Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

INTELLIGENT CLOUD STORAGE MANAGEMENT USING HYBRID TECHNIQUES

MUHAMMAD ABID SALEEM *
Department of Computer Science, The Islamia University Bahawalpur, Punjab Pakistan.
*Corresponding Author Email address. abidsaleemiub@gmail.com

ZAIGHAM MUSHTAQ
Department of Computer Science, The Islamia University Bahawalpur, Punjab Pakistan

MUHAMMAD SULEMAN
Department of Computer Science, The Islamia University Bahawalpur, Punjab Pakistan

OMER RIAZ
Department of Computer Science, The Islamia University Bahawalpur, Punjab Pakistan

Abstract

Cloud applications that utilize the "Functions as a Service" (FaaS) model, have gained significant popularity. However, their stateless nature necessitates frequent interaction with an external data store, which can limit their performance. To address this issue, a system for FaaS platforms that is transparent, vertically and horizontally elastic, and distributed across worker nodes. It mitigates effectively the problem by leveraging two common sources of resource wastage (I) cloud tenants typically observe overprovision memory resources for their functions due to their non-trivial input-dependence, and (ii) FaaS providers keep function sandboxes active for several minutes to avoid cold starts. By utilizing machine learning models tailored to different types of function input data including data mining, multimedia applications and gaming. We employed a novel technique that classifies the data in Azure Storage Blobs in hot and cold. Hot data is used frequently and cold data is not frequently required. There are no straight forward annotations or features that directly label the hot items. We have to examine the attributes of each data item and segregate them into classes. We employed a Naïve Bayes classifier to mark the data items with improvisation of Sine-Cosine optimization for hyper parameter tuning. This methodology significantly boosts the accuracy from 57.44% to 94.34%.

Index Terms: Resource Discovery, Resource Allocation, function as service, amazon web service, Stateless, Virtual Machine

1. INTRODUCTION

The Function as a Service (FaaS) has gained popularity among cloud service providers due to its simplicity and developer-centric approach. Major cloud providers offer their own FaaS solutions, with their own branding and features. For instance, Amazon Web Services (AWS) provides AWS Lambda, Red Hat offers Open Shift, and Microsoft has Azure Functions [1-3]. Traditionally, cloud vendors provided virtual machines (VMs) or containers for consumers to deploy and manage their applications. However, the FaaS model revolutionizes the design, operation and billing of cloud-native applications. Instead of managing complex infrastructure, developers can focus solely on writing application code, organized as stateless functions triggered by events [4]. This server-less approach allows developers to offload the burden of infrastructure management to the cloud

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

provider and concentrate on business logic and application development. FaaS offers several advantages for developers and businesses. One key benefit is to improve resource efficiency. In a FaaS environment, resources are allocated dynamically and automatically scale based according to demand. Functions are executed only when triggered by events, ensuring optimal resource utilization and cost-effectiveness [5]. Another advantage of FaaS is its pay-as-you-go pricing model. With FaaS, developers are charged only for the actual resources consumed during the execution of their functions. This granularity enables cost savings, as users are not billed for idle resources, but rather for the specific usage of their applications [6]. Furthermore, FaaS promotes faster development cycles and increased agility. By abstracting away the underlying infrastructure, developers can focus on writing modular and reusable functions, enabling faster deployment and easier maintenance of cloud-native applications [7].

1.1. Attributes of Server less

Trigger based Execution: FaaS platforms are designed to execute functions in response to specific events or triggers. These events can include HTTP requests, database events, file uploads, message queue events, timers and many more. These events can be generated by human activity or automatically. Events based applications provide a loosely coupled components that not only accelerate the software development process but simplifies the deployment and consequently, the system orchestration becomes more abstract [8].

Scalability: FaaS platforms automatically handle the scaling of function instances based on the incoming workload. Functions can scale up or down dynamically to accommodate varying levels of demand, ensuring efficient resource utilization and high availability. FaaS enables a cloud infrastructure to increase resource provision without prior alarm. It means the scalability is itself a complex task and requires lot of research to implement a flawless structure of elasticity [9-11].

Pay-per-use billing: Economy is also an important effect of cloud computing. The reason behind the popularity of cloud is its payment pattern. FaaS platforms often follow a pay-per-use pricing model. You are charged based on the number of function invocations and the resources consumed during execution, rather than paying for dedicated infrastructure or idle time [12]. Two important factors also included in cloud costing methods: quality of service (QoS) that identifies the user satisfaction [13] while the SLA indicates the performance of cloud vender [14].

Multi-cloud support: FaaS offerings are available across various cloud service providers, allowing developers to choose the platform that fits the best according to their requirements. This enables portability and flexibility in deploying functions across different cloud environments.

Integration with other cloud services: As we know that FaaS enables micro services paradigm which more efficient and independent compartments of an application [15]. FaaS platforms often provide seamless integration with other cloud services, such as storage, databases, message queues, authentication services, and more. This makes it

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

easier to build server free applications that leverage a wide range of cloud resources and services [16].

Fault tolerance and resilience: FaaS platforms typically provide built-in fault tolerance and resilience mechanisms. Functions are executed in a distributed environment, and the platform automatically handles failures, retries and recovery, ensuring high availability and fault tolerance. There are two main fault tolerance methods are deployed: (i) reactive which comes to action when some fault occurs, (ii) proactive, which prevents the faults to appear [FaaS support both type of fault tolerance techniques [17]

Table 1: List of Avsilable Faas Solution

Vendor	FaaS Service Name
Microsoft	Azure Function
Red Hat	Open Shift Server less
Amazon	Lambda Function
Google	Google Cloud Function
Oracle	Oracle Cloud Function
OpenFaaS	OpenFaaS
Akamai	Edge Workers
Digital Ocean	Digital Ocean Functions
Cloudflare	Cloud flare Workers

1.2. Azure Blob Storage

Blob (Binary Large Object) storage is different kind of storage that stores the data into a special object format which is more easily accessible by cloud servers. This a flat storage system with the comparison of file based or block-based storage [18]. Blobs are fairly great advantages rather than conventional storage systems such as scalability, programming language independence and cost effectiveness. The requirements of blob storage are not only for currently demanded items but also support multimedia files, monitoring logs, backup and disaster recovery management [18]. The windows azure provide blob storage in three distinct layers. First one is the account which is associated with physical entity such as human or organization. Second layer is containers who manages the underneath blobs. The last layer is further consisted on three types of blobs. Block blobs that manage the data into text or binary format but these are stationary in nature. Append blobs are define for continuously updating and refreshing data such as monitoring logs and third is page blob that use for arbitrarily access files [19].

All the operations such as read, write and update are performs on directly files but Windows Azure do not manage files directly.

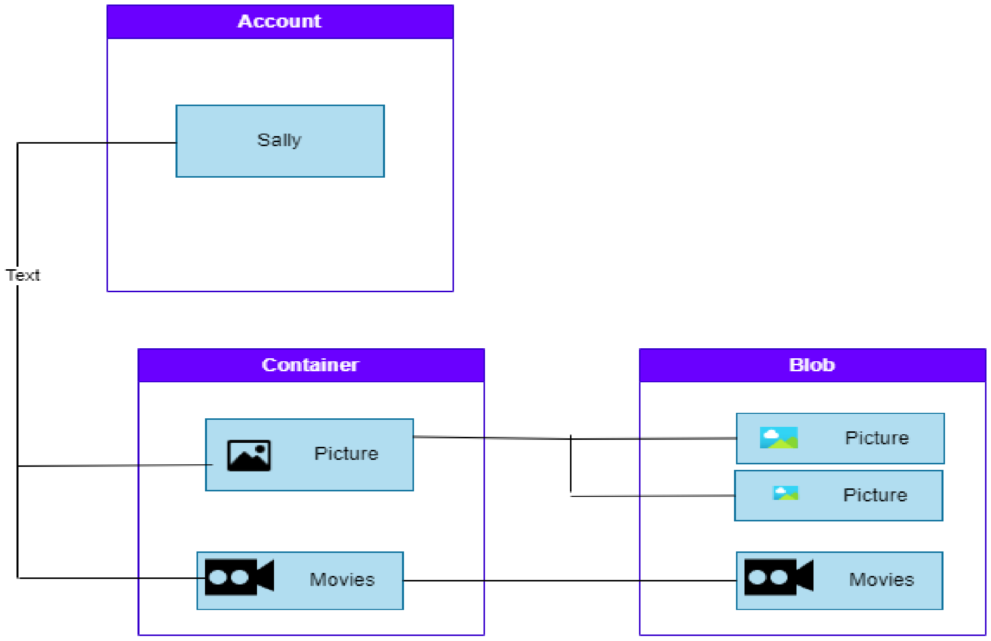

Figure 1: Windows Azure Blobs Storage

This research paper focuses on following problems to solve while using machine learning:

1) Cloud are large infrastructures of IT hardware and lot of heterogeneity involved in storage devices. Mainly two leagues created. One is slower paced storage. These devices are HDD technology based and rather cheap in cost. The other league is rapidly response SSD devices and obviously requires more cost. Cloud is shifting on faster technologies to meet high demand in service but installing a complete new set of hardware is not feasible. An intuitive solution of such situation is that Cloud management segregate the demanded data (files, objects, blobs etc.) into hot and cold classes. Hot items remain in faster technology so that their turn-around time may be reduced and may increase quality of service while cold components store at old slow response drives to cut down the cost.

2) Machine learning may be helpful to identify the high and low demanded data on the base different features due to its classification capability. The underneath dataset has a bunch of attributes which are helpful to define an instance as hot or cold. We employed Naïve Bayes classifier to label the instances.

3) The provided dataset has a large number of fluctuations in instance formation. It is difficult to classify them as hot and cold items. That's why we implement the Sine-Cosine optimization. This optimization technique is helpful in hyper parameter tuning for machine learning model to increase the accuracy and precision.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

2. LITERATURE REVIEW

According to B. Rochwerger *et al.* Optimization technique while NB is a classification algorithm we have a lot of traces production in cloud utilization of resources. To optimize the classification process on the data logs, we have to develop an integrated solution. The optimization method identifies the traces and as the result it will able to predict best available resources. In recent past, Federated, Mobile Cloud, Edge Computing, Peer to Peer and many other technologies evolve and emerge under the umbrella term of Cloud [20]. M. Taghipour, M. Soofi, M. Mahboobi, and J. Abdi stated in the application of cloud computing that they have common objectives of providing infrastructure, resources and services [21] on larger scale for different kind of online activities including processing, streaming, gamming, analytics and backups. M. Masdari and A. Khoshnevis also acknowledged it. [22]. Cloud (and its child infrastructures) provides virtually unlimited set of resources for high-end performance. A. Kumari, S. Tanwar *et al.* To achieve the illusion of limitless list of resources, they perform multiplexing of resources. Most of QoS & QoE based on more appropriate discovery and selection of resources in such arrangement [23].

