Multicloud Deployment of AI Workflows Using FaaS and Storage Services

Manju Ramesh, Dheeraj Chahal, Rekha Singhal
TCS Research
Mumbai, India
{manju.ramesh1, d.chahal, rekha.singhal}@tcs.com

2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 978-1-6654-7706-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/COMSNETS56262.2023.10041365

***Abstract*—Multicloud deployments are getting traction for benefits such as agility, avoiding vendor lock-in, resilience, etc. Use of Function-as-a-Service (FaaS) platforms is emerging as a preferred choice for deploying AI workflows. These platforms from different cloud service providers (CSPs) have unique specifications and cost models. The onus is on a user to find a cost-performance optimal mapping of its application workflows to FaaS and its associated services in a multicloud deployment setting.**

In this work, we have presented an empirical study of multicloud deployment of AI inference-workflows using FaaS and cloud storage services. Our evaluation shows that the cost for executing AI workflow reduces by 83% using an optimal combination of CSPs over a naive deployment. Further, we also propose and evaluate analytical models for estimating the cost of deployment in multicloud using FaaS and storage services.

***Index Terms*—Cloud computing, Function-as-a-Service, Storage Services, Performance and Cost, Multicloud**

I. Introduction

Multicloud deployment of an application involves cloud services from multiple cloud service providers (CSPs), such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud platform (GCP). Multicloud deployment helps in avoiding vendor lock-in, ensure high availability, and promote a better quality of services at a competitive price from the CSPs. Additionally, the use of a free tier quota from all CSPs helps in reducing the deployment cost.

Many popular CSPs offer serverless architecture as Function-as-a-service (FaaS) such as Lambda [1] by AWS, Azure functions [1] by Azure, and Cloud functions [2] by GCP. However, FaaS platforms have different specifications and cost models in different CSPs. Further, serverless functions are stateless and AI inference-workflow will require storage as a communication channel across different components(or tasks/models) of the workflow. It is palpable that the performance and cost of executing a function on different CSPs FaaS platforms are unique. Exploring mappings of application workflows to FaaS and storage services by utilizing their unique specification across different CSPs is a daunting task. However, it is desirable since an optimal mapping might result in the best performance in terms of latency and cost.

AI inference workflows, such as data-driven information extraction applications (e.g. invoice processing), may consist of a directed acyclic graph of deep-learning models interspersed with data preprocessing and post-processing functions. AI inference-workflows have been deployed optimally, for cost and performance, using FaaS and cloud storage services [2]–[4] on a single CSP. Each of the models in the graph may have unique compute and storage resource demands. Such workloads can benefit from heterogeneity in the cost and performance of FaaS services across CSPs [5]. For example, AWS Lambda provides up to six compute cores while Google functions can have a maximum of 2 cores only. Additionally, the cost model of each of these serverless platforms differs significantly. Hence, optimal deployment of AI inference workflow would require intelligent mapping of each of the tasks in the workflows to FaaS of an appropriate cloud. Currently, the onus is on the user to find a cost-performance optimal mapping of its application workflows to FaaS and its associated services in a multicloud deployment setting. The need is to have a methodology or tool to map these heterogeneous tasks to appropriate cloud services of the CSPs which results in optimal performance and cost of deployment.

The cloud storage services such as object storage S3 [6], Blob Storage [7], Cloud storage [8] from AWS, Azure, and GCP may be used as a communication channel for FaaS platform. In multicloud deployment, data movement is imperative across CSPs, so repatriation costs from a CSP and data transfer performance/cost added to the complexity of the problem. Finally, there is no defined cost model for deployments using hybrid services across multiple CSPs.

In this paper, we empirically show the supremacy of multicloud deployment for AI inference workflows, in terms of cost and performance. We propose a methodology for cost and performance optimal deployment using FaaS and storage services across CSPs. The methodology evaluates the resource requirements of the different models in the inference-workflow, tables the FaaS specific features like compute resources from each CSP, storage services' features, billing methodologies, scaling capabilities, data transfer rates, etc. Based on this, we present an optimal architecture for inference-workflows using multicloud FaaS and storage service. We evaluate the methodology for data-driven information extraction workflows spanning across CSPs. Finally, We propose an end-to-end cost model considering the cost of computing, network, storage, and all components in a multicloud scenario.

Succinctly, our contributions are as follows

[1] https://aws.amazon.com/lambda/
[2] https://cloud.google.com/functions

- We empirically study the performance and cost trade-offs for deploying AI inference workflows in a multicloud setting. Specifically, we consider FaaS and storage services in unison to study performance and cost across multiple cloud vendors
- We propose a comprehensive cost model for estimating the cost of AI workflows in a multicloud environment

In this work, we are using FaaS and object storage services from popular cloud vendors AWS, Azure, and GCP. The study is conducted using information extraction (IE) workflows. However, this work can be extended to other cloud vendors and similar workflows.

The rest of the paper is structured as follows. We discuss related work in Section II. A brief introduction to the FaaS platforms and storage services used in this work is given in Section III. The cost model proposed in this work is discussed in Section IV. Methodology for multicloud workload deployments is explained in Section V. The use cases in this study are given in VI. The experimental setup is explained in Section VII followed by an analysis of results in Section VIII. The conclusion is presented in Section IX.

II. Related Work

Mapping workflows to public and multicloud has been studied extensively in the past [9]–[11]. In [12]–[14]. Algorithms are designed to minimize the execution cost while satisfying the defined constraints. In an exhaustive study, pricing strategy from 23 popular cloud vendors is presented in [15].

