CSIT6910A - Independent Project

The Final Report of Project

Research on Intercloud Brokerage Service for Sky Computing

Student: XUE Hongbin

Student ID: 20922852

Supervisor: Professor WANG Wei

Date: 2023/08/08

TABLE OF CONTENT

1. Acknowledgment

Firstly, I would like to express my deepest gratitude to my project supervisor Professor Wang Wei for giving me this opportunity to carry out this independent project and for his continuous support and patient guidance throughout the project. Under his direction, I have gained a deeper understanding of cloud computing and sky computing, which will benefit me for a lifetime.

Secondly, I would like to thank Professor Wang Wei's doctoral student Yang Lingyun for his assistance and constructive feedback on this independent project. Under his guidance, I confirmed my experimental direction and significantly accelerated the progress of my experiment.

Finally, I would like to thank my parents, who suffered from cancer last year and this year respectively. I should have taken more care of them and accompanied them, but they have kept me focused on my studies and always supported me.

Your support and guidance have played an essential role in completing this project, and I am deeply grateful.

2. Abstract

The emergence of cloud computing has completely changed how we store, process, and access data. However, as the field of cloud computing becomes increasingly diverse and complex, our demand for more flexible and efficient cloud services is also growing. For example, to comply with data processing and placement regulations in different regions and countries, and to avoid the dire situation of cloud service provider downtime, more and more users hope to migrate workloads between different clouds easily. This has led to the emergence of Sky Computing, which can be seen as the next stage of cloud computing, emphasizing the development and compatibility of multi-cloud applications, allowing applications to run on any cloud provider.

To implement Sky Computing, researchers have proposed the concept of intercloud brokerage services, which play a crucial role in implementing Sky Computing. It can automatically select the most suitable cloud service provider based on user needs and preferences, such as cost, performance, and execution location.

3. Introduction

With the development of computer science, cloud computing has become an indispensable technology in our daily lives, especially during the epidemic period - schools carry out online teaching on various cloud platforms, enterprises carry out video conferences on cloud platforms and engage in online collaborative work whether online teaching or work, stable cloud services are needed to ensure stability.

Cloud computing is also one of my favorite computer fields. I learned about cloud computing when I was an undergraduate. Still, for a long time, my use of cloud computing has been limited to virtual machines (Infrastructure as a Service) and Object storage services (OSS) provided by major cloud service providers. And building soft routing, and so on. My understanding of cloud computing is very shallow, and I am more focused on its practice. Therefore, when I first took Professor Wang Wei's course, I thought it would be the same teaching method — constantly introducing the concepts and definitions of cloud computing. But what caught my eye was that Professor Wang first introduced interesting concepts such as data centers and servers to cloud computing, and also mentioned the pricing models of cloud service providers. At the end of the semester, Professor Wang also mentioned Sky Computing. I am very interested in all of these.

Therefore, when Professor Wang Wei assigned me the research direction of an independent project Sky Computing, I was very excited and looking forward to it. Although this is my first exposure to scientific research, and I encountered many difficulties, I still enjoy the scientific research process very much. And as I gain a deeper understanding of Sky Computing, I am increasingly confident that it will eventually become popular.

This project first explored Sky Computing in-depth, focusing on its core concepts, technical requirements, and future trends. Then, it will focus on studying intercloud brokerage services' working principles and implementation details and replicate some implemented experiments. Through the replication of relevant experiments, we will gain a deeper understanding of the core concepts of Sky Computing and the working principles of cloud brokerage services. Finally, we will explore how cloud brokerage services can automatically route and optimize workloads among multiple cloud providers to achieve optimal performance and cost-effectiveness and the importance of cloud economy services in the Sky Computing field.

4. Literature review

Sky Computing is a very cutting-edge research direction. Although researchers such as Katarzyna Keakey have published relevant papers as early as 2009, the concept did not immediately achieve widespread application and development due to technological and market reasons. This concept did not receive renewed attention until recent years, so we will focus on studying the three most representative papers in recent years. At the same time, I would like to express my gratitude to Professor Wang Wei for providing me with the necessary information, which enabled me to launch my independent project quickly and helped me determine the direction of my experiment. It also gave me my own thoughts on the field of Sky Computing.