R. N. Calheiros *et. al.* explain how the management system monitors the usability of resources and keep track under and over provision of resources. There are multiple factors that involve in resource discovery and utilization schemes such as location, frequency of utilization and performance of the resource. The resource discovery and employment structure also compete the issues of scalability, load balancing heterogeneity and compatibility with nature of workload The overall resource discovery issue related to different kind of components [24].

Machine learning for intelligent cloud storage management. W. Chen, B. Liu, I. Paik, Z. Li, and Z. Zheng, Review the literature on various machine learning techniques, including deep learning, reinforcement learning, and Bayesian networks, and highlight the key challenges and opportunities in this area [25]. Reinforcement learning-based approach to resource allocation in distributed storage systems. H. Liu develop an algorithm that learns to optimize resource allocation based on feedback from the system, and demonstrate its effectiveness in a simulation environment [26]. Machine learning-based approach to data placement in geo-distributed cloud storage systems. The L. Mercl and J. Pavlik develop a model that predicts the workload of each data center, and use this information to optimize data placement for improved system performance [27]. Privacy-preserving cloud storage system that uses machine learning techniques to protect user data. Z. Yu develop an encryption scheme that uses machine learning to learn the encryption key based on user behavior, and demonstrate its effectiveness in a simulated environment proposes a privacy-preserving cloud storage system that uses machine learning techniques to protect user data. The authors develop an encryption scheme that uses machine learning to learn the encryption key based on user behavior, and demonstrate its effectiveness in a simulated environment [28]. X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao proposes a [29] deep reinforcement learning-based approach to intelligent cloud storage management. R. Li *et al.*, develop a model that learns to optimize storage performance based on

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

feedback from the system, and demonstrate its effectiveness in a real-world cloud storage environment. Machine learning-based approach to resource management in cloud storage systems. The authors develop a model that predicts resource demand based on historical usage data, and use this information to optimize resource allocation [30].

Proposes a scalable cloud storage system that uses machine learning to optimize system performance. The P. S. Rawat develop a model that learns to predict system performance based on various factors, such as workload and network congestion, and use this information to optimize system configurations [31]. Reinforcement learning-based approach to intelligent cloud storage management that can adapt to dynamic system configurations. The Y. Zhang, J. Yao, and H. Guan develop an algorithm that learns to optimize storage performance based on feedback from the system, and demonstrate its effectiveness in a simulated environment [32].

Hybrid machine learning-based approach to secure data storage in cloud computing. The P. Sonia and R. Malika develop a model that combines supervised and unsupervised learning to detect and prevent cyber-attacks, and demonstrate its effectiveness in a simulated environment [33].

Use of machine learning for intelligent storage management in edge computing environments. The X. Liu, J. Yu, J. Wang, and Y. Gao review the literature on various machine learning techniques, and highlight the key challenges and opportunities in this area [34]. Overall, these references illustrate the wide range of topics and techniques being explored in the literature on intelligent cloud storage management. From resource management and optimization to security and privacy, machine learning is being used to tackle some of the biggest challenges facing. Proposes a machine learning-based approach to optimizing cloud storage costs for cold data. The A. Erradi and Y. Mansouri develop a model that learns to predict the access frequency of data and uses this information to determine the optimal storage tier for each data item [35].

Analysis of the storage consumption patterns in public clouds, with a focus on the tradeoffs between hot and cold storage. The Y. Mansouri and R. Buyya explore various storage tiers and their associated costs, and provide recommendations for optimizing storage consumption [36].

Explores the challenges and opportunities associated with managing hot and cold data in cloud storage systems. The H.-E. Chihoub *et al.* review the literature on various storage tiers and management strategies, and provide insights into the tradeoffs between cost, performance, and reliability [37]. Hybrid approach for efficient cold data storage in cloud-based archival systems. The A. Alzahrani *et al.* develop a model that combines machine learning and heuristics to optimize data placement and retrieval for improved system performance [38]. Hot and cold storage have different advantages and disadvantages depending on the specific use case and the characteristics of the data being stored.

Ristov *et al.* introduces FaaSt, a novel list-based function choreography (FC) scheduler for federated Function-as-a-Service (FaaS) platforms. It addresses the challenges of heterogeneous FaaS environments by optimizing the makespan of FCs running in

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

federated FaaS. FaaSt focuses on selecting the appropriate FaaS provider, cloud region, and memory settings for scientific FCs, which have longer function runtimes and complex compute and I/O operations [39]. The evaluation demonstrates that FaaSt outperforms other schedulers, achieving up to 2.82x speedup when running FCs across multiple cloud regions compared to a single region. It also achieves up to 1.74x speedup compared to other state-of-the-art FC schedulers in the same multi-region setup.

Copik, Marcin, et al work on "Process-as-a-Service: FaaSt Stateful Computing with Optimized Data Planes" introduces a new concept called Process-as-a-Service (PaaS) within the context of Function-as-a-Service (FaaS) environments. The main focus is on addressing the challenges of managing stateful operations and data in serverless computing. The proposed approach involves optimizing the data plane to effectively handle state and ensure smooth execution of stateful processes. The paper provides insights into the design and implementation of the PaaS framework, which utilizes containerization and data plane optimizations. The evaluation demonstrates the framework's ability to deliver performance and scalability benefits. It emphasizes the significance of optimized data planes in supporting stateful computations and managing complex workflows within the FaaSt model [40]. Overall, the paper presents a promising approach to stateful computing in serverless architectures and highlights the potential of Process-as-a-Service to enhance the efficiency and capabilities of FaaS platforms. PATIL, R. S., KOTWAL, A. Stated in his work that cloud computing has revolutionized the storage and access of data, but it also poses security challenges that impede its development. Data security in the cloud has become a major concern due to frequent hacking incidents. To address this, the article proposes a machine learning-based method for secure data storage in the cloud. The approach involves compressing the data using the Huffman algorithm to minimize storage and transmission requirements, followed by encrypting the compressed data using a novel cryptographic technique. Additionally, a Weighted Chimp Algorithm optimized Gaussian Kernel Radial Basis Function Neural Network is proposed to detect and identify malicious code in the cloud platform. The proposed method demonstrates higher performance compared to other existing methods such as Fully Homomorphic Encryption (FHE), Ciphertext Policy-Attribute based Encryption (CP-ABE), and Quasi Modified Levy Flight Distribution Reversed Sheamir Algorithm (QMLFD-RSA). The results show improved deduplication rate, throughput, cipher text, and encryption time, indicating that the proposed method offers better security and enhanced retrieval of information in a protected manner [41]. Here are some general advantages and disadvantages of each: Automatically recommending storage space allocation in object-based cloud storage systems. The goal is to optimize storage resource utilization and performance by dynamically allocating space based on the characteristics and usage patterns of stored objects. The proposed system leverages machine learning algorithms to analyze various object attributes and usage data to predict future storage requirements. It employs a regression model to estimate the required storage space for new objects and recommends appropriate allocation strategies. Experimental results demonstrate the effectiveness of the approach in improving storage efficiency and minimizing resource wastage. Overall, the system offers an automated and intelligent solution for storage space recommendation in object-based cloud storage systems, enhancing performance and resource utilization.

Table 2: Comparison of Literature Summary of Efficient Resource Management of Cloud Computing

Sr.#	Ref.	Name	Year	Proposed Method	Tools	Problem
1	[42]	Xiang et al	2022	Monitoring energy-adaptive network functions. framework	Ubuntu- 20.04 LTS Cisco TRex Docker XDP-Tools Turbostat	Reducing power consumption
2	[43]	Kim, Lee, Kim, & Pack	2020	DQN based Algorithm	Not described	Minimize back-haul control traffic cost
3	[44]	Shah, Gregory, & Li	2021	Envisions a cloud native 5G microservices Architecture	Kubernetes	Cloud native 5G core study and design
4	[45]	Wu, Wang, Yan, Wu, & Lu	2022	Intrinsic Cloud Security framework with heterogeneous resource pool management	Not described	Security
5	[46]	Imadali & Bousselmi	2018	5GaaS service architecture and 5G CN-VNF framework	OAI	Review current NFV Management Solutions
6	[47]	Kim, Lee, Kim, Pack, & Management	2022	DQN based algorithm	Not described	Minimize back-haul control traffic cost, CNF launching costs, and CNF operating costs
7	[48]	Qiang, Chunming, Xincheng, Qiumei, & Management	2021	Intrinsic Cloud Security Framework	DPDK	Security
8	[49]	Lee et al., 2021	2021	Automate the load balancer deployment	DPDK Cisco TRex	Load balancer Deployment
9	[50]	Sharma, Miller, & Francini	2017	A cloud native approach to network slicing	Linux VM Djiang Nginx PostgreSQL OpenStack Apache projects	Lifecycle management
10	[51]	Leconte, Paschos, Mertikopoulos, & Kozat	2018	Alternating direction method of multipliers algorithm	Not described	Resource allocation

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

11	[52]	Schmidt, Nikaein, & Management	2021	Designs a service-oriented RAN architecture	OAI Mosaic5G	Resource allocation for RAN
12	[53]	Mudvari, Makris, & Tassiulas	2021	Uses LSTM RNN model to predict network load in the future	OAI Kubernetes	Resource allocation for RAN
13	[54]	Y. Wu et al.	2022	A network slicing management architecture for IoT applications	Not described	Network slicing management
14	[55]	Boudi, Bagaa, Pöyhönen, Taleb, & Flinc	2021	DRL algorithm	Kubernetes	Resource allocation and minimize slice migration overhead
15	[56]	Bolla et al.	2021	Validates the MATILDA platform	MATILDA	Lifecycle management
16	[57]	Abbas, Khan, Afaq, & Song	2021	Uses LSTM RNN model to predict future resource utilization	OpenStack OAI IBN tool	Lifecycle management
17	[58]	Mekki, Arora, Ksentini, & Management	2021	Uses components of the slice collection agents in a new framework	OAI Openshift Kubernetes	Network slices monitoring
18	[59]	Bektas, Monhof, Kurtz, & Wietfeld,	2018	Proposes a management and orchestration controller for slice creation	NextEPC CommAgility-SmallCellSTACK - eNodeB	Service guarantees
19	[60]	Mao et al.	2020	Monitoring system and analyze completion times	Docker and Kubernetes	Performance analysis and container strategies
20	[61]	Saha, Beltre, Uminski, & Govindaraju	22018	Quantized MPI	Docker and Singularity	Performance analysis
21	[62]	Podolskiy, Mayo, Koay, Gerndt, & Patros	2019	Self-adaptive resource sharing	Kubernetes	Vertical container Expansion
22	[63]	R ankston, J Guo	2018	Use iperf3 for analysis	AWS	Containers and performance
23	[64]	P Wang, J Xu, M Ma, W Lin, D Pan	2018	CloudRanger	IBM Bluemix	Error detection
24	[65]	CM Aderaldo, NC	2019	Kubow	Kubernetes	Self-adaptive

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

		Mendonça, B Schmerl				
25	[66]	J Kosińska, K Zieliński	2020	AMoCNA	VM	Self-management
26	[67]	DBuchaca, J LL Berral, C Wang	2020	Use MLP to predict load	Not described	Workloads for ML

3. METHODOLOGY

3.1 Dataset Description

The dataset for the study was provided by Microsoft Azure Functions [https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsBlobDataset2020.md]. Data was gathered from 23rd of November 2020 to 6th December 2020 and was released in September, 2021. A telemetry system is deployed that records the activity every time some read or write operations performed on blob. According to architecture of Azure Functions, every tenant deploys his or her applications on Cloud. These applications are just collection of functions which are called on demand and perform specific operations. A function performs any operation which is required including blob read and write. Data in blob is stored in the form of object. This special telemetry system records the activity and properties of this activity. The features are listed below.