A fault-tolerant workflow scheduling is presented in [16]. The scheduling of scientific algorithms in multicloud using a hybrid multi-objective optimization algorithm is presented in [17]. Charkavarthi et al. [18] presented a cost-effective Normalization-based Reliable Budget constraint Workflow Scheduling (NRBWS) algorithm to reduce the execution time. Another work presented by Zhu et al. addresses task scheduling on multicloud under security and reliability constraints [19]. Unfortunately, all these works either provide methodology and algorithms for single cloud deployment using Infrastructure as a Service (IaaS) or using one service on a cloud. However, we study multi-cloud deployment and also multiple services within a cloud specifically FaaS platforms and object storage services.

Quenum et al. presented a solution that integrates hierarchical planned and constraint satisfaction solver for FaaS in multicloud [20].A very recent work by Baarzi et al. [21] elaborates the merits and viability of the multicloud serverless platforms. This work envisions a virtual serverless aggregator (VSA) especially for avoiding vendor lock-in, cost optimizations and performance gains. Stoica et al. [22] presented their vision of unified software platform for cloud called *Sky computing* and the role of technical trends and market forces in its emergence. Acceleration of data layer to transfer data between serverless platforms and cloud storage services have been studied in [23].

[24] discusses benefits of multicloud serverless functions and proposes performance metrics monitoring-based approach for dynamically allocating function invocations to a particular cloud provider. However, the functions are standalone without any data sharing and not dependent on each other like in workflows. There are several studies for task scheduling in the cloud using different optimization algorithms based approaches [25] [26], which allocates a certain set of tasks to different VMs, but they are from the perspective of the cloud provider, to maximize the utilization of the resources of the cloud, while giving good latency. Our study is focused on different serverless services across cloud providers, to choose the best mapping in terms of performance and cost from a cloud user perspective.

To the best of our knowledge, this is the first study to understand performance and cost of multicloud deployment of AI workflows using FaaS platforms in conjunction with object storage services of popular cloud vendors.

III. FaaS platforms and Storage services

In this section, we first discuss and compare the FaaS platforms followed by storage services from popular cloud vendors including AWS, Azure and GCP.

A. FaaS Platforms

AWS Lambda The FaaS platform Lambda from AWS can be configured with memory and ephemeral storage up to 10GB. The compute cores allocated with each serverless instance is proportional to the memory. A maximum of 6 cores are allocated with memory configurations of 10GB. However, AWS Lambda has a limit of 250 MB for code including all layers and code. For functions with dependencies that exceeds the size, we use Elastic File System(EFS) to store the required packages.

Google Cloud functions Google Cloud functions can be configured with memory varying from 128MB to 8GB. However, maximum compute cores available with each function is limited to 2 vCPU assigned with 4GB and 8GB memory configurations. Google functions have a limit of 500 MB for storing code and dependencies in each instance. If the requirements exceed the limit of the instance storage, Google Cloud Storage is used for storing the dependencies or files which can be downloaded to the function environment during the runtime.

Azure functions Consumption plan billing is for the actual memory consumed by the functions rather than the memory configured for the function. Azure Premium plan functions provide larger compute function instances with memory up to 14 GB and one instance is active for the duration for which the premium plan is switched on. It is billed for the total configured memory and cores for all instances active for the allocated duration.

B. Storage services

Since FaaS instances are stateless, we need to use some service to store required data. When different components of the same application run in different clouds, there is a need to transfer data between CSPs. We use the object storage services provided by the cloud providers for this - AWS Simple

TABLE I
SYMBOLS USED IN COST MODEL

Symbol	Cost parameter	Cost metric in per $
R_{req_i}	Invocations	Million requests
R_{mem_i}	Memory-Time	GB-second
R_{cpu_i}	CPU-Time	GHz-second
$R_{pw\_mem\_i}$	Pre-warmed Memory-Time	GB-second
$R_{pw\_cpu\_i}$	pre-warmed CPU-Time	GHz-second
R_{st_i}	Data Size in storage	GB-month
$R_{read\_st\_i}$	Reads from storage	No. of Reads
$R_{write\_st\_i}$	Writes to storage	No. of writes
R_{nw_i}	Size of data transferred	GB
R_{NAT}	NAT Duration	hour
R_{NAT_tr}	Data size transfer through NAT	GB
R_{efs}	Data size in EFS	GB-month for storage
R_{codeSt_i}	Code size in container	GB-month for storage

storage service (S3), Google Cloud Storage and Azure Blob Storage. They are billed for the storage size and the number of operations performed on the objects.

IV. COST MODEL

In this section, we present the cost model for deploying workloads using AWS, Azure, GCP FaaS platforms, and storage services. All symbols used to denote cost parameters and metrics in this model are listed in Table I. Usage rates and parameters for pricing are vendor-dependent.

The total cost C of a multicloud pipeline is the aggregate cost of components from all N cloud providers

$$C = \sum_{i=1}^{N} C_i \tag{1}$$

where C_i is the cost incurred in each cloud, which is given by

$$C_i = C_{Ex\_i} + C_{St\_i} + C_{Nw\_i} + C_{Other\_i} \tag{2}$$

where C_{Ex_i}, C_{St_i}, C_{Nw_i}, C_{Other_i} are costs due to functions, storage, network, and other service usage respectively in the i^{th} cloud. Cost due to each of these services is calculted as follows

Functions cost: Serverless function executions are billed for the compute resources and the number of requests processed.

$$C_{Ex\_i} = C_{Compute\_i} + C_{Req\_i} \tag{3}$$

where $C_{Compute\_i}$ and C_{Req_i} denote the Compute cost and invocation costs in i^{th} cloud.

$$C_{Req\_i} = I_i * R_{req\_i} \tag{4}$$

where I_i is the total number of invocations and R_{req_i} is the cost per million invocations in the i^{th} cloud.