4.1 From Cloud Computing to Sky Computing

This paper is one of the most essential papers in Sky Computing in recent years, as it proposes a concept that has not been mentioned or even seen in a long time but is incredibly innovative - Sky Computing. This paper has vigorously promoted the development of Sky Computing, and without exaggeration, I believe it has opened a new era.

The paper proposes that we are about to transition from the era of cloud computing to Sky Computing. It clarifies the concept of Sky Computing: Sky Computing is the next stage of cloud computing, aiming to achieve multi-cloud application development and compatibility, enabling applications to run flexibly among different cloud providers. And its goal: to allow applications to have the ability to write once, run anywhere, which means they can be written once on any cloud provider and then run on multiple cloud platforms.

To implement Sky Computing, this paper proposes a three-level architecture based on the design architecture of the Internet: Compatibility Layer, Intercloud Layer, and Peering Between Clouds. The first is the Compatibility Layer. In my opinion, it is an abstraction layer that can abstract the services provided by cloud service providers, enabling applications to migrate seamlessly on different cloud platforms. For example, we can easily move our workload from Azure to AWS or Google Cloud. I think the compatibility layer is similar to the IP protocol in the internet but more extensive. It manages computer resources and exposes APIs to applications, making it more similar to operating systems such as Linux.

The second is the Intercloud Layer, which is also an Abstraction layer and is built on the Compatibility layer. By blocking technical differences between users and cloud service providers, Intercloud layers can enable users to focus more on application development and deployment without worrying about the implementation details of underlying technologies. In my opinion, an Intercloud layer is actually the prototype of an Intercloud broker service because it allows users to express their preferences between performance, availability, and cost, which is the same as the cloud broker service.

Finally, Peering Between Clouds refers to the collaborative relationship between cloud service providers, which allows different cloud platforms to connect to each other, allowing users to move their applications and data between multiple cloud platforms. It aims to give users more flexibility in using cloud services without worrying about application and data migration issues.

Overall, this paper introduces the concept and theoretical framework of Sky Computing, which provides a new research direction for academia and industry, promotes the development and innovation of cloud computing technology, and provides a theoretical and practical foundation for building a more compatible and flexible cloud computing ecosystem.

4.2 The Sky Above The Clouds

RISELab of UC Berkeley proposed this white paper, and its members still include Scott Shenker and Ion Stoica. In my opinion, the first paper is an "innovator" in the field of cloud computing in recent years. It proposes the concept and architecture of Sky Computing, which is more like a draft or proposal. It does not discuss in detail how to implement Sky Computing and its advantages and challenges. This white paper is a severe planning paper that significantly enriches the concept and architecture of Sky Computing, discusses in more detail how to implement Sky Computing, and provides more specific solutions. And most importantly, it officially proposes the indispensable role of intercloud broker in implementing Sky Computing.

The white paper first discusses the definition of Sky Computing. It emphasizes the importance of cloud brokers: Although many cloud service providers are available in the market, different cloud service providers use different APIs, making data migration between other clouds very cumbersome. Users need to handle data migration and write API calls for various cloud services themselves. To more conveniently utilize the advantageous services of different cloud service providers, it is necessary to have an intermediate layer (cloud broker) to connect different cloud services and provide users with a unified API to describe tasks. The intermediate layer is responsible for scheduling services between different clouds.

It should be emphasized that the article points out that the cloud broker service does not redefine a unified API and require all cloud service providers to follow it, as this may lead to overly complex standards that are difficult to implement and may also hinder innovation. Therefore, the intercloud broker service will define a set of compatible feature sets, and cloud services that implement corresponding features can be added to Sky Computing. Users can specify their desired features and submit them to the Intercloud broker for analysis and work. We will conduct a detailed analysis of the Intercloud broker service in the following chapters.

Additionally, I believe that this white paper addresses a vital user pain point: Sky Computing eliminates the need for users to directly interact with cloud service providers, purchase services and deploy software provided by cloud service providers, and manually migrate data from one cloud to another, eliminating the need for tedious system operations. Users only need to specify the running script for the workload and assign it to the intercloud broker for processing, which can be automatically completed.