Total 1,048,576 instances were recorded over the period of two weeks and following statistics ascertained from data.

Timestamp: The time of invocation. This is an integer number that represents the milliseconds elapsed from 1970.

Region: Cloud may be access from anywhere and Cloud orchestration record the client location. This dataset has 19 different geographical regions.

User ID: Unique Id of user. Users are tenants of Cloud infrastructure. This dataset contains 152 unique users.

App Name: Application is collection of function. 238 applications found in dataset in given period of time

Function Invocation ID: Every time when a function invocated, a unique id generated.

Blob Name: This is the actual storage unit on Cloud which is access by invocated function. Dataset recorded on 236750 blobs were accessed in operation performed.

Table 3: Used Features Recorded Through the Telemetry System

Feature	Description
Timestamp	Mili Second Function access time since 1970
AnonRegion	ID of region
AnonUserId	ID of user
AnonAppName	Name of Application
AnonFunctionInvocationId	ID of invocation
AnonBlobName	Name of Storage Blob (Encrypted)
BlobType	Object Type of Blob
AnonBlobETag	Blob Version
BlobBytes	Number of Bytes (Read or Write)
Read	True or False (True if operation is Read)
Write	True of False (True if operation is Write)

Cloud data in the form of objects. These objects may be consisted on their actual file type such as ZIP or PNG. 18 types are mentioned in dataset.

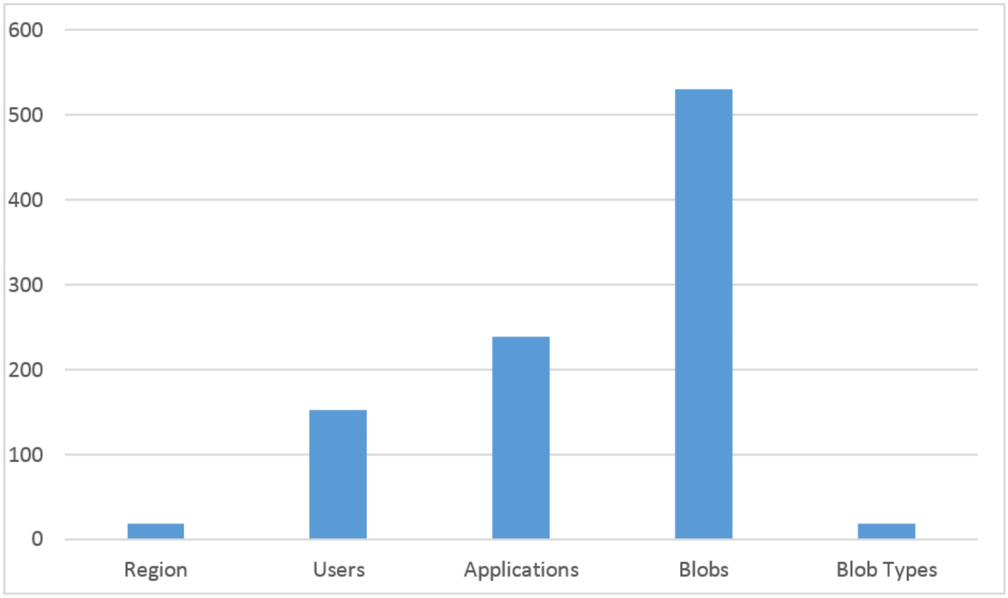

Figure 2: Histogram of used instances Blob Type

Version (E-Tag): Version of blob accessed.

Blob Bytes: Number bytes that read or write in a blob in given function invocation.

Read and Write: Name of the operation perform on blob. If this is “Read” operation then its value will be true and if this is “Write” operation then its Write value is true. According to dataset both operations are mutually exclusive.

ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

Table 4: Data Items Distribution

Data Field	Region	Users	Applications	Blobs	Blob Types
No of Instances	19	152	238	236,750	18

3.2 Data Preparation

Main goal of the research is to classify the method invocations in hot (high invocated in given interval of time) and cold (less invocated in similar time interval). For this purpose, we have to translate our data in such formation that provide an insight for time interval-based baskets. According to

[https://www.microsoft.com/en-us/research/uploads/prod/2020/05/serverless-ATC20.pdf] the basket size should be 20 minutes. The classifier model will examine the invocation pattern that includes its properties and invocation frequency per minute and define a particular function into hot or cold item. On the base class of a function, Cloud resource manager pre-warm the hot items load required resources into memory.

3.3 Data Translation

The current state of the dataset is defined in time intervals but we have timestamps to convert it into one minuet bins. After this operation we constructed a new dataset. To construct a required dataset, we develop a bin of 6 hours. Schema of newly constructed dataset as follows

Table 5: Schema of Newly Constructed Dataset

Field	Description
blob_name	Storage blob name
User_id	Unique Id of user
App_name	Function Name
Blob_type	Blob Type
1…360	Access frequency of each blob_name with per minute

The main focus of this dataset to produce access frequency table because this table elaborates the hotness of a data item.

3.4 New Features Description

Translated dataset explains the access frequency of data items but these data items still lot of preparation to get referable solution. For this purpose, we describe the dataset with different features of this access frequency table. A new dataset is generated with following features.

Table 6: Used Features for The Development Of New Dataset

Field	Description
Mean	Average of access frequency
Std	Standard deviation
Sem	Standard error mean
Kurtosis	Peak of curve in normal distribution
Skew	Shape of curve in normal distribution

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

As data description shows there are features which are coupled with normal distribution. Probability is the foundation of naïve bays classifier which will be base algorithm in machine learning. Three main functions are performed in our purposed hybrid solution which are given below.

3.5 Scheduling of components

Scheduling of components in the cloud involves managing the allocation, sequencing, and utilization of resources to optimize performance, scalability, and reliability. It encompasses techniques like load balancing, resource allocation, task prioritization, and fault tolerance to ensure efficient operation and utilization of the cloud infrastructure.

3.6 Over provisioning of resources

Over provisioning of resources refers to the practice of allocating more resources, such as CPU, memory, or storage, to a system or application than what is actually required for its optimal performance. It is often done as a precautionary measure to ensure that the system can handle peak workloads or to account for potential growth in demand.

However, over provisioning can lead to several negative consequences. Firstly, it results in wastage of resources, including increased costs for infrastructure and energy consumption. Secondly, over provisioning can lead to inefficient resource utilization, as the excess resources remain idle or underutilized for a significant portion of time. This inefficiency can limit scalability and hinder the ability to accommodate more workloads or users.

Over provisioning can also impact system performance. When resources are allocated beyond what is needed, it can lead to increased contention for shared resources, causing bottlenecks and reduced overall performance. Additionally, over provisioning can complicate resource management and capacity planning, as accurately determining the appropriate resource requirements becomes challenging.

To mitigate the negative effects of over provisioning, it is important to accurately assess the resource needs of the system or application. This can be done through monitoring and performance analysis to identify the actual resource utilization patterns. By understanding the workload characteristics and demand patterns, resources can be provisioned more efficiently, ensuring optimal performance while minimizing waste.

Cloud computing and virtualization technologies have provided more flexibility in resource provisioning, allowing for dynamic allocation and scaling based on actual demand. This helps in avoiding over provisioning by providing the ability to adjust resource allocations in real-time, matching the current workload requirements.

3.7 Memory wastage

The first source of waste is attributed to cloud tenants over dimensioning the memory resources allocated to their function sandboxes. Despite configuring sandboxes with larger memory sizes, the actual memory usage remains much lower. This is due to workload variations, where the same function code can be triggered with different arguments and input data, resulting in varying memory needs. The author presents evidence showing that memory usage can differ significantly depending on the input data and arguments, justifying the practice of over dimensioning resources. This trend is likely to continue and potentially worsen as FaaS platforms offer a wider range of memory configurations.

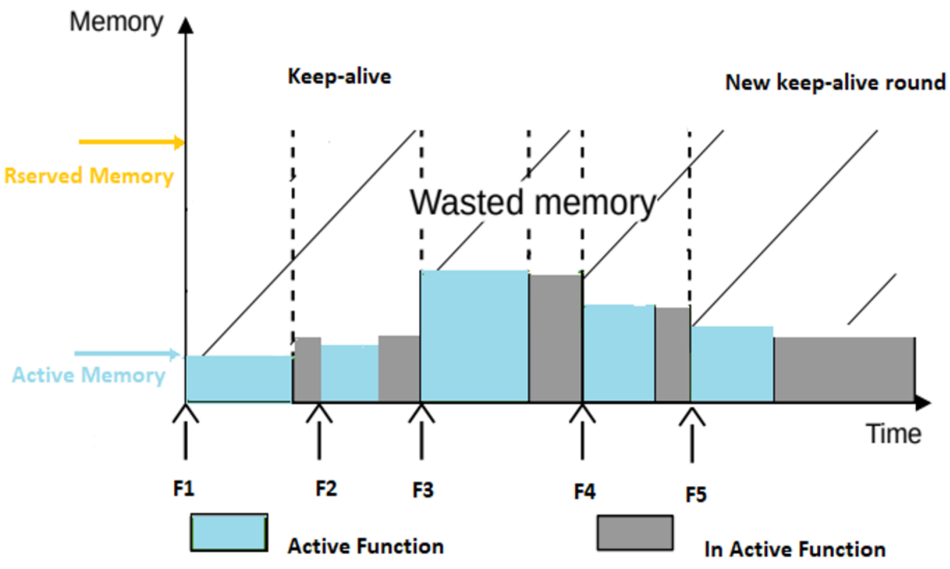

Figure 3: Memory configuration obtained using purposed model

The second cause of waste is the sandbox keep-alive policy. To mitigate the latency of cold starts, FaaS platforms keep sandboxes running for several minutes after handling the first event that triggered their startup. However, many applications are invoked infrequently, resulting in long periods of underutilized resources. The authors suggest that sandbox keep-alive is still essential in FaaS platforms, as optimizing cold starts is challenging and the cost of keeping applications warm can be high. They propose leveraging the physical memory waste resulting from sandbox keep-alive as a potential solution.