Total Compute cost $C_{Compute\_i}$ is the sum of the compute cost of all the functions in the cloud. For P_i functions in the i^{th} cloud, the compute costs are given by

$$C_{Compute\_i} = \sum_{j=1}^{P_i} C_{Compute\_ij} \tag{5}$$

where $C_{Compute\_ij}$ is the compute cost of the j^{th} function in i^{th} cloud. Compute cost for a function comprises the costs for the memory and CPU resources allocated or used for a function execution. The function resources are billed for the execution duration, as per the *pay-as-you-go* policy which is the core of the serverless paradigm.

Compute cost $C_{Compute\_ij}$ for the j^{th} function having Q_{ij} executions in the cloud i is given by

$$C_{Compute\_ij} = \sum_{k=1}^{Q_{ij}} t_{ijk}(m_{ij} * R_{mem\_i} + cpu_{ij} * R_{cpu\_i}) \tag{6}$$

where t_{ijk}, represents the execution duration for k^{th} invocation, m_{ij} is the memory configuration, cpu_{ij} is the CPU frequency of function j in cloud i. In AWS and Azure consumption plans, there is no separate billing for the CPU time and hence in both these cases, $R_{cpu\_i} = 0$.

In each cloud provider, there are options to keep a certain number of function instances warm (eg. Azure Premium Plan). Pre-warmed instances are billed even when there are no incoming requests or executions because the function instances remain active throughout the duration for which it is configured. The cost of pre-warmed instances C_{pw_ij} is calculated as

$$C_{pw\_ij} = npw_{ij} * pwt_{ij}(m_{ij} * R_{pw\_mem\_i} + cpu_{ij} * R_{pw\_cpu\_i}) \tag{7}$$

npw_{ij} number of pre-warmed instances which is configured for a duration of pwt_{ij} for the j^{th} function in the ith cloud.

Storage cost: Object Storage services in each cloud are billed for the size of the data stored and the operations on the data(read, write, delete). Cost due to storage (C_{St_i}) depends on the vendor and their operations

$$C_{St\_i} = D\_St_i * R_{st\_i} + \sum_{q=1}^{op} n\_st_{q\_st\_i} * R_{q\_st\_i} \tag{8}$$

where D_St_i denotes the total data size stored in cloud i, op is the different operations possible on storage service, $n\_st_{q\_st\_i}$ is the number of q^{th} operations done on storage service of the i^{th} cloud.

Network cost: All outbound data transfers incur network charges. This includes data to other cloud providers, as well as to other regions of the same cloud provider.

$$C_{Nw\_i} = \sum_{j=1}^{P_i} Q_{ij} * R_{nw\_i} * D\_StOut_{ij} \tag{9}$$

D_StOut_{ij} is the average data size transferred out by the j^{th} function in the i^{th} cloud. Q_{ij} is the number of executions of j^{th} function in the i^{th} cloud.

Other services cost: In addition to compute, storage, and network usage cost there are some other services that attribute to the total in cloud.

In AWS, we use EFS to store the prerequisites which exceed the allowed deployment size. It is billed for the storage size used.

$$C_{EFS\_AWS} = D\_EFS * R_{efs} \tag{10}$$

where D_EFS is the data size stored in EFS.

Additionally, AWS Lambda function needs VPC to access EFS. NAT Gateway is required for Lambda functions in VPC to access internet for data transfer across cloud. NAT Gateway is billed for the duration for which it is active and for the data that is transferred through the NAT gateway as given in the equation,

$$C_{NAT\_AWS} = t\_NAT * R_{NAT} + D\_tr * R_{NAT\_tr} \quad (11)$$

where t_NAT represents the duration for which NAT is configured, D_tr denotes all data transfer through the NAT Gateway - data transfer between function and external cloud storage service. So total additional cost in AWS due to EFS and NAT Gateway is derived using equation 10 and 11

$$C_{Other\_AWS} = C_{NAT\_AWS} + C_{EFS\_AWS} \quad (12)$$

In GCP, cloud functions code is stored as containers in the Container Registry which is billed for storage. Hence, additional cost in GCP is given as

$$C_{Other\_GCP} = D\_code * R_{codeSt\_i} \quad (13)$$

where D_code is the size of the code containers.

For most of the services, the cloud providers provide a free tier quota. The billing is done for the usage above the free quota.

V. Workload deployment on multicloud

In this section, we discuss the methodology for deploying AI application on multicloud FaaS platforms and storage services. We first decompose an application into a set of independent tasks, functions, microservices, or APIs (henceforth termed as application component). The refactoring decisions are made based on functionality of independent component of the application.

Next, we characterize each of these individual application components in isolation to understand their memory, compute, IO, data requirements (storage and network transfer). The characterization is performed offline in a controlled environment with machine configurations similar to the FaaS platform configurations [2]. The FaaS instances can be configured with different memory configurations, which impacts the performance and cost of the application components. Post characterization, we calculate the execution time and cost of deploying individual component using various configurations of different cloud FaaS platforms. Cloud object storage services are used to store, read and transfer data between components deployed using serverless functions. A benchmarking exercise is also conducted to understand the data transfer performance between FaaS platforms and storage services from multiple CSPs [23].

Considering there are n application components in a workflow, m CSP, s storage services available from CSPs, and each FaaS instance can be configured with c possible configurations. The total number of possible mappings M for the workflow can be calculated as

$$M = n * m * s * c \quad (14)$$

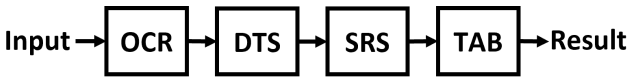


Fig. 1. Deep Reader workflow with API Optical Character Recognition (OCR), Data type Service (DTS), Spatial Relationship service (SRS) and Table extractor service (TAB)

In our coarse grain implementation, we predict the latency and cost for all possible mappings of the complete workflow consisting of multiple application components deployed in a multicloud environment. The overall latency is estimated using the execution time of independent component on a serverless function in isolation and latency due to data transfer between FaaS instance storage service within the same or different cloud as discussed above. Based on the latency of FaaS functions, size of data stored in storage service and network bandwidth used for transfer between the cloud providers, we estimate the cost incurred using the cost model discussed in previous section.