This white paper also discusses in detail the advantages and challenges of Sky Computing. Simply put, its benefits include interoperability across cloud providers, dynamic resource allocation, multi-tenant support, security, and privacy protection. However, it also faces challenges, such as competition among cloud providers, security and privacy protection, and multi-tenant support. And there is a problem that Sky Computing will meet in the early stages: the tasks and functions it supports in the early stages may be very limited.

In summary, this white paper greatly enriches the content of Sky Computing. It explores its concepts, characteristics, advantages, challenges, and future development directions, providing a strong "blueprint" for the future development of Sky Computing. At the same time, this article also clearly calls on everyone to work hard to achieve Sky Computing, build a fine-grained Two-sided market, and cloud service providers should avoid price wars. It has taken an enormous step forward in the development of Sky Computing.

4.3 SkyPilot: An Intercloud Broker for Sky Computing

The first two articles clarified the concept, general framework, and specific development direction of Sky Computing. And this article indeed implements the intercloud broker in Sky Computing and has already applied it to practical projects. At the same time, the paper also proposes a new direction: SkyPilot allows users to run tasks on local clusters, which can use idle regional clusters and migrate workloads to cloud services when the load on the local cluster is too high.

This article introduces SkyPilot, which I believe is the cloud broker service envisioned in the previous two reports. Simply put, it is an automated cloud resource selector that can select the best cloud resources to run different types of tasks based on user requirements and budgets. We will discuss in detail the exploration and analysis of SkyPilot later.

At the same time, to verify the performance of SkyPilot, multiple experiments were conducted in this paper, such as ML pipeline experiments and bioinformatics experiments. Through the experimental results of these experiments, the article proves that SkyPilot can indeed provide better performance and lower operating costs and that SkyPilot has transformed the diversity of cloud services from a "potential flaw" to an advantage: to meet the task needs of users, Regardless of the hardware configuration and geographical location of a cloud resource, as long as it meets the requirements, SkyPilot will find the cloud resource and use it.

In addition, the article also conducted a formal economic analysis of future cloud services, predicting that cloud services will be divided into two categories: one type of cloud service providers will continue to maintain proprietary cloud services, providing their APIs and charging data export fees; Another type of cloud service provider will join Sky Computing's cloud services, fully accepting open source interfaces and conducting data peer-to-peer with other cloud service providers who join Sky.

Overall, this article implements the intercloud broker service, optimizes the diversity of cloud services, supports local cluster operation tasks, and conducts economic analysis. Significant improvements and innovations have been made compared to the previous two articles.

5. Project Assumptions and Expectations

After completing a detailed review of relevant literature, we are already familiar with Sky Computing and have a clear understanding of the role and importance of cloud brokerage services in the field of Sky Computing. Therefore, we can start our research, but before that, we need to make some initial assumptions and expectations for this project to help us better experiment.

1. The transition from cloud computing to Sky Computing: We assume that by improving and optimizing the basic architecture of cloud computing, Sky Computing can be achieved. This includes optimizing the essential components of cloud computing, such as computing nodes, storage nodes, and network nodes, and improving the management and scheduling strategies of cloud computing. We expect Sky Computing to provide higher flexibility and efficiency, better meeting users' needs, including computing resources, data processing and migration capabilities, and service quality.

2. Future Development of Sky Computing: We assume that technological and market forces will drive the development of Sky Computing. This includes the development of new computing technologies, such as edge computing, fog computing, and the emergence of new market demands, such as Big data processing, artificial intelligence, and the Internet of Things. We expect sky computing to be widely applied in the next few years and become an important development direction for cloud computing.

3. Implementation of Intercloud broker service: We assume that by implementing the Intercloud broker service, we can fully utilize the diversity of cloud services and support local cluster running tasks. This includes the selection and scheduling of cloud services, the allocation and execution of jobs, and the monitoring and management of service quality. We anticipate that cloud-based agents will improve the efficiency of cloud service usage, reduce user costs, and provide a better user experience. In addition, we expect that the cloud broker service will play a crucial role in the development of Sky Computing, becoming an important bridge connecting different cloud services.