The text also mentions the use of machine learning for memory prediction. Predicting memory waste based solely on input data size or specific arguments is challenging due to the complex relationship between these factors and memory usage. Machine learning can effectively handle this complexity by considering uncharacterized function-specific arguments and input data features.

Additionally, the impact of utilizing a remote shared data store (RSDS) and the potential use of an in-memory object cache (IMOC) are discussed. The authors analyze the contribution of the ETL (Extract, Transform, and Load) phases for different functions and highlight the benefits and downsides of using an IMOC-based solution, such as Redis, to mitigate the latency of accessing the RSDS.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

Overall, the text addresses the issues of memory waste in FaaS platforms, discusses the causes behind it, and suggests potential solutions using machine learning and optimization strategies for keep-alive and resource allocation.

Flow chart

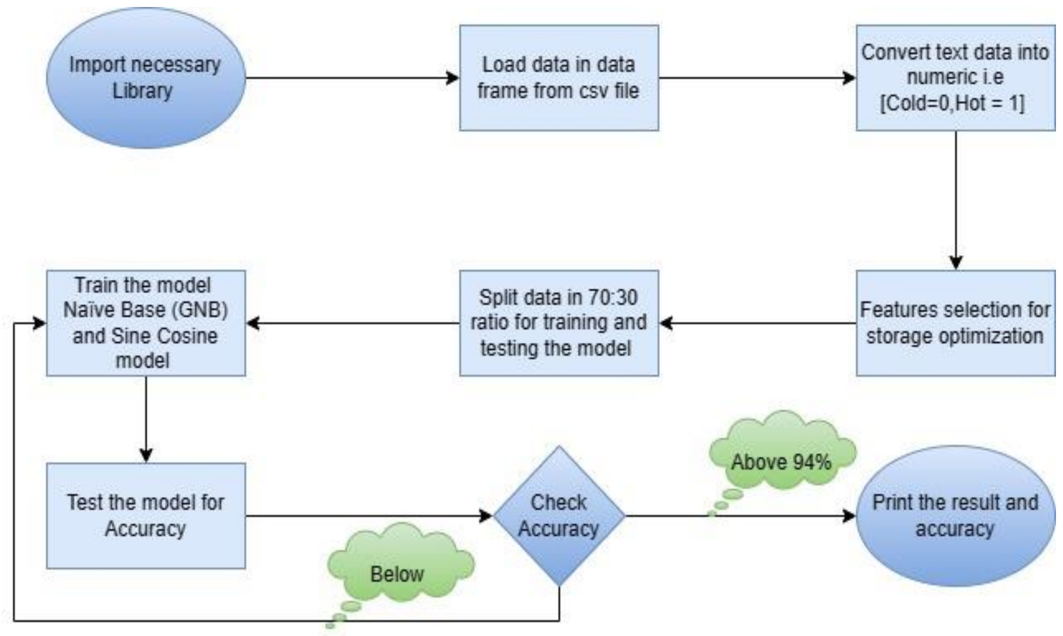
Import necessary Library
Load data in data frame from csv file
Convert text data into numeric i.e [Cold=0,Hot = 1]
Features selection for storage optimization
Split data in 70:30 ratio for training and testing the model
Train the model Naïve Base (GNB) and Sine Cosine model
Test the model for Accuracy
Check Accuracy
Above 94%
Below
Print the result and accuracy

Figure 4: Flow chart of our purposed model

3.8 Classification

Naïve Bayes Classifier is a probabilistic machine learning algorithm that is commonly used for classification tasks. It is based on the Bayes' theorem and makes the assumption of feature independence, which simplifies the calculations involved. Despite its simplicity, Naïve Bayes often performs well and can be quite effective in a variety of domains, especially when the independence assumption holds reasonably well.

The equation for Naïve Bayes Classifier can be derived from Bayes' theorem, which states:

$P(x) = \frac{P(y)P(y)}{P(x)}$**............1**

Where: P(y|x) is the posterior probability of class y given predictor x. P(x|y) is the likelihood of predictor x given class y. P(y) is the prior probability of class y. P(x) is the probability of predictor x.

Naïve Bayes assumes that all predictors are independent of each other given the class. With this assumption, the equation simplifies to:

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

$$P(x_1, x_2, x_3, \dots x_n) = \frac{p(y)P(y)P(y)\dots P(y)P(y)}{P(x_1,x_2,x_3,\dots x_n)} \dots\dots\dots\dots\dots \mathbf{2}$$

Where: x1, x2, xn represent the n predictors/features.

In other hand The Sine Cosine Algorithm (SCA) is an optimization algorithm used in cloud computing. It optimizes resource allocation by mimicking the oscillatory behavior of sine and cosine functions. By iteratively updating candidate solutions, it aims to find the optimal configuration that minimizes costs and maximizes resource utilization in the cloud system.

The Sine Cosine Algorithm (SCA) can be summarized as follows:

For each candidate solution Xi (i=1 to N, where N is the population size):

Initialize the position vector Xi randomly within the defined search space. Calculate the fitness value of Xi using the objective function. Update the velocity vector Vi and position vector Xi of the candidate solution as follows:

$$V_i = w\ x\ V_i + C_1\ x\ rand(0,1)x\ (P_i - X_i) + C_2\ x\ rand(0,1)x\ (G - X_i)$$

$$X_i = X_i + V_i$$

Where: Vi represents the velocity vector of candidate solution Xi. Pi represents the best position visited by Xi or the personal best. G represents the best position visited by any candidate solution in the population or the global best. The inertia weight. C1 and c2 are the acceleration coefficients. rand (0,1) generates a random number between 0 and 1. If the fitness value of the updated position Xi is better than its previous best position Pi, update Pi.

If the fitness value of Xi is better than the fitness value of G, update G. Repeat steps from calculate the fitness value until a termination criterion (e.g., maximum number of iterations or desired fitness value) is met. This algorithm work as iterative process of updating candidate solutions' velocity and position vectors based on personal and global best positions, allowing the Sine Cosine Algorithm to explore and converge towards an optimal solution in cloud computing optimization tasks.

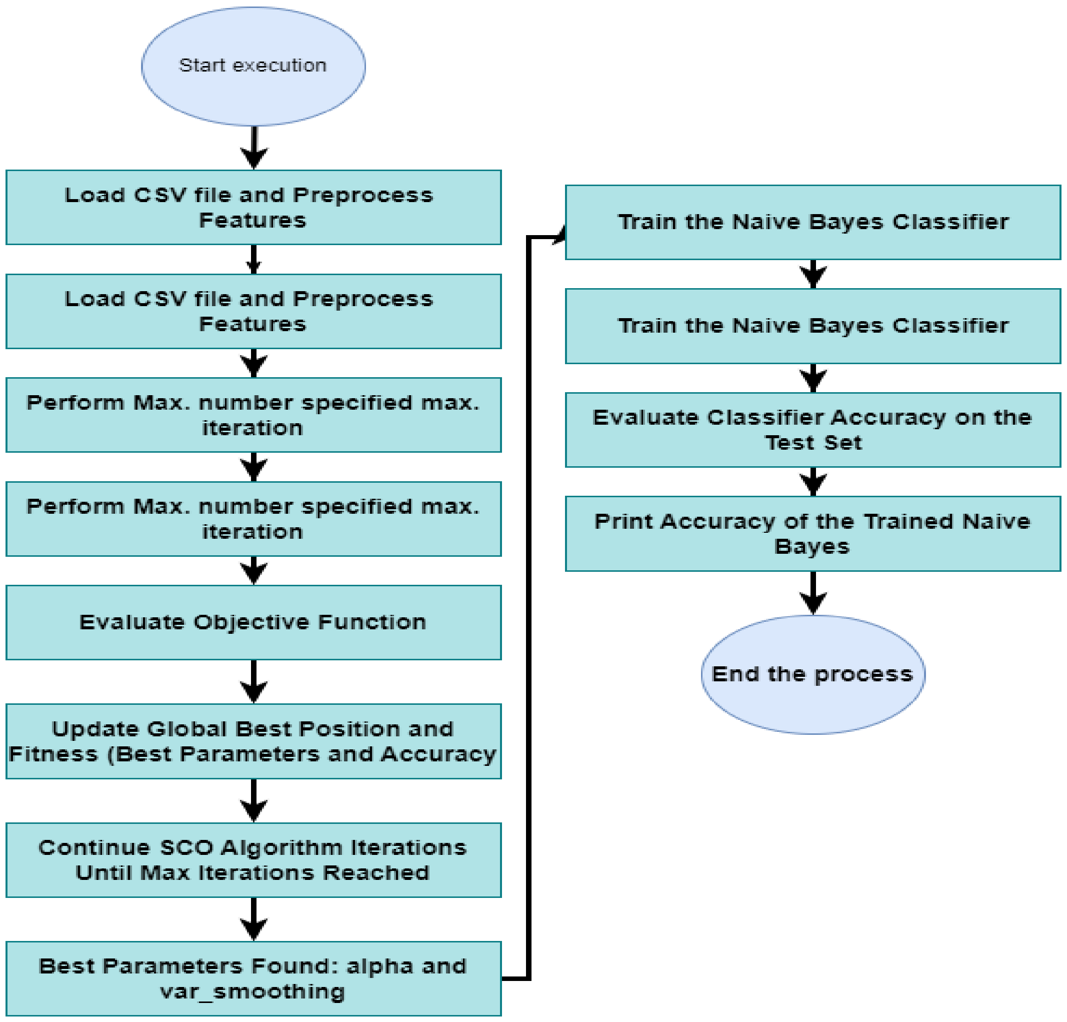
Start execution
Load CSV file and Preprocess Features
Load CSV file and Preprocess Features
Perform Max. number specified max. iteration
Perform Max. number specified max. iteration
Evaluate Objective Function
Update Global Best Position and Fitness (Best Parameters and Accuracy
Continue SCO Algorithm Iterations Until Max Iterations Reached
Best Parameters Found: alpha and var_smoothing
Train the Naive Bayes Classifier
Train the Naive Bayes Classifier
Evaluate Classifier Accuracy on the Test Set
Print Accuracy of the Trained Naive Bayes
End the process

Figure 5: Sine cosine optimization algorithm detailed workflow diagram

Now our data is fully prepared for classification. We use naïve Bayes classifier to bisect the items into hot and cold groups. Mean attributes of NB algorithm executions are following:

Although the NB is proper algorithm that provides models for classification but the data is too much diverse and fluctuate target classes then its accuracy only remain 57%. To increase the accuracy, we have to restructure the parameters. For this purpose, we employed Sine Cosine Optimization (SCO) to tune hyper parameters. The accuracy after using SCO we able to upgrade the accuracy to 94%.