We estimate the performance and cost for all possible mappings in the multicloud setting. From this large number of mappings, we can choose the optimal workflows considering latency and cost. To verify accuracy of our estimated results, we select few single cloud and multicloud workflows resulting in best as well as worst performance and cost. The selected workflows are then deployed end-to-end in a multicloud setting for detailed analysis and comparison with estimated results as discussed in section VIII.

VI. Use Cases

In this section, we discuss two AI inference workloads that we used to study the performance and cost of multicloud deployment.

A. Deep Reader

The first use-case is our in-house document processing application called Deep Reader [27] that extracts text from document PDF or images and converts it to a query-able data. Deep Reader has four components or APIs namely Optical Character Recognition (OCR) which detects and recognizes text, Data type Service (DTS) which identifies the semantic data type of the words, Spatial Relationship service (SRS) which finds the neighboring blocks for every word or sentence blocks and Table extractor service (TAB) which detects the tables in the document using a custom model based on VGG 16 [28] and converts to CSV file. Document processing involves the use of one or more of these components. Thus we have a workflow with 4 APIs as shown in Fig. 1. The output generated by an API is consumed by one or more other APIs.

B. Digital Drawings

The second use case is Digital Drawings, which is a computer vision and deep learning application which extracts information about equipment and process flow from Piping and Instrumentation (P&ID) diagrams [29]. It is used

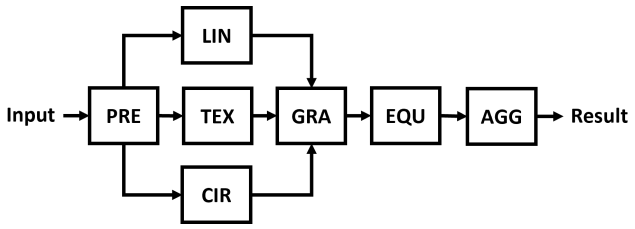


Fig. 2. Digital Drawings workflow with tasks Preprocess (PRE), Text (TEX), Lines (LIN), Circle (CIR), Graph (GRA), Equipments (EQU) and Aggregate (AGG)

to automatically digitize the P&ID drawings. The different APIs in the workflow are shown in Fig. 2. Preprocess task (PRE) resizes the input diagram to width of 8192 pixels and proportionate height for further processing. The text module (TXT) uses Character Region Awareness for Text Detection (CRAFT) model [30] for detection and Tesseract Engine for text recognition. The lines module (LIN) extracts the lines connecting different components denoting the process flow. The Circle API (CIR) detects the circles while Equipment API (EQU) detects basic shapes like square, rectangles, and diamond. Graph API (GRA) creates the basic layout of the P&ID network by considering components as vertices and the connections or flow between them as edges. Finally the aggregate (AGG) task combines information from all the modules, and assign each component a digital id and maps to its label, graph vertex, thus creating the digitized version of the process flow. Like Deep Reader, each of these APIs in the Digital Drawings produces an output that may be consumed by subsequent APIs in the workflow.

VII. Experimental setup

We have deployed all application components of both workflows to AWS Lambda, GCP functions and Azure functions. The storage service to be used is specified as a parameter in the trigger event of each workflow.

A workload of 50 documents of size varying from 75 KB to 1 MB, was processed by different Deep Reader APIs served by multicloud serverless functions. To understand how the APIs scale in each cloud provider and the impact of scaling on performance, the workload was processed in 3 batch sizes - 1, 10 and 50. Each of the experiments was conducted 3 times and the average is taken.

Out of all possible multicloud workflows theoretically possible, we consider a representative set of 15 and 10 workflows as mentioned in Table II and Table IV for Deep Reader and Digital drawing repectively. These workflows are selected from best and worst performing clusters in term of latency and cost. 15 deployment schemes of Deep Reader application represent mapping of four APIs in AWS, GCP, Azure premium (AZRP), Azure consumption (AZR) FaaS platforms as well as storage services. For example, experiment 1 in Table II represents the deployment of Deep Reader APIs OCR, DTS, SRS, and TAB respectively on AWS, Azure(AZR), AZR, and GCP serverless functions and storage on GCP. Memory configuration for the

TABLE II
API-Cloud mapping of Deep Reader in experiments

No.	OCR	DTS	SRS	TAB	Storage
1	AWS	AZR	AZR	GCP	GCP
2	GCP	AZR	AZR	AWS	AWS
3	AZRP	AWS	GCP	AWS	AWS
4	AZRP	GCP	AWS	GCP	AZR
5	AWS	GCP	GCP	AZRP	GCP
6	AWS	AWS	AWS	AWS	AWS
7	GCP	GCP	GCP	GCP	GCP
8	AZRP	AZR	AZR	AZRP	AZR
9	AWS	AZR	AZR	AWS	AZR
10	AWS	GCP	GCP	GCP	GCP
11	AZRP	AWS	AZR	GCP	AWS
12	AZR	AZR	AZR	AZR	AZR
13	AWS	AWS	AWS	AWS	AWS
14	GCP	GCP	GCP	GCP	GCP
15	GCP	AZR	AZR	GCP	GCP

TABLE III
Memory configuration (in GB) of serverless functions of Deep Reader APIs in different cloud providers

API name	AWS	GCP	Azure	Azure Premium
OCR	8	8	1.5	14
DTS	1	1	1.5	14
SRS	1	1	1.5	14
TAB	10	8	1.5	14

functions for all experiments except 13,14 is given in Table III, and is based on the resource characterization of the APIs. For experiment 13 and 14, memory configuration for OCR and TAB was set to 2 GB and DTS and SRS were allocated 1 GB.