4. Performance and efficiency of cloud broker service: We assume that cloud brokers' performance and efficiency will significantly impact the implementation and application of Sky Computing. This includes task scheduling strategies, resource management strategies, and service quality management strategies for cloud agents. We expect to further improve the performance and efficiency of sky computing by optimizing the performance and efficiency of cloud-based agents.

5. Reproduction of ML pipeline experiment: We assume that the ML pipeline experiment in the paper can be successfully replicated, and its results can provide valuable data for our research. This includes the design and implementation of the ML pipeline, the collection and analysis of experimental data, and the interpretation and evaluation of experimental results. We anticipate that through replication experiments,

we can gain a deeper understanding of the performance and effectiveness of intercloud brokers in practical applications and conclude that Sky Computing saves time and money compared to single clouds for such experiments.

These preliminary assumptions and expectations will guide our research and help us better understand the transition from cloud computing to sky computing and the implementation of SkyPilot.

6. Methodology

In this project, we adopted a hybrid research method that combines theoretical and empirical research to gain a deeper understanding of the application and impact of cloud brokerage services in Sky Computing.

1. Theoretical research: We first conducted an extensive literature review, including key papers such as "From Cloud Computing to Sky Computing," "The Sky Above The Clouds: A Berkeley View on the Future of Cloud Computing," and "SkyPilot: An Intercloud Broker for Sky Computing." These papers provide us with the fundamental theories and concepts of cloud computing, sky computing, and intercloud broker services. We have conducted an in-depth analysis and understanding of these theories and concepts to establish our research framework.

2. Empirical research: Based on theoretical research, we selected an ML pipeline experiment from the paper "The Sky Above The Clouds: A Berkeley View on the Future of Cloud Computing" for replication. Through replication experiments, we can gain a deeper understanding of the performance and effectiveness of intercloud brokers in practical applications. We collected experimental data, conducted detailed analyses, and explained and evaluated the experimental results.

3. Exploration of Intercloud Brokerage Service: We have conducted an in-depth exploration of the implementation of SkyPilot, including its working principle, performance, and efficiency, as well as its application and impact in sky computing. From this, we analyzed how intercloud broker services optimize the diversity of cloud services and improve the efficiency and user experience of cloud service usage.

Through this hybrid research method, we can deeply understand the application and impact of cloud proxy services in sky computing from both theoretical and practical perspectives, thereby providing valuable research results for the development of sky computing.

7. Sky Computing And Intercloud Brokerage Service

Although we have already discussed the relevant concepts and developments of Sky Computing and Intercloud Brokerage Service when reviewing those three papers earlier, those are mostly the opinions and ideas of the paper authors. Therefore, in this section, I will summarize the concepts, developments, and related content of Sky Computing and Intercloud Brokerage Service based on my own understanding of them.

7.1 Sky Computing from the Perspective of ordinary users

Before starting this independent project, my understanding of Sky Computing in this independent project is limited to the content taught by Professor Wang Wei in class. Before this, I had almost no knowledge of Sky Computing, and I believe this is also the current situation for most ordinary users (or developers). The reason is simple: Sky Computing is relatively far from ordinary users. Why do I say that? Taking one of the most prominent features of Sky Computing - the ability to achieve data transmission parity between cloud service providers but many ordinary users do not even have the need for Data migration, or under the monopoly advantage of cloud service providers, they have "got used to" the high data export fees of cloud service providers.

For example, Baidu Netdisk, the Cloud storage service provided by Baidu Cloud, has forced users to endure its high data export fees and poor user experience because of its monopoly advantage in China. As a user, it needs to pay 10 yuan a month to enjoy the super ample cloud storage space provided by cloud service providers. When you have a super much Cloud storage space, you will find that, The data transmission speed when you retrieve your data may only be 100k/s. According to calculations, if your total data size is 3T, you need at least one year to recover all your data. For this reason, you have to activate the super membership service provided by the cloud service provider at a price of 30 RMB per month to obtain higher transmission rates. From this, it can also be seen how high the data export cost of cloud services is.