4. RESULTS

For a detailed analysis of dataset, a separate application is written that provides the insights of data to generate classes for segregation of hot and cold data items. The application first, convert data into proper bins that we able to analyze the hot and cold

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

storage blobs. Blob temperature is depending on many features such as application, region, users, and blob type and invocation pattern.

To simplify the dataset, we squeeze less numbered feature names in to "Others". Following graph shows region wise distribution of blobs.

Figure 2: Region wise distribution of Blobs. Some Regions are very high number of blobs while some are very low

The graph shows the regions that are important attribute of data. A highly active region (i.e., *nd6* access more than 13 million blobs) sends more triggers than a silent region such as *gwm* only generate 2 calls.

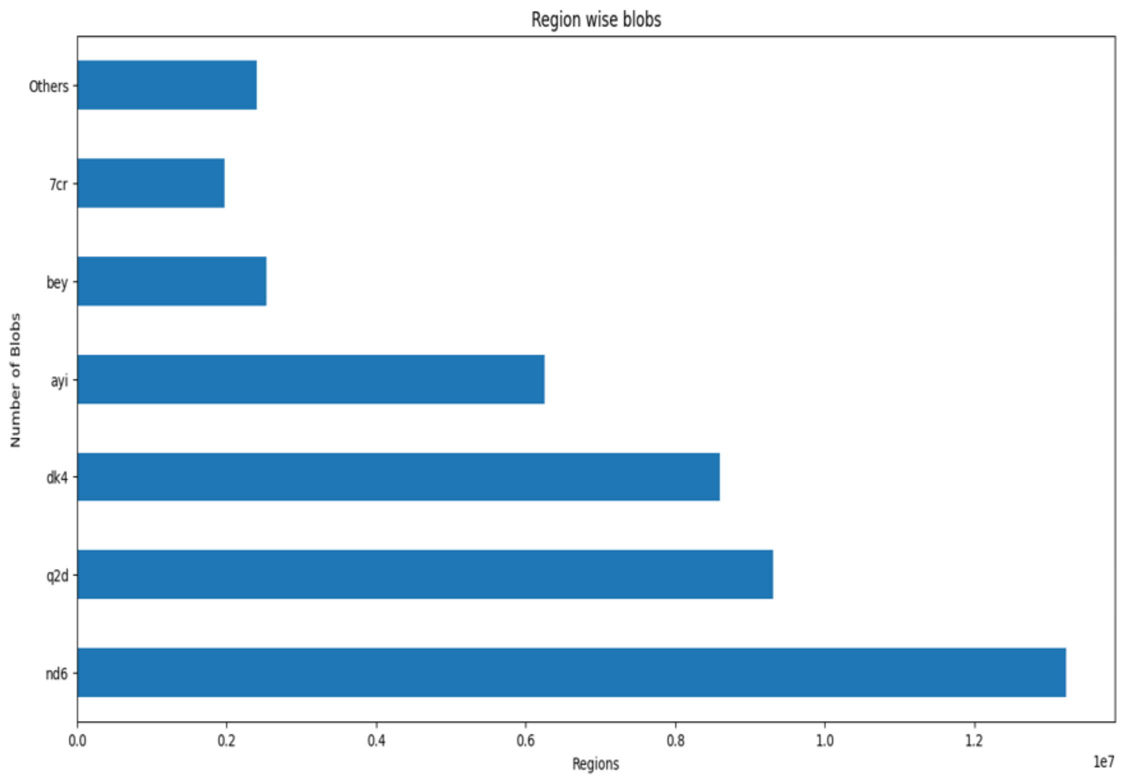

Figure 6: Region Wise Blobs

There are different kinds of blobs are in dataset. Some types are very often to call while others are rather rare to invoke. Such as *Block Blob/text/plain* contains about 30% of the total invokes. Due to this unevenness of call, the blob type plays a role in invocation frequency. System should be able to identify the highly active types and then map the blobs with these types to identify the hot blobs.

**Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954**

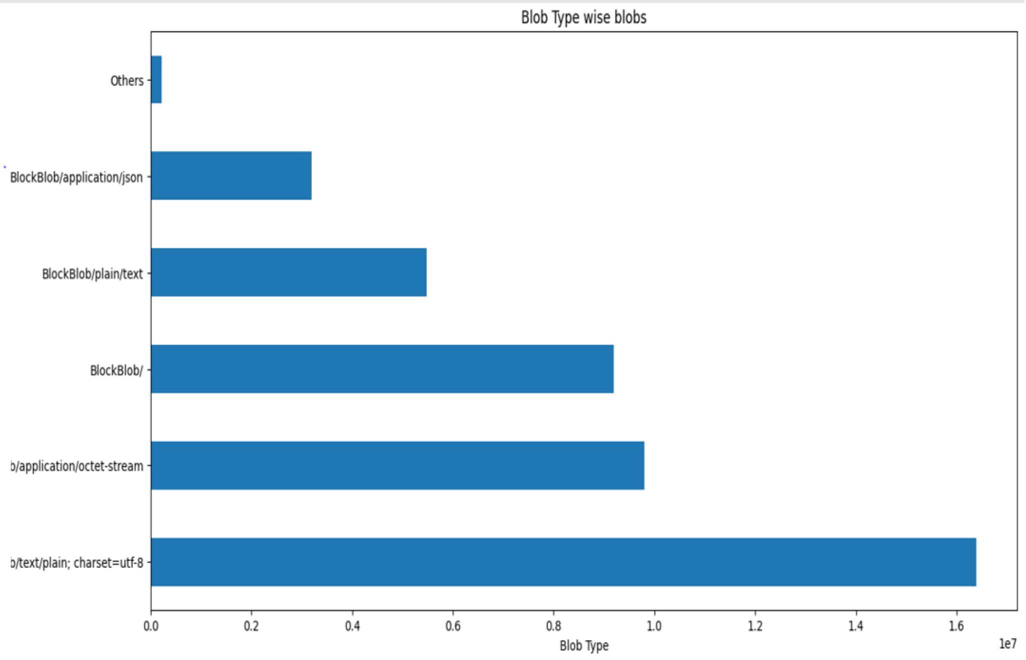

Figure 7: Type wise distribution of Blobs

Application is another significant feature because applications are the logical collections of function that is basic building block of FaaS. It is also clear that blobs distribution over the applications is also not even. We have total 238 applications. In a real time, scenario, the number of applications may increase to thousands. It is also important that highly active applications demand more active blobs and storage resources.

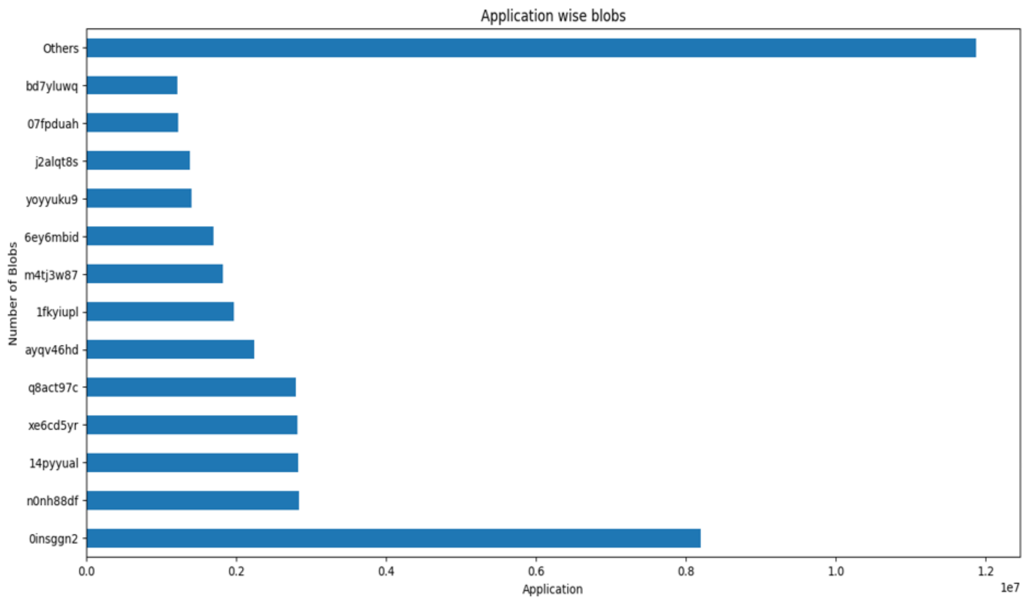

Figure 8: Application and their respective blobs

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

We can also see that the “Others” set is larger than significantly active applications.

The fluctuation of data also can be visualized from Users. These are the owners of application and like other features, these are also playing an important role in blobs invocation. A user can own multiple application which are the collections of function.

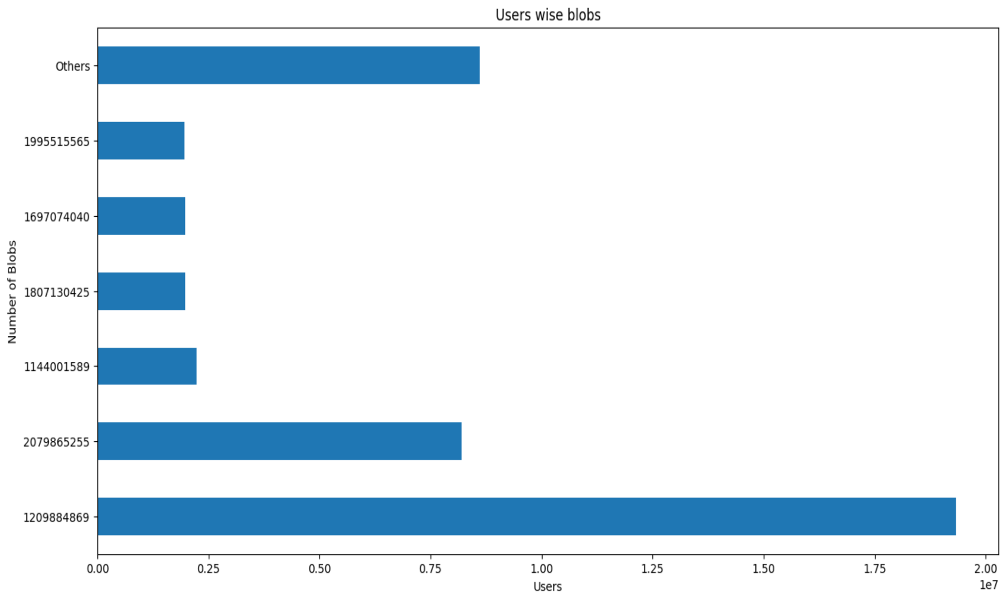

Figure 9: User wise Blobs distribution

If we take a look at blobs itself and definitions of the frequency of blobs calling, the following pattern may be produced (consider that blob names are encrypted). We have more 200k number of unique blobs in the dataset and it is not possible to show the frequency of all blobs but following top 20 blobs which are calling. Here we again witness the uneven numbers. Some blobs may be invoked more such first blob is called more than 140 million time.