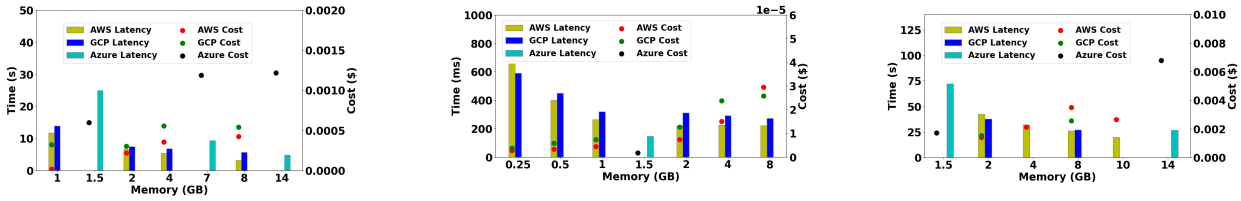
Similarly, for Digital Drawings, a workload of 10 diagrams of average size of 3 MB were processed in batches of 1, 5, and 10. Memory for all Digital drawing APIs except preprocess (PRE) is set to 8 GB in AWS and GCP. PRE module is allocated 2 GB. In Azure, we have used the Consumption plan, for which maximum memory available is 1.5 GB.

VIII. Experimental Analysis

In this section, we discuss our experimental analysis of multicloud deployment of AI workflows. We also present an evaluation of the cost model that we proposed. Experiment number in the figures denote the corresponding cloud cluster mapping mentioned in Table II and Table IV for Deep Reader and Digital Drawings respectively.

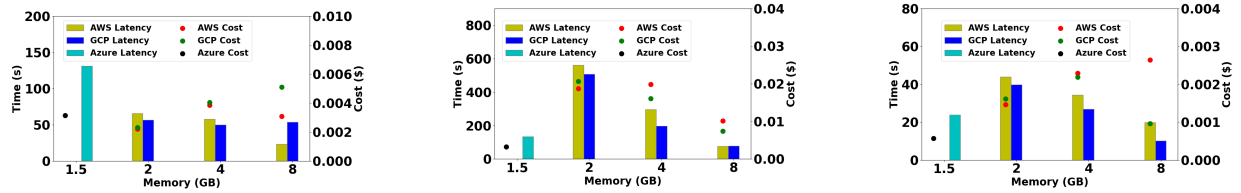
TABLE IV
API-Cloud mapping of Digital Drawings in experiments

No.	PRE	TEX	LIN	CIR	GRA	EQU	AGG	Storage
1	AWS	AWS	AWS	AWS	AWS	AWS	AWS	AWS
2	GCP	GCP	GCP	GCP	GCP	GCP	GCP	GCP
3	AZR	AZR	AZR	AZR	AZR	AZR	AZR	AZR
4	AZR	AWS	GCP	GCP	GCP	AWS	AZR	AWS
5	AWS	AWS	GCP	GCP	GCP	AWS	GCP	GCP
6	GCP	GCP	AWS	AZR	AZR	AWS	AWS	AZR
7	AZR	AWS	AZR	GCP	AWS	GCP	AWS	AWS
8	GCP	AWS	GCP	GCP	AWS	AWS	AWS	AWS
9	AZR	GCP	AZR	AZR	GCP	GCP	GCP	GCP
10	AZR	AWS	AZR	AZR	AWs	AWS	AWS	AZR



(a) (b) (c)

Fig. 3. Effect of AWS Lambda, Azure function, and GCP Functions FaaS instance memory configuration on latency and cost for Deep Reader APIs (a) OCR (b) DTS and (c) TAB



(a) (b) (c)

Fig. 4. Effect of AWS Lambda, Azure function, and GCP FaaS instance memory configuration on latency and cost for Digital Drawings APIs (a) text (b) equipment and (c) aggregate

A. Effect of FaaS memory configuration on performance and cost

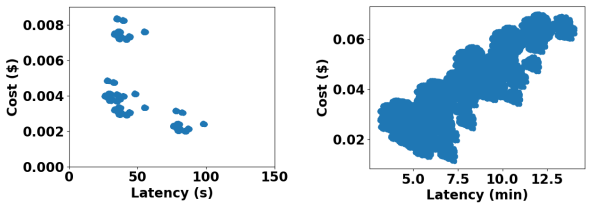
In this experiment, we study the effect of changing memory configurations on latency and cost of deployment on AWS, Azure, and GCP FaaS instances. Fig. 3 and 4 show the effect of memory configuration on three major APIs of Deep Reader and Digital Drawings respectively. We can conclude following from these figures

- The execution time and cost of an API vary significantly with change in the memory configuration and hence compute power of the FaaS instance.
- The execution time and cost also vary significantly with the choice of FaaS platform Lambda, Azure function, and Google function.
- Best performance is achieved with AWS Lambda at higher memory configuration (8GB)

B. Estimated and Actual Performance of the selected workflows

As discussed in section V, we estimate the latency and cost for different workflows generated by mapping various APIs to multicloud FaaS platforms, configurations and storage services. The predicted latency and cost for the different combinations of API-FaaS-storage mappings of Deep Reader and Digital Drawings is shown in Fig. 5 and Fig. VIII-B respectively.

