a framework that captures key cost-performance trade-offs, we acknowledge that they may vary as pricing models across the 3D Continuum evolve.

VI. Related Work

Costless [29] analyzes the different factors that affect the pricing in serverless functions and presents a framework that predicts the cost and decides about function placement and fusion to minimize the costs. Costless cost model provides granular insights into AWS Lambda functions. However, their case study is tailored for one cloud provider and does not capture the dynamic characteristics of heterogeneous environments such as the 3D Continuum. In [30], authors present a cost model that estimates the total cost of geodistributed training in a multicloud environment, considering the compute cost, storage cost, and data transfer cost. Nevertheless, this cost model only considers storage as a BaaS service, neglecting additional services such as API gateways, costs specific to data transfers, and fixed costs associated with AI-related services. In [16], the authors introduce a tool to predict and optimize the costs of serverless workflows without time-consuming experimentation. The proposed approach leverages Mixture Density Networks to model response time and Monte Carlo simulations to estimate the costs of entire workflows. Although this framework offers high accuracy, it considers the workflow as a whole, making it challenging to identify the most costly parts and how to minimize their impact on costs. COSTA [19] proposes an adaptive cost management framework that constantly monitors and migrates microservices applications across cloud providers to adjust to real-time pricing, reducing costs and avoiding cost SLO violations. However, such frameworks are tailored for VM applications, such as microservices, overlooking serverless-specific costs like BaaS and related services. It also focuses on runtime costs, neglecting fixed costs such as subscriptions or reserved instances, which can be significant for some functions.

Unlike these frameworks that aggregate serverless workflow costs, Cosmos categorizes costs into fixed (e.g., monthly fees) and operational (e.g., invocation, compute, BaaS, storage, and data transfer), allowing precise cost breakdowns and targeted optimizations.

VII. Conclusion

In this paper, we introduce Cosmos, a cost model and performance-cost tradeoff model for serverless workflows, focusing on a detailed classification of main cost drivers such as invocation, compute, data transfer, state management, and BaaS. By analyzing serverless functions with different workload characteristics, such as data and compute intensity, our cost model provides a comprehensive understanding of serverless costs in the 3D Continuum. To validate our proposed cost model, we executed experiments on leading cloud platforms such as AWS and GCP. The results show that data transfer and state management costs significantly account for 75% of the costs of IO-intensive functions. On the other hand, BaaS costs dominate compute-intensive functions, accounting for as much as 97%. While processing data closer to the function provides lower latency, it incurs up to 35% higher costs. These

results highlight the need to align workload characteristics with the dynamic conditions of the 3D Continuum. A fine-grained cost classification is crucial for addressing workload-specific requirements while maximizing performance and minimizing operational costs. In future work, we plan to introduce different optimization mechanisms based on workload characteristics and compare them against state-of-the-art approaches. Additionally, we aim to develop an intelligent framework capable of predicting costs for individual functions in a workflow and dynamically selecting the most suitable layer and provider, accounting for workload-specific characteristics and SLOs, such as latency and cost.

ACKNOWLEDGMENT

This work is partially funded by the Austrian Research Promotion Agency (FFG) under the project RapidREC (Project No. 903884). This work has received funding from the Austrian Internet Stiftung under the NetIdee project LEO Trek (ID 7442). This research received funding from the EU's Horizon Europe Research and Innovation Program through TEADAL (GA No. 101070186) and NexaSphere projects (GA No. 101192912).

REFERENCES

[1] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Serverless applications: Why, when, and how?" *IEEE Software*, vol. 38, no. 1, pp. 32–39, 2021.

[2] S. Eismann, J. Scheuner, E. v. Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "The state of serverless applications: Collection, characterization, and community consensus," *IEEE Transactions on Software Engineering*, vol. 48, no. 10, 2022.

[3] C. Marcelino and S. Nastic, "Cwasi: A webassembly runtime shim for inter-function communication in the serverless edge-cloud continuum," in *Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing*, ser. SEC '23, 2024, p. 158–170.

[4] L. Wang, Y. Jiang, and N. Mi, "Advancing serverless computing for scalable ai model inference: Challenges and opportunities," in *Proceedings of the 10th International Workshop on Serverless Computing*, ser. WoSC10 '24, 2024, p. 1–6.

[5] Z. Hong, J. Lin, S. Guo, S. Luo, W. Chen, R. Wattenhofer, and Y. Yu, "Optimus: Warming serverless ml inference via inter-function model transformation," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24, 2024, p. 1039–1053.

[6] Y. Fu, L. Xue, Y. Huang, A.-O. Brabete, D. Ustiugov, Y. Patel, and L. Mai, "ServerlessLLM: Low-Latency serverless inference for large language models," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024, pp. 135–153.

[7] J. Wen, Z. Chen, X. Jin, and X. Liu, "Rise of the planet of serverless computing: A systematic review," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, Jul. 2023.

[8] S. Nastic, "Self-provisioning infrastructures for the next generation serverless computing," *SN Computer Science*, vol. 5, no. 6, pp. 678 – 693, 2024.