Name	Membership mechanism	Storage	Download bandwidth	Cost
Baidu NetDisk	Non-merber	105G	1Mbps	0 CNY/Year
	Backup Package Member	3T	1Mbps	108 CNY/Year
	SVIP	5T	300Mbps	298 CNY/Year
Tencent MicroCloud	Non-merber	10G	16Mbps	0 CNY/Year
	VIP	1T	16Mbps	600 CNY/Year
	SVIP	6T	Depends on user bandwidth	360 CNY/Year
Bitqiu NetDisk	Non-merber	20G	1Mbps	0 CNY/Year
	VIP	3T	100Mbps	198 CNY/Year
	SVIP	6T	300Mbps	498 CNY/Year

Table1. Price comparison of Cloud storage services provided by three cloud service providers in China

At the same time, as cloud computing belongs to a professional technology in the computer field, its use has a certain threshold, so its user base is probably ordinary developers. For ordinary developers, virtual machines are one of the most commonly used services, so overall, their demand range may be very narrow, to the point where they may not intersect with the advantages and features of Sky Computing. Therefore, I have concluded that Sky Computing is relatively far from ordinary users.

For me, Sky Computing is a new type of computing model that has been developed based on cloud computing. Its core idea is to connect multiple cloud computing platforms distributed worldwide through the network, forming a unified and large-scale computing environment that is ubiquitous and omnipotent, like the sky. The future it envisions is very idealistic, and its design intention is perfect. However, the above analysis shows its development path is still full of challenges and obstacles.

7.2 The Architecture of Sky Computing

In recent years, the theoretical architecture of Sky Computing has undergone multiple updates and optimizations. In the architecture of Sky Computing, we can layer various components in more detail to better understand their working principles. Based on my understanding, I will divide Sky Computing into the following top-down hierarchical structure:

Application layer: This layer contains the user's application programs. These applications can be of various types, such as data analysis, machine learning, Web services, etc. These applications interact with the cloud proxy layer through APIs, request computing resources, submit tasks, and obtain results. This layer also needs to handle user input and output and application state management.

Intercloud layer: This layer is the core of the Sky Computing architecture, responsible for coordination and management among multiple cloud service providers. The intercloud broker needs to deal with resource allocation, task scheduling, Data migration, and other issues. It needs to interact with the resource management layers of various cloud service providers to obtain and use computing resources.

Compatibility layer: This layer is part of the Intercloud layer, responsible for handling compatibility issues between different cloud services, providing unified APIs and service interfaces, so that applications do not need to worry about the specific implementation of the underlying cloud services.

Cloud service provider layer: This layer includes various cloud service providers, such as Amazon AWS, Google Cloud Platform, Microsoft Azure, etc. Each cloud service provider has its own API and services, which need to be unified through the cloud layer. This layer also includes the internal resource management systems of each cloud service provider, such as Amazon's EC2 and Google's Compute Engine. These systems are responsible for managing and scheduling the computing resources of cloud service providers.

Hardware layer: This layer is the actual computing resources, including servers, storage devices, network devices, etc. The resource management systems of various cloud service providers manage these resources.

Security and privacy protection layer: This layer runs through the entire architecture and includes various security and privacy protection mechanisms, such as data encryption, access control, privacy protection, etc. These mechanisms need to be effectively applied throughout the entire architecture to protect the security and privacy of user data. In addition, this layer also needs to address the issue of trust relationships between cloud services to ensure the safety of resource access and data exchange in cross-cloud environments.

In addition, the architecture of Sky Computing also needs to consider the dynamic scalability of cloud services, which dynamically increases or decreases computing resources according to the application's requirements. This requires a high degree of flexibility and adaptability in the cloud layer in order to dynamically schedule resources and balance loads among multiple cloud service providers.