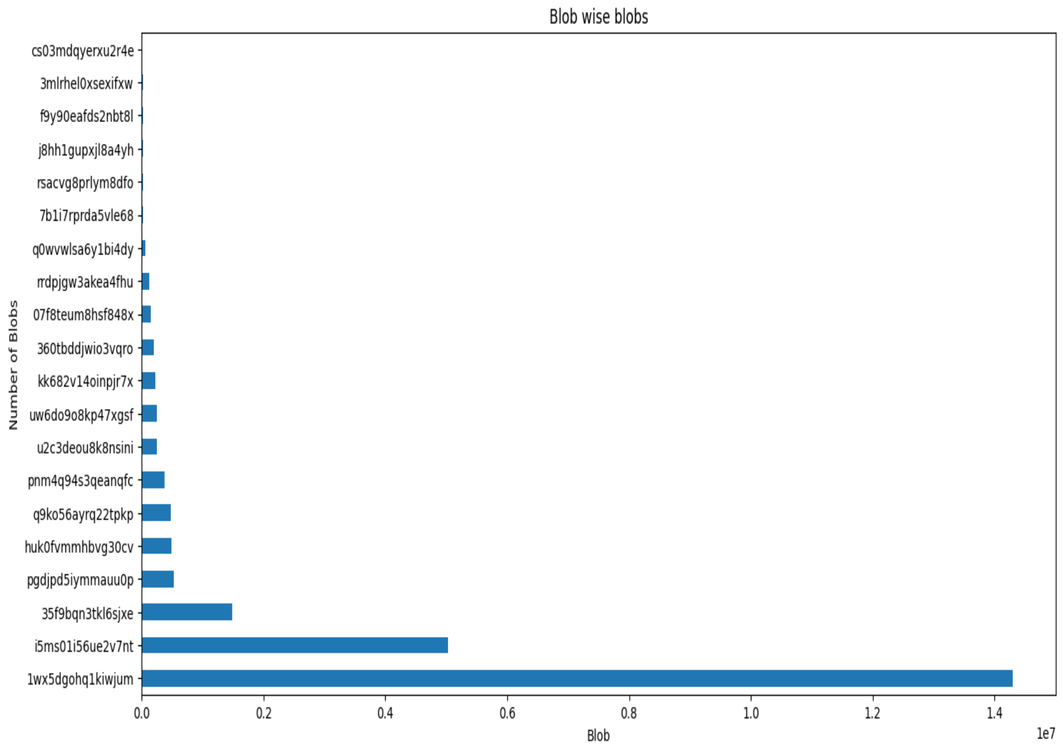

Figure 10: Top 20 blobs

The last blob on graph called more than 16k times and all other blobs which are omitted called lesser times than this. The above picture shows that blob frequency and its features are also important in the prediction of a blob redundancy.

Besides these attributes, we have also look at the time-based attributes of dataset because these attributes also define the temperature of blobs. Before time series analysis, we convert the dataset in bins of equal amount of time span. This procedure generated an insightful information for understanding about the occurrence of a blob event. For each bin we calculate the description of top 20 blobs. All descriptive features of a bin are not required but some important features and their insights are following. Due to large number of blobs involved, we will present only top performers in following displays.

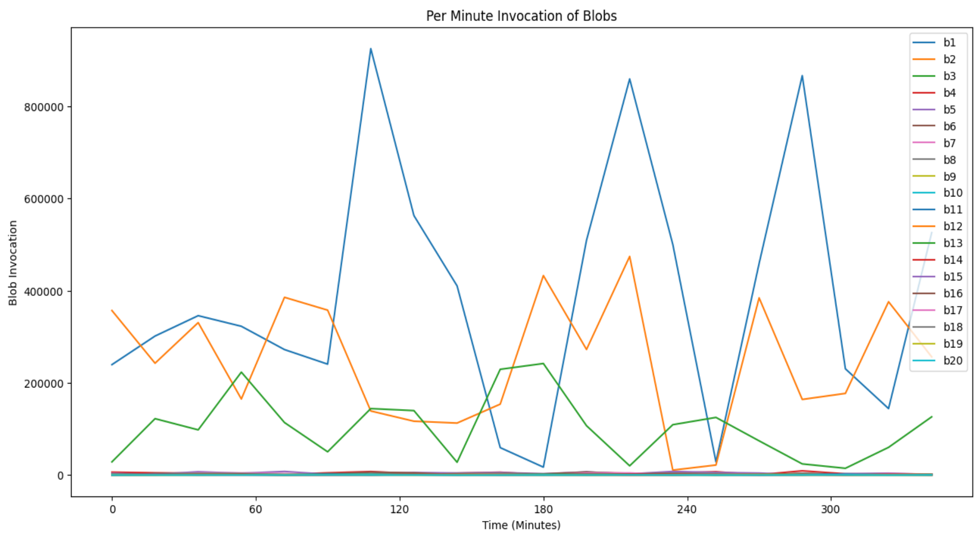

Figure 11: Blobs Invocation per Minute. Data is showing 6 hours span

After analysis of 236,750 blobs, we define them into the 3 categories of hot (very frequent), warm (less frequent) and cold (rare). For testing purpose, we classify 16351 blobs as hot, 48668 blobs as warm and 171731 as cold. To testify the integrity of results, we execute the model six times with simple naïve Bayes and then with sine cosine optimization. Every time we increase the train set up to 10% to check out the validity of results. We find that as training set increased, the accuracy also increased but the ratio of increment in accuracy doesn't significant as train set size significantly increased (10% each run) such as training set size was 20% of total data, the accuracy was about 56.51% and when training set size was much larger up to 60% of total data the accuracy increases on 63.88%. As we add support of Sine-Cosine Optimization, the accuracy increased significantly. It was 78.50% on 20 percent of training data while approaches to 93.41% with 60% of training set. It is also observed that the accuracy remained steady after maintaining the training set size to 40%.

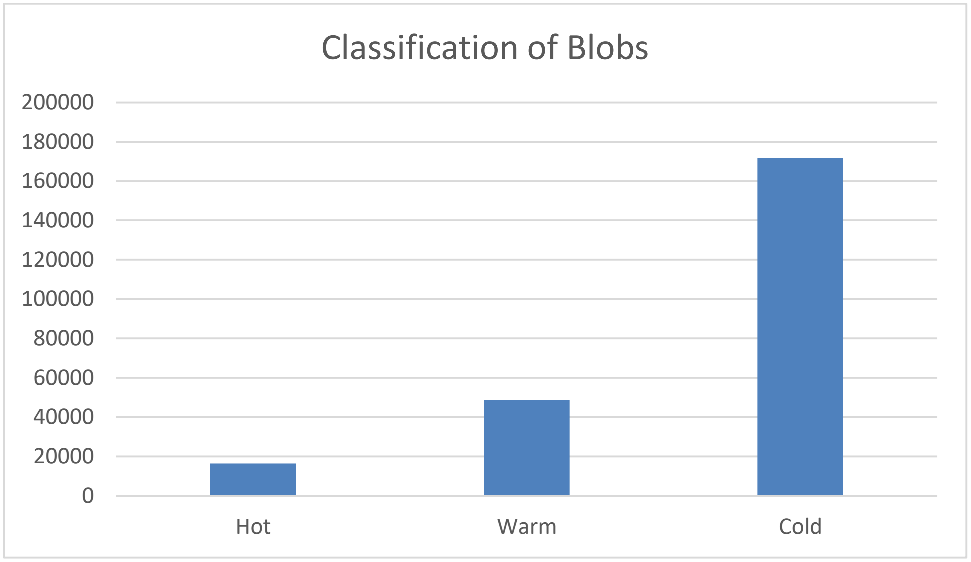

Figure 12: Distribution of Blobs into Classes

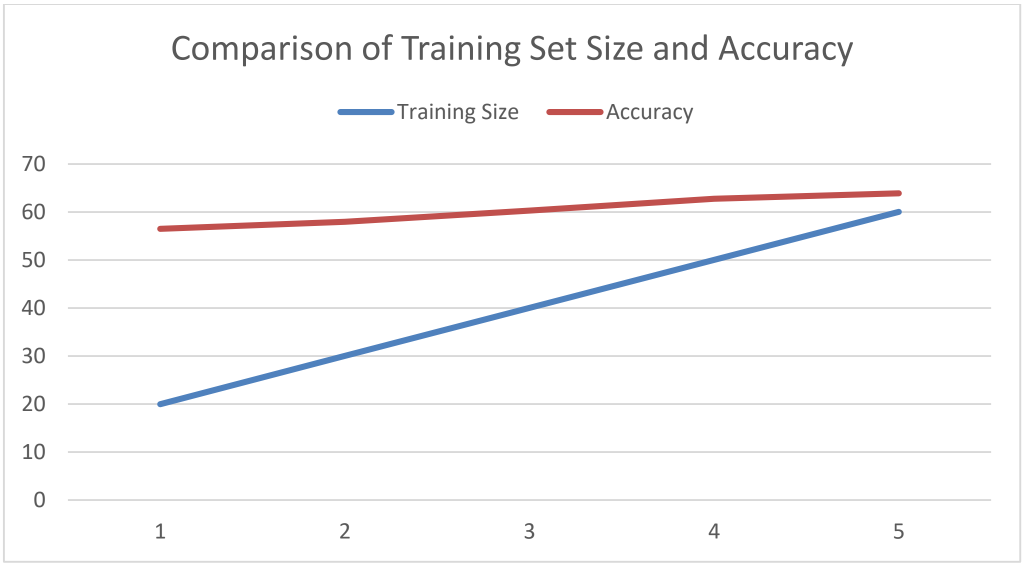

Figure 13: Comparison of Training Size and Accuracy (Simple Naïve Bayes). We can see that accuracy is not significantly increase as training set increased