[9] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Communications of the ACM*, 2019.

[10] C. Marcelino and S. Nastic, "Truffle: Efficient data passing for data-intensive serverless workflows in the edge-cloud continuum," in *Proceedings of the IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC '24)*, ser. UCC '24, 2024.

[11] T. Pusztai, C. Marcelino, and S. Nastic, "Hyperdrive: Scheduling serverless functions in the edge-cloud-space 3d continuum," in *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, 2024, pp. 265–278.

[12] D. Lei and A. Saeed, "Do we need a million satellites in orbit? constellation-as-a-service with modular satellites: Challenges and opportunities," in *Proceedings of the 2nd International Workshop on LEO Networking and Communication*, ser. LEO-NET '24, 2024, p. 61–66.

[13] I. E. Akkus, R. Chen, I. Rimac, M. Stein, K. Satzke, A. Beck, P. Aditya, and V. Hilt, "SAND: Towards High-Performance serverless computing," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, Jul. 2018, pp. 923–935.

[14] S. Amershi, A. Begel *et al.*, "Software engineering for machine learning: a case study," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '19, 2019.

[15] C. Marcelino, J. Shahhoud, and S. Nastic, "Goldfish: Serverless actors with short-term memory state for the edge-cloud continuum," in *Proceedings of the 14th International Conference on the Internet of Things*, ser. IoT '24. New York, NY, USA: ACM, 2024.

[16] S. Eismann, J. Grohmann, E. van Eyk, N. Herbst, and S. Kounev, "Predicting the costs of serverless workflows," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '20, 2020, p. 265–276.

[17] R. Cordingly, W. Shu, and W. J. Lloyd, "Predicting performance and cost of serverless computing functions with saaf," in *2020 IEEE International Conference on Cloud and Big Data Computing*, 2020, pp. 640–649.

[18] C. Lin and H. Khazaei, "Modeling and optimization of performance and cost of serverless applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 615–632, 2020.

[19] L. C. da Silva, R. De Medeiros, and N. Rosa, "Costa: A cost-driven solution for migrating applications in multi-cloud environments," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '23, 2023, p. 57–63.

[20] F. Liu and Y. Niu, "Demystifying the cost of serverless computing: Towards a win-win deal," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 1, pp. 59–72, 2024.

[21] P. Garcia Lopez, A. Slominski, B. Metzler, M. Berhendt, and S. Shillaker, "Serverless end game: Disaggregation enabling transparency," in *Proceedings of the 2nd Workshop on SErverless Systems, Applications and MEthodologies*, ser. SESAME '24, 2024, p. 9–14.

[22] J. Doblas, M. S. Reis *et al.*, "Deter-r: An operational near-real time tropical forest disturbance warning system based on sentinel-1 time series analysis," *Remote Sensing*, vol. 14, no. 15, 2022.

[23] "Sentinel-2 operations," https://www.esa.int/enabling_support/operations/sentinel-2_operations.

[24] "Airbus built sentinel-2c satellite successfully launched," https://www.airbus.com/en/newsroom/press-releases/2024-09-airbus-built-sentinel-2c-satellite-successfully-launched.

[25] "European data relay satellite system (edrs) overview," https://connectivity.esa.int/european-data-relay-satellite-system-edrs-overview.

[26] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19, 2019, p. 341–354.

[27] M. Handley, "Delay is not an option: Low latency routing in space," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, ser. HotNets '18, 2018, p. 85–91.

[28] T. Pfandzelter, J. Hasenburg, and D. Bermbach, "Towards a computing platform for the leo edge," in *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, 2021, pp. 43–48.

[29] T. Elgamal, A. Sandur, K. Nahrstedt, and G. Agha, "Costless: Optimizing cost of serverless computing through function fusion and placement," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018.

[30] C. Phalak, D. Chahal, M. Ramesh, and R. Singhal, "Towards geo-distributed training of ml models in a multi-cloud environment," in *Companion of the 15th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '24 Companion, 2024, p. 211–217.

[31] A. Mahgoub, K. Shankar, S. Mitra, A. Klimovic, S. Chaterji, and S. Bagchi, "SONIC: Application-aware data passing for chained serverless applications," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Jul. 2021, pp. 285–301.

[32] M. Majewski, M. Pawlik, and M. Malawski, "Algorithms for scheduling scientific workflows on serverless architecture," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 782–789.

[33] L. A. Bejarano, H. E. Espitia, and C. E. Montenegro, "Clustering analysis for the pareto optimal front in multi-objective optimization," *Computation*, vol. 10, no. 3, 2022.

[34] "AWS Lambda," https://aws.amazon.com/lambda/.

[35] "Google Cloud Functions," https://cloud.google.com/functions.

[36] "Orbitlab: Ai lab in space," https://www.tm2.space/orbitlab.

[37] "Serverless performance: Cloudflare workers, lambda and lambda@edge," https://blog.cloudflare.com/serverless-performance-comparison-workers-lambda/.