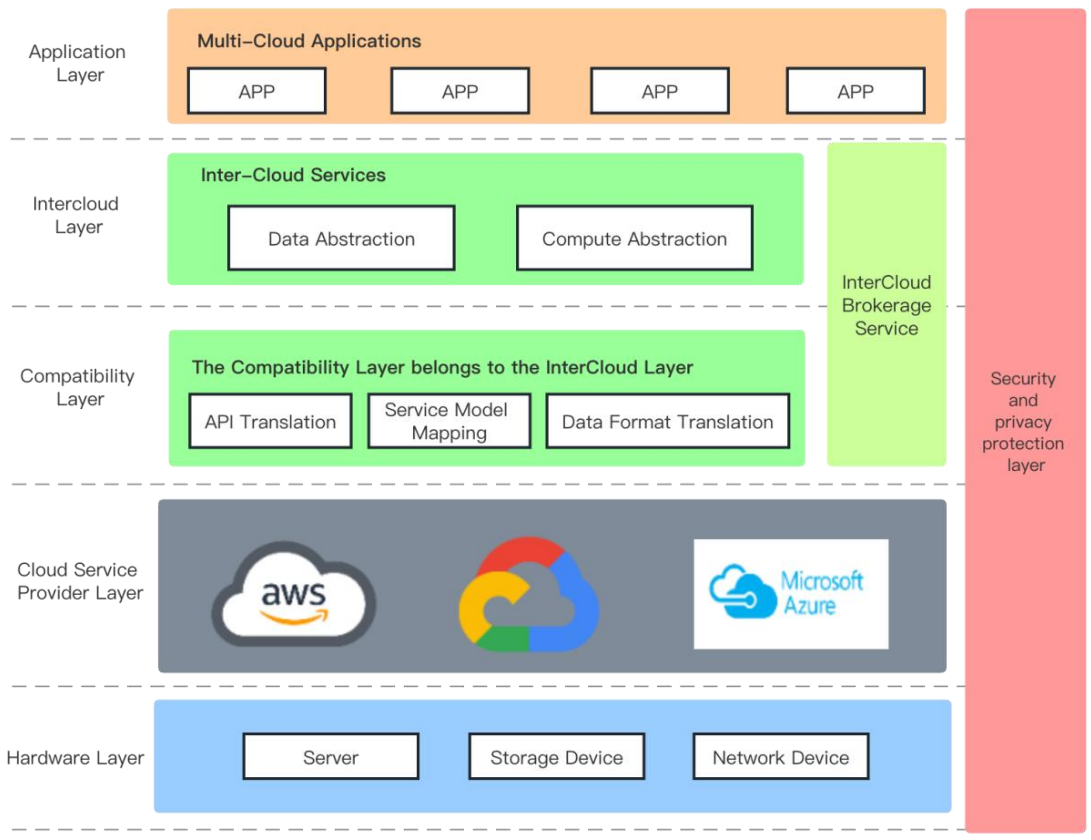


Figure1. The Architecture of Sky Computing

In addition, it is worth mentioning that in the early design of Sky Computing architecture, researchers adopted the Architectural pattern of the Internet, some of which are still in use today. Objectively speaking, the architecture of the Internet and Sky Computing will indeed be very similar. Table 2 shows the similarities between the two: routers are similar to servers, Autonomous Systems are similar to availability zones, and ISPs are similar to cloud providers. The architecture of the Internet has contributed a great deal of theoretical basis to the architecture design of Sky Computing.

Internet	Sky Computing
Autonomous System	Datacenter / Availability Zone
Router	Server
Internet Service Provider	Cloud Provider
Internet Protocol	Compatibility Layer
BGP	Intercloud Layer

Table2. The Comparison of Internet and Sky Computing Architecture

7.3 The Concept of Intercloud Brokerage Service

The concept of Intercloud Brokerage Service has evolved significantly with the development of Sky Computing. The role of an Intercloud Broker is to facilitate the interaction between multiple cloud service providers and the end-users. It acts as an intermediary, managing resources, scheduling tasks, and handling data migration among different cloud providers.

In the early stages of Sky Computing, the focus was primarily on creating a unified platform that could integrate resources from multiple cloud providers. The Intercloud Brokerage Service was seen as a means to achieve this integration, providing a single interface for users to access and manage resources across different clouds. Figure 2 shows the architecture of the early Intercloud Brokerage Service.