In this experiment, we compare the predicted latency with the observed latency of some selected workflows listed in Table II and Table IV for our two usecases. Fig. 6 and Fig. 7 shows the comparison of estimated and actual latency as well as cost for the selected Deep Reader and Digital Drawings workflows respectively. The cost of deployment is estimated using cost models discussed in section IV. Both latency and cost for the selected workflows are predicted with less than 10% error in most of the cases.



(a) (b)

Fig. 5. Latency and cost estimation of (a) Deep Reader and (b) Digital Drawings for possible workflows in multicloud setting

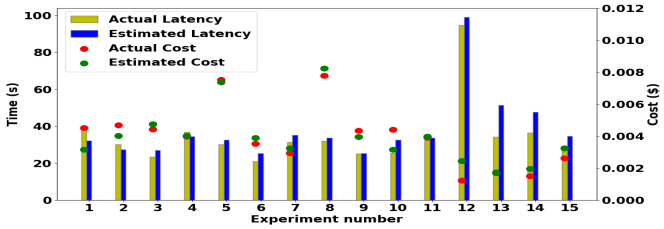


Fig. 6. Comparison of estimated and actual latency as well as the cost of selected Deep Reader workflows in multicloud setting. The cloud cluster used in each experiment is as mentioned in the Table II

C. Single cloud vs multicloud deployment

In this experiment, we compare the cost and performance of deploying AI workflows in a single cloud and multicloud environment. Fig. 8 shows the observed latency and cost of Deep Reader workflow for the different API-FaaS-storage combinations. We execute a workload of 50 document images in the batches of 1, 10 , and 50 images at a time.

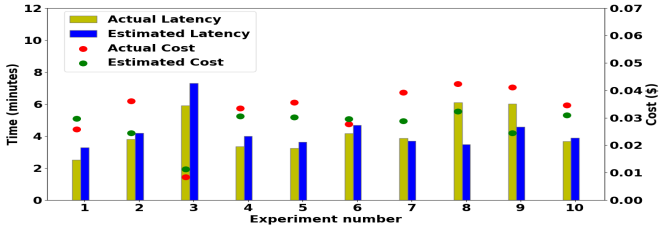


Fig. 7. Comparison of estimated and actual latency as well as the cost of selected Digital Drawings workflows in multicloud setting. The cloud cluster used in each experiment is as mentioned in the Table IV

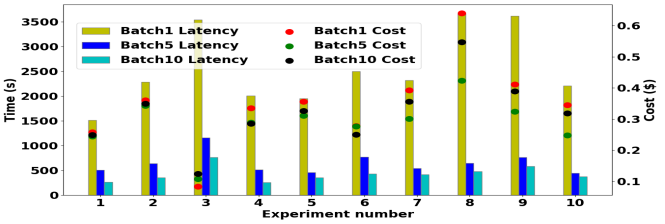


Fig. 9. Total time and cost for processing 10 images using Digital Drawings in batches of 1, 5, 10 images concurrently. The cloud cluster used in each experiment is as mentioned in the Table IV

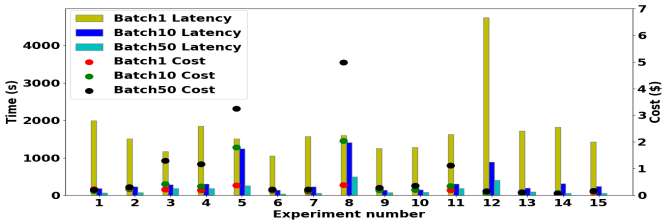
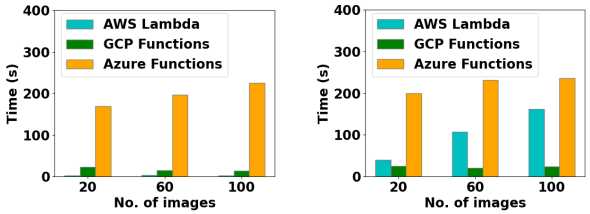


Fig. 8. Total time and cost for processing 50 images using Deep Reader in batches of 1, 10, 50 images concurrently. The cloud cluster used in each experiment is as mentioned in the Table II



(a) (b)

Fig. 10. Effect of (a) warm start and (b) cold start on the execution time of OCR API varying workload in AWS Lambda (8 GB), GCP functions (8 GB), and Azure functions (1.5 GB)

In general, there is a significant improvement in the total time taken to process a workload when using multicloud combinations (experiments 1-5,9-11,13-15) rather than just a single cloud alone (experiments 7 and 12). An exception to this is the AWS-only scenario (experiment 6).

For the serial processing of a workload of 50 images using batch size 1, all pipelines have comparable latency, except for Azure functions consumption plan pipeline (Experiment 12). For batch size of 10, pipelines with Azure Premium (AZRP) functions perform worse than Azure consumption plan (Experiments 5,8).

For Digital Drawings (Fig. 9), the latency and cost trend for processing 10 images is similar for different batch sizes (1, 5, 10). Unicloud deployment using AWS Lambda and storage service S3 delivers the least latency when images are processed serially. When batch size is increased, multicloud mappings perform better than AWS alone mapping. For instance, Experiments 5,10 for batch size 5 and Experiment 4 for batch size 10. Unicloud Azure Consumption is the worst in terms of latency, however, it has the least cost.

From these experiments, we can conclude that the choice of cloud functions for deploying individual API results in a significant change in the overall execution time of the workflow. Average latency per image improves by 47% (Experiment 6,1) in Deep Reader and 58% (Experiment 1,8) in Digital Drawings. The cost per image reduces by 83% (Experiment 12,8) in Deep Reader and 80% (Experiment 3,8) in Digital Drawings.