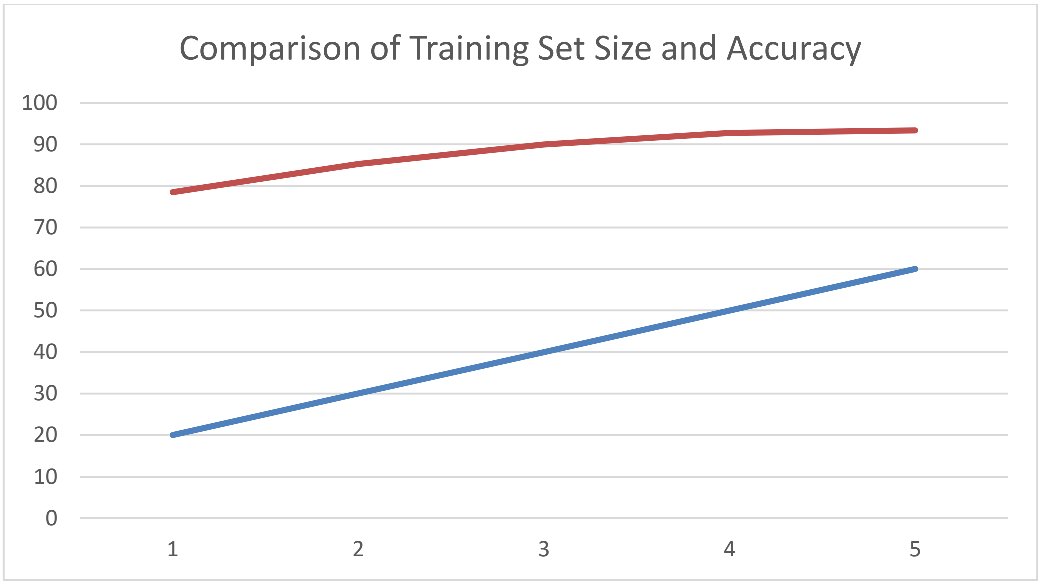

Figure 14: Comparison of Training Set Size and Accuracy (Naive Bayes optimized with SCO). Accuracy significantly increases in even small size of training set

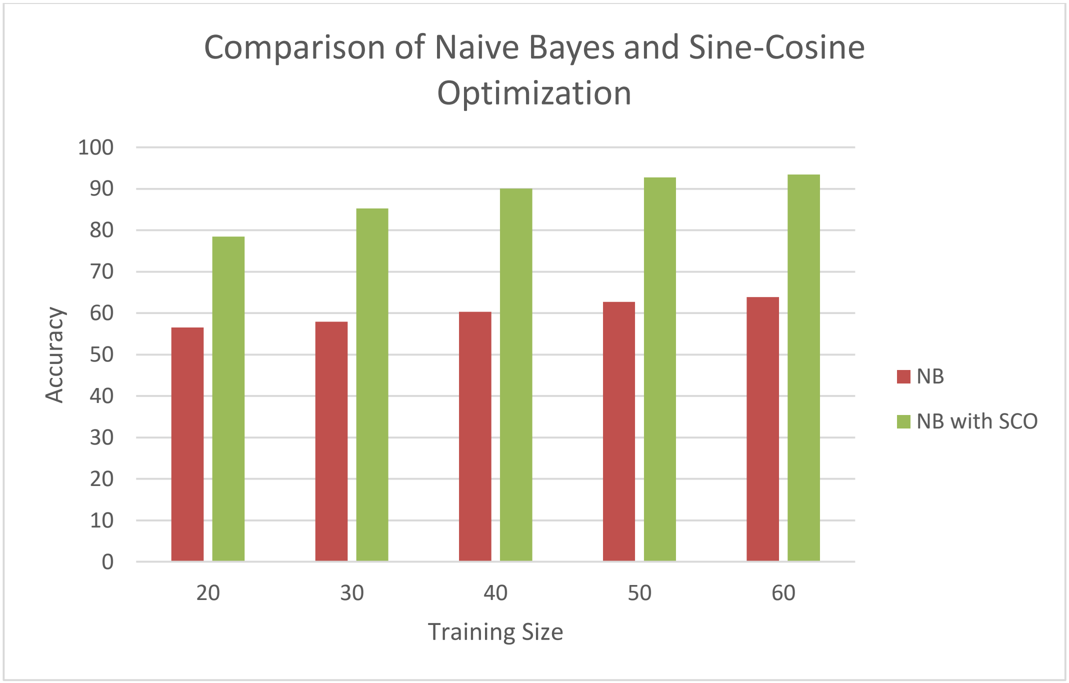

Figure 15: Comparison of Simple NB and NB Supported with SCO. NB Supported with SCO is significantly high

5. THREATS AND VALIDITY

Cloud storage management offers many advantages, such as scalability, accessibility, and cost-effectiveness. However, it also comes with its share of threats and challenges. Here are some common threats and validity concerns associated with cloud storage management.

Cloud provider provide APIs (Application Programming Interfaces) to integration with other applications. If these APIs are not secured, they can be exploited by attackers to gain unauthorized access to cloud resources. cloud service provider have backup and disaster measures, the data loss can be occur due to various reasons which may be hardware failures, software, or human errors. I ensue that proper data backup and recovery strategies are in place.

Cloud storage systems may be targeted by cybercriminals attempting to gain unauthorized access to sensitive data. Data breaches can lead to the exposure of personal information, financial data, or other confidential data, resulting in severe consequences for both individuals and organizations. Unauthorized access to the cloud system can lead to data theft, manipulation, or other malicious activities. Implementing strong access controls and monitoring systems can help mitigate this risk. When using cloud storage, organizations are reliant on their cloud service provider to manage and secure their data.

Cloud storage services may experience downtime due to maintenance, technical issues, or other reasons. Organizations relying heavily on cloud services may face disruptions in their operations during these periods. Lock-In: Switching cloud service providers can be challenging and costly, leading to concerns about vendor lock-in. Organizations need to carefully consider their options before committing to a particular provider.

5.1 Validity Concerns:

The cloud storage management has become widely used, some validity concerns persist, particularly for certain use cases:

Cloud storage requires a stable internet connection to access and manage data. In areas with limited or unreliable internet connectivity, cloud storage may not be a viable option. Some users worry about the privacy of their data and whether cloud service providers may use or analyze their data for purposes beyond storage and management.

Transferring large volumes of data to and from the cloud can be time-consuming, particularly for users with limited bandwidth while cloud storage can be cost-effective, managing costs can be challenging, as pricing structures can be complex, and unexpected usage spikes may result in higher bills.

Addressing these threats and validity concerns requires a proactive approach, including rigorous security measures, data encryption, access controls, thorough due diligence when selecting a cloud provider, and clear service level agreements (SLAs) to ensure the provider meets the organization's requirements for data protection, availability, and performance.

6. CONCLUSION

The above presentation unleashed the support of SCO to uplift the accuracy of fundamental naïve bayes classifier. As we discussed the accuracy, we can conclude that our methodology can correctly identify the candidate of keep-alive policy. As the accuracy increases, the probability of correct blobs also increases that remain in memory for next use and consequently, system has reduced the round trips of blob loading every time when they required.

In conclusion, we have introduced and demonstrated the effectiveness of the Sine cosine algorithm in significantly improving the performance of diverse FaaS workloads in a cost-effective manner. Our approach can be seamlessly integrated into existing cloud infrastructures, including FaaS platforms and object storage services, with minimal modifications. It ensures full transparency for application-level code and eliminates the need for explicit booking or provisioning of additional storage resources. By addressing the challenges of scheduling and execution latency, the Sine cosine algorithm provides an efficient and practical solution for enhancing the overall performance of cloud-native applications.

Although the NB is proper algorithm that provides models for classification yet the data is too much diverse and fluctuate target classes then its accuracy only remain 57%. To increase the accuracy, we have to restructure the parameters. For this purpose, we employed Sine Cosine Optimization (SCO) to tune hyper parameters. The accuracy after using SCO we able to upgrade the accuracy to 94%.

6.1 Limitations and Future Work

Although we tried to find the best method to generate effective policy for FaaS but it has certain limitations that may reduce the impression. First, the dataset we use, was taken from Windows Azure Function. There are lot of other vendors and infrastructures that may have different properties and these properties may affect the performance of our system. In next work we will evaluate this methodology on other available datasets.

References

[1] Amazon. (12 July 2020). *AWS Cloudtrail.* Available: https://aws.amazon.com/cloudtrail/?nc1=h_ls

[2] Azure. (12/05/2023 *Functions:* . Available: https://azure.microsoft.com/en-us/services/functions/

[3] R. Hat. (10/5/2023). *Red Hat OpenShift:* . Available: https://www.redhat.com/en/technologies/cloud-computing/openshift

[4] M. Kiener, M. Chadha, and M. Gerndt, "Towards demystifying intra-function parallelism). *Azure* in serverless computing," in *Proceedings of the Seventh International Workshop on Serverless Computing (WoSC7) 2021*, 2021, pp. 42-49.

[5] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing—The business perspective," *Decision support systems,* vol. 51, no. 1, pp. 176-189, 2011.

[6] G. Adzic and R. Chatley, "Serverless computing: economic and architectural impact," in *Proceedings of the 2017 11th joint meeting on foundations of software engineering*, 2017, pp. 884-889.

[7] S. Horovitz, R. Amos, O. Baruch, T. Cohen, T. Oyar, and A. Deri, "Faastest-machine learning based cost and performance faas optimization," in *Economics of Grids, Clouds, Systems, and Services: 15th International Conference, GECON 2018, Pisa, Italy, September 18–20, 2018, Proceedings 15*, 2019, pp. 171-186: Springer.

[8] K. C. Ayyagari. (10, May). *Functions, events, triggers oh my! How to build event-driven app*. Available: https://cloud.google.com/blog/products/serverless/learn-about-cloud-functions-events-and-triggers

[9] A. Wang *et al.*, "Faasnet: Scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021.

[10] S. Lehrig, H. Eikerling, and S. Becker, "Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics," in *Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures*, 2015, pp. 83-92.

[11] S. Malla and K. Christensen, "HPC in the cloud: Performance comparison of function as a service (FaaS) vs infrastructure as a service (IaaS)," *Internet Technology Letters,* vol. 3, no. 1, p. e137, 2020.

[12] J. Manner, M. Endreß, T. Heckel, and G. Wirtz, "Cold start influencing factors in function as a service," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 2018, pp. 181-188: IEEE.