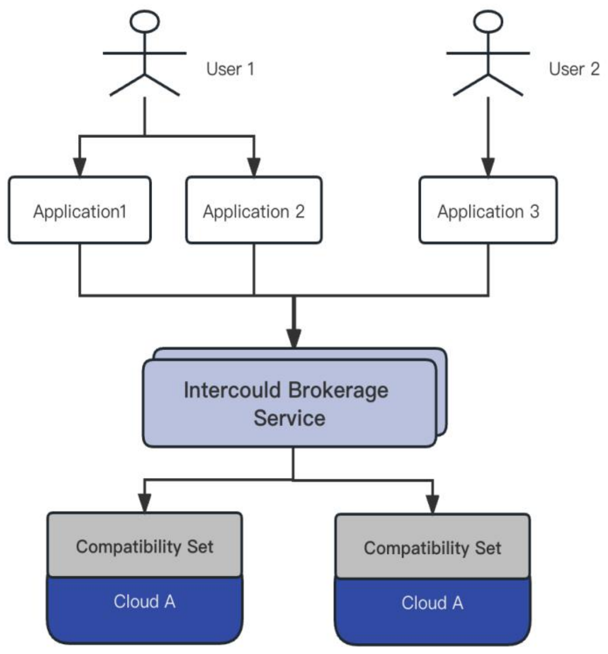


Figure2. The architecture of the early Intercloud Brokerage Service

As Sky Computing evolved, the role of the Intercloud Brokerage Service expanded. It started to handle more complex tasks such as managing compatibility issues between different cloud services, providing a unified API and service interface, and dynamically allocating resources based on application needs. This evolution was driven by the increasing complexity of cloud services and the growing demand for more flexible and efficient resource management.

In recent years, the development of Intercloud Brokerage Service has been focusing on improving the efficiency and reliability of resource management across multiple cloud providers. New strategies and algorithms have been developed to optimize resource allocation, task scheduling, and data migration. Moreover, there has been a growing emphasis on ensuring the security and privacy of data in the Intercloud Brokerage Service.

The future development of Intercloud Brokerage Service is expected to focus on further improving the efficiency, reliability, and security of resource management in Sky Computing. This includes developing more advanced strategies for resource allocation and task scheduling, improving the compatibility between different cloud services, and enhancing security and privacy protection mechanisms.

7.4 The Importance of Intercloud Brokerage Service

The importance of Intercloud Brokerage Service in Sky Computing cannot be ignored. With the popularization of cloud computing, enterprises and individual users increasingly rely on cloud services to handle various computing tasks. However, due to differences in service characteristics, prices, and service quality among various cloud service providers, users often need to use multiple cloud services to meet their needs. This poses a problem of how to effectively manage resources and schedule tasks among multiple cloud services.

The Intercloud Broker Service is an intermediary that can help users manage resources and schedule tasks across multiple cloud services. It can dynamically allocate resources and schedule tasks based on user needs and the characteristics of various cloud services. This not only improves the efficiency of resource utilization and reduces computational costs, but also improves the execution speed of tasks and improves the quality of services.

In addition, the Intercloud Brokerage Service can also help users handle compatibility issues between various cloud services. Due to the fact that different cloud service providers may use different technologies and standards, this may lead to compatibility issues. The Intercloud Broker Service can provide a unified API and service interface, allowing users to simplify their operations without worrying about the specific implementation of the underlying cloud services.

It is worth noting that the Intercloud Brokerage Service also needs to address the "data gravity" issue. Data gravity refers to the aggregation effect of data in a cloud environment, where the more data there is, the greater its attractiveness to computing resources. This is because the cost of data migration is often very high, especially in cross-cloud environments. Figure 3 shows the various types of files that users store in the cloud, and regardless of the type of file they store in the cloud, they will face high data export costs when migrating these files. Therefore, the Intercloud Brokerage Service needs to consider the issue of data gravity to determine how to allocate resources and schedule tasks optimally.