D. Scaling and cold starts

In this experiment, we study the effect of cold start and warm start on the execution time of APIs and workflows. The cold start latency in serverless functions is due to time spent in preparing the execution environment, loading the function and dependency in the FaaS instance, etc. When there is a burst of workload, scaling is handled by the cloud provider. Several new function instances are instantiated at the same time, resulting in higher cold starts. For cold starts, the performance of GCP is better than AWS and Azure. If the traffic is bursty, we face cold starts more often. As shown in Fig. 10(a), GCP functions are a better choice over AWS Lambda and Azure functions. However, if we have consistent traffic which keeps FaaS instance warm, then AWS Lambda is a better choice as it has low latency and consistent performance as shown in Fig. VIII-B.The impact of cold start and warm start with changing workload for Deep Reader workflow is as shown in Fig. 11. For experiments 2,7,9,10 there is no significant difference in the cold start time and warm start time for processing workloads of 20 and 80 images. GCP functions is used for these experiments except 9.

E. Data transfer between FaaS platforms and storage services

In this experiment, we study the data download and upload transfer time between FaaS platforms and storage services for varying file sizes. Fig. 12(a) and 12(b) show the data download performance for small and large files respectively. Both the serverless function as well as the storage service plays a role in the transfer rate.

Case 1: Data download within a cloud As shown in Fig. 12(a) for small file sizes and Fig. 12(b) for medium and large size file transfers, the fastest data transfer is achieved between AWS FaaS platform Lambda and storage S3. The slowest data transfer is observed between Azure function

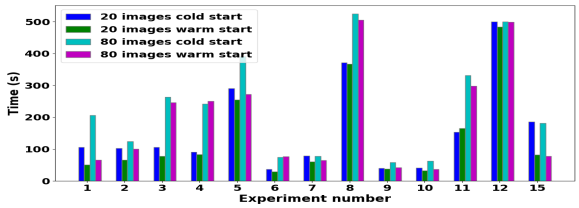
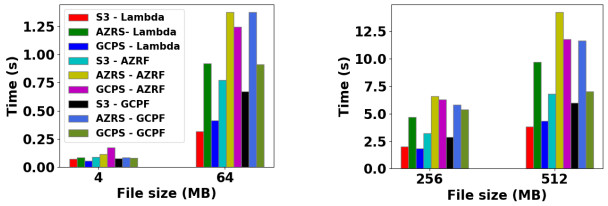


Fig. 11. Cold start and warm start latency observed in Deep Reader experiments for workload of 20 images and 80 images. The cloud cluster used in each experiment is as mentioned in the Table II



(a) (b)

Fig. 12. Time taken for download from storage services to serverless functions of AWS, Azure, and GCP for low, medium, and large files

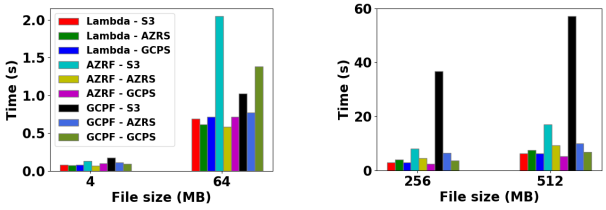
(consumption plan) and its blob storage service.

Case 2: Data download across cloud When the cross-cloud transfer happens between the FaaS platform of one cloud and storage of other, we observed that best performance is observed for data transfer between Google cloud functions and AWS storage S3. The slowest data transfer is observed for Google cloud function and Azure storage as well as Azure functions and Google storage.

When case 1 and case 2 is compared for all file sizes, the slowest data transfer is observed between the Azure function and Azure blob storage (case 1) and the fastest data transfer is observed between AWS Lambda and S3 (case 1). Fig. 13(a) and 13(b) show data upload performance for small and large size files respectively.

F. Cost model evaluation

We proposed a cost model in section IV to estimate the cost of deploying workflows using FaaS platforms and storage services in multicloud environment. In this experiment, we evaluate the accuracy of our cost model. In our evaluation, we extract from cloud vendor's billing dashboard the monthly cost of using the individual services involved in deploying workflows. Also, the usage for the billing period is fetched from each of the cloud service provider log metrics, which is used in the cost model presented in section IV. Fig. 14 shows the comparison of the actual and estimated costs for using FaaS platforms AWS Lambda, Azure functions, Google functions and storage services S3, Azure Blob, and Google cloud storage. The figure shows the cost of using individual services from various CSPs for 30 days. During this period we run multiple workloads using various workflows for Deep Reader and Digital Drawings. As shown in the figure, the total



(a) (b)

Fig. 13. Time taken for data upload from serverless functions to the storage services of AWS, Azure, and GCP for low, medium, and large files

cost of running our two workflows with varying workloads for 30 days is predicted with less than 2% error.

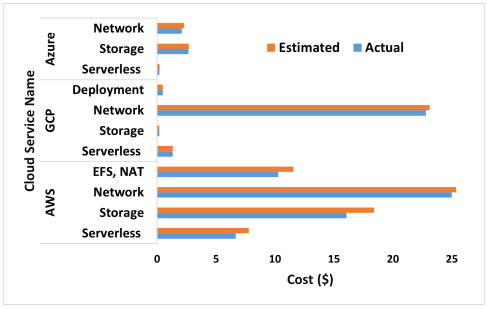


Fig. 14. Comparison of actual bill and the cost estimated by the cost model for different FaaS platforms, storage and other services used for multicloud workflows

We can conclude the following from the results presented above:

- Some multicloud deployment schemes of AI workflows using the FaaS platform and storage services results in a better performance and cost as compared to the single cloud deployment configurations.
- The cost of multicloud deployment can be estimated successfully using a robust cost model.