[13] D. S. Hirolikar, G. Ezhilarasan, K. Sharma, B. S. Alfurhood, D. Gangodkar, and A. Kudale, "Smart Electricity Billing Management System Using Artificial Intelligent Based for the Implementation of Pre and Post Paid Tariffs," in *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*, 2023, pp. 1114-1119: IEEE.

[14] A. Raza, Z. Zhang, N. Akhtar, V. Isahagian, and I. Matta, "LIBRA: An economical hybrid approach for cloud applications with strict SLAs," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 136-146: IEEE.

[15] Y. Verginadis *et al.*, "Prestocloud: a novel framework for data-intensive multi-cloud, fog, and edge function-as-a-service applications," *Information Resources Management Journal (IRMJ),* vol. 34, no. 1, pp. 66-85, 2021.

[16] H. Zhao, Z. Benomar, T. Pfandzelter, and N. Georgantas, "Supporting Multi-Cloud in Serverless Computing," *arXiv preprint arXiv:2209.09367,* 2022.

[17] V. Sreekanti, C. Wu, S. Chhatrapati, J. E. Gonzalez, J. M. Hellerstein, and J. M. Faleiro, "A fault-tolerance shim for serverless computing," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1-15.

[18] Z. Daher and H. Hajjdiab, "Cloud storage comparative analysis amazon simple storage vs. Microsoft azure blob storage," *International Journal of Machine Learning and Computing,* vol. 8, no. 1, pp. 85-9, 2018.

[19] Microsoft. *Introduction to Azure Blob Storage*. Available: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

[20] B. Rochwerger *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development,* vol. 53, no. 4, pp. 4: 1-4: 11, 2009.

[21] M. Taghipour, M. Soofi, M. Mahboobi, and J. Abdi, "Application of cloud computing in system management in order to control the process," *Management,* vol. 3, no. 3, pp. 34-55, 2020.

[22] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing,* vol. 23, no. 4, pp. 2399-2424, 2020.

[23] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, R. M. Parizi, and K.-K. R. Choo, "Fog data analytics: A taxonomy and process model," *Journal of Network and Computer Applications,* vol. 128, pp. 90-104, 2019.

[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience,* vol. 41, no. 1, pp. 23-50, 2011.

[25] W. Chen, B. Liu, I. Paik, Z. Li, and Z. Zheng, "QoS-aware data placement for MapReduce applications in geo-distributed data centers," *IEEE Transactions on Engineering Management,* vol. 68, no. 1, pp. 120-136, 2020.

[26] H. Liu, S. Liu, and K. Zheng, "A reinforcement learning-based resource allocation scheme for cloud robotics," *IEEE Access,* vol. 6, pp. 17215-17222, 2018.

[27] L. Mercl and J. Pavlik, "Public cloud kubernetes storage performance analysis," in *Computational Collective Intelligence: 11th International Conference, ICCCI 2019, Hendaye, France, September 4–6, 2019, Proceedings, Part II 11*, 2019, pp. 649-660: Springer.

[28] Z. Yu, J. Hu, G. Min, Z. Wang, W. Miao, and S. Li, "Privacy-preserving federated deep learning for cooperative hierarchical caching in fog computing," *IEEE Internet of Things Journal,* vol. 9, no. 22, pp. 22246-22255, 2021.

[29] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "PDLM: Privacy-preserving deep learning model on cloud with multiple keys," *IEEE Transactions on Services Computing,* vol. 14, no. 4, pp. 1251-1263, 2018.

[30] R. Li *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access,* vol. 6, pp. 74429-74441, 2018.

[31] P. S. Rawat, P. Dimri, P. Gupta, and G. P. Saroha, "Resource provisioning in scalable cloud using bio-inspired artificial neural network model," *Applied Soft Computing,* vol. 99, p. 106876, 2021.

[32] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Computing,* vol. 4, no. 6, pp. 60-69, 2017.

[33] P. Sonia and R. Malika, "A hybrid cloud security model for securing data on cloud," in *Proceedings of the Workshop on Computer Networks and Communications, Chennai, India*, 2021, vol. 1.

[34] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet of Things Journal,* vol. 7, no. 4, pp. 3415-3426, 2020.

[35] A. Erradi and Y. Mansouri, "Online cost optimization algorithms for tiered cloud storage services," *Journal of Systems and Software,* vol. 160, p. 110457, 2020.

[36] Y. Mansouri and R. Buyya, "Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers," *Journal of Parallel and Distributed Computing,* vol. 126, pp. 121-133, 2019.

[37] H.-E. Chihoub, S. Ibrahim, Y. Li, G. Antoniu, M. S. Perez, and L. Bougé, "Exploring energy-consistency trade-offs in cassandra cloud storage system," in *2015 27th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2015, pp. 146-153: IEEE.

[38] A. Alzahrani, T. Alyas, K. Alissa, Q. Abbas, Y. Alsaawy, and N. Tabassum, "Hybrid Approach for Improving the Performance of Data Reliability in Cloud Storage Management," *Sensors,* vol. 22, no. 16, p. 5966, 2022.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

[39] S. Ristov and P. Gritsch, "FaaSt: Optimize makespan of serverless workflows in federated commercial FaaS," in *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, 2022, pp. 183-194: IEEE.

[40] M. Copik, A. Calotoiu, R. Bruno, R. Böhringer, and T. Hoefler, "Process-as-a-Service: FaaSt Stateful Computing with Optimized Data Planes," ed: Accessed, 2022.

[41] R. S. PATIL, A. KOTWAL, and S. S. Patil, "Efficient Iot-Based Cloud Computing Framework For Secure Data Storage Using Machine Learning Algorithm," *Journal Of Theoretical And Applied Information Technology,* Vol. 101, No. 10, 2023.

[42] Z. Xiang, M. Höweler, D. You, M. Reisslein, F. H. J. I. T. o. N. Fitzek, and S. Management, "X-MAN: A non-intrusive power manager for energy-adaptive cloud-native network functions," vol. 19, no. 2, pp. 1017-1035, 2021.

[43] J. Kim, J. Lee, T. Kim, and S. Pack, "Deep reinforcement learning based cloud-native network function placement in private 5g networks," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1-6: IEEE.

[44] S. D. A. Shah, M. A. Gregory, and S. J. I. A. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," vol. 9, pp. 10903-10924, 2021.

[45] Q. Wu, R. Wang, X. Yan, C. Wu, and R. J. I. W. C. Lu, "Intrinsic Security: A Robust Framework for Cloud-Native Network Slicing via a Proactive Defense Paradigm," vol. 29, no. 2, pp. 146-153, 2022.

[46] S. Imadali and A. Bousselmi, "Cloud native 5g virtual network functions: Design principles and use cases," in *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*, 2018, pp. 91-96: IEEE.

[47] J. Kim, J. Lee, T. Kim, S. J. I. T. o. N. Pack, and S. Management, "Deep Q-Network-based Cloud-Native Network Function Placement in Edge Cloud-Enabled Non-Public Networks," 2022.

[48] W. Qiang, W. Chunming, Y. Xincheng, C. J. I. T. o. N. Qiumei, and S. Management, "Intrinsic security and self-adaptive cooperative protection enabling cloud native network slicing," vol. 18, no. 2, pp. 1287-1304, 2021.

[49] J.-B. Lee, T.-H. Yoo, E.-H. Lee, B.-H. Hwang, S.-W. Ahn, and C.-H. J. I. A. Cho, "High-performance software load balancer for cloud-native architecture," vol. 9, pp. 123704-123716, 2021.

[50] S. Sharma, R. Miller, and A. J. I. C. M. Francini, "A cloud-native approach to 5G network slicing," vol. 55, no. 8, pp. 120-127, 2017.

[51] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2018, pp. 2177-2185: IEEE.

[52] R. Schmidt, N. J. I. T. o. N. Nikaein, and S. Management, "RAN engine: Service-oriented RAN through containerized micro-services," vol. 18, no. 1, pp. 469-481, 2021.

[53] A. Mudvari, N. Makris, and L. Tassiulas, "ML-driven scaling of 5G Cloud-Native RANs," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1-6: IEEE.

[54] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, S. J. I. C. S. Guo, and Tutorials, "A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory," vol. 24, no. 2, pp. 1175-1211, 2022.

[55] A. Boudi, M. Bagaa, P. Pöyhönen, T. Taleb, and H. J. I. N. Flinck, "AI-based resource management in beyond 5G cloud native environment," vol. 35, no. 2, pp. 128-135, 2021.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 10:2023
DOI: 10.5281/zenodo.10005954

[56] R. Bolla *et al.*, "From cloud-native to 5g-ready vertical applications: An industry 4.0 use case," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1-6: IEEE.

[57] K. Abbas, T. A. Khan, M. Afaq, and W.-C. J. I. A. Song, "Network slice lifecycle management for 5g mobile networks: An intent-based networking approach," vol. 9, pp. 80128-80146, 2021.

[58] M. Mekki, S. Arora, A. J. I. T. o. N. Ksentini, and S. Management, "A scalable monitoring framework for network slicing in 5g and beyond mobile networks," vol. 19, no. 1, pp. 413-423, 2021.

[59] C. Bektas, S. Monhof, F. Kurtz, and C. Wietfeld, "Towards 5G: An empirical evaluation of software-defined end-to-end network slicing," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1-6: IEEE.

[60] Y. Mao, Y. Fu, S. Gu, S. Vhaduri, L. Cheng, and Q. J. a. p. a. Liu, "Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes," 2020.

[61] P. Saha, A. Beltre, P. Uminski, and M. Govindaraju, "Evaluation of docker containers for scientific workloads in the cloud," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018, pp. 1-8.

[62] V. Podolskiy, M. Mayo, A. Koay, M. Gerndt, and P. Patros, "Maintaining SLOs of cloud-native applications via self-adaptive resource sharing," in *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2019, pp. 72-81: IEEE.

[63] R. Bankston and J. Guo, "Performance of container network technologies in cloud environments," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 0277-0283: IEEE.

[64] P. Wang *et al.*, "Cloudranger: Root cause identification for cloud native systems," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2018, pp. 492-502: IEEE.

[65] C. M. Aderaldo, N. C. Mendonça, B. Schmerl, and D. Garlan, "Kubow: An architecture-based self-adaptation service for cloud native applications," in *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, 2019, pp. 42-45.

[66] J. Kosińska and K. J. J. o. G. C. Zieliński, "Autonomic management framework for cloud-native applications," vol. 18, pp. 779-796, 2020.

[67] D. Buchaca, J. L. Berral, C. Wang, and A. Youssef, "Proactive container auto-scaling for cloud native machine learning services," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 475-479: IEEE.