IX. CONCLUSION

We have presented a detailed experimental analysis of performance and trade-offs while deploying AI workflows in a multicloud setting. We deployed information extraction AI workflows using FaaS platforms from AWS, Azure, and GCP and their storage services. Our study also shows that multicloud deployment of AI workloads based on their characterization data results in performance and cost benefits. We show that cold start and data transfer rate in a multicloud environment results in a significant differences in performance and cost.

We proposed a cost model for predicting the cost of workflow deployment using AWS, Azure and Google cloud FaaS platforms and storage services. The proposed comprehensive models include cost from multiple cloud services.

For larger workflows with more tasks and dependencies, the mapping of workflows to FaaS platforms becomes a complex problem, which requires optimization algorithms for efficient deployment. This work is currently ongoing.

REFERENCES

[1] Azure, "Azure functions," Accessed August 10, 2022, 2021. [Online]. Available: https://azure.microsoft.com/en-in/services/functions/

[2] D. Chahal, S. C. Palepu, and R. Singhal, "Scalable and cost-effective serverless architecture for information extraction workflows," in *Proceedings of the 2nd Workshop on High Performance Serverless Computing*, ser. HiPS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 15–23. [Online]. Available: https://doi.org/10.1145/3526060.3535458

[3] D. Chahal, M. Ramesh, R. Ojha, and R. Singhal, "High performance serverless architecture for deep learning workflows," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 790–796.

[4] D. Chahal, R. Ojha, M. Ramesh, and R. Singhal, "Migrating large deep learning models to serverless architecture," in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2020, pp. 111–116.

[5] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1–14. [Online]. Available: https://doi.org/10.1145/1879141.1879143

[6] Amazon, "Aws s3," Accessed August 6, 2022, 2022. [Online]. Available: https://aws.amazon.com/s3

[7] Azure, "Azure storage blobs," Accessed August 10, 2022, 2021. [Online]. Available: https://azure.microsoft.com/en-in/services/storage/blobs/

[8] G. Cloud, "Cloud storage." Accessed August 06, 2022, 2022. [Online]. Available: https://cloud.google.com/storage

[9] P. Wangsom, K. Lavangnananda, and P. Bouvry, "Multi-objective scientific-workflow scheduling with data movement awareness in cloud," *IEEE Access*, vol. 7, pp. 177 063–177 081, 2019.

[10] T. SARAVANAN, K. Jeyaraj, and G. Nagarajan, "Workflow scheduling for intelligent modern medical system with privacy protection constraints for iaas cloud using modified grasshopper algorithm," 2022.

[11] D. Chahal, R. Ojha, S. R. Choudhury, and M. Nambiar, "Migrating a recommendation system to cloud using ml workflow," in *Companion of the ACM/SPEC International Conference on Performance Engineering*, 2020, pp. 1–4.

[12] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 581–594, 2016.

[13] W. Guo, B. Lin, G. Chen, Y. Chen, and F. Liang, "Cost-driven scheduling for deadline-based workflow across multiple clouds," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1571–1585, 2018.

[14] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "Cost-effective workflow scheduling approach on cloud under deadline constraint using firefly algorithm," *Applied Intelligence*, vol. 51, no. 3, pp. 1629–1644, 2021.

[15] C. Georgios, F. Evangelia, M. Christos, and N. Maria, "Exploring cost-efficient bundling in a multi-cloud environment," *Simulation Modelling Practice and Theory*, vol. 111, p. 102338, 2021.

[16] X. Tang, "Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2021.

[17] A. Mohammadzadeh and M. Masdari, "Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–21, 2021.

[18] K. K. Chakravarthi, P. Neelakantan, L. Shyamala, and V. Vaidehi, "Reliable budget aware workflow scheduling strategy on multi-cloud environment," *Cluster Computing*, pp. 1–17, 2022.

[19] Q.-H. Zhu, H. Tang, J.-J. Huang, and Y. Hou, "Task scheduling for multi-cloud computing subject to security and reliability constraints," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 848–865, 2021.

[20] J. G. Quenum and J. Josua, *Multi-Cloud Serverless Function Composition*. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3468737.3494090

[21] A. F. Baarzi, G. Kesidis, C. Joe-Wong, and M. Shahrad, "On merits and viability of multi-cloud serverless," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 600–608. [Online]. Available: https://doi.org/10.1145/3472883.3487002

[22] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2021, pp. 26–32.

[23] S. C. Palepu, D. Chahal, M. Ramesh, and R. Singhal, "Benchmarking the data layer across serverless platforms," in *Proceedings of the 2nd Workshop on High Performance Serverless Computing*, ser. HiPS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3–7. [Online]. Available: https://doi.org/10.1145/3526060.3535460

[24] A. F. Baarzi, G. Kesidis, C. Joe-Wong, and M. Shahrad, "On merits and viability of multi-cloud serverless," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 600–608. [Online]. Available: https://doi.org/10.1145/3472883.3487002

[25] A. Al-maamari and F. Omara, "Task scheduling using pso algorithm in cloud computing environments," *International Journal of Grid and Distributed Computing*, vol. 8, pp. 245–256, 10 2015.

[26] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "Adpso: Adaptive pso-based task scheduling approach for cloud computing," *Sensors*, vol. 22, no. 3, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/920

[27] V. D, R. Rahul, G. Sehgal, Swati, A. Chowdhury, M. Sharma, L. Vig, G. Shroff, and A. Srinivasan, "Deep reader: Information extraction from document images via relation extraction and natural language," 2018. [Online]. Available: https://arxiv.org/abs/1812.04377

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[29] S. Paliwal, A. Jain, M. Sharma, and L. Vig, *Digitize-PID: Automatic Digitization of Piping and Instrumentation Diagrams*, 05 2021, pp. 168–180.

[30] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," *CoRR*, vol. abs/1904.01941, 2019. [Online]. Available: http://arxiv.org/abs/1904.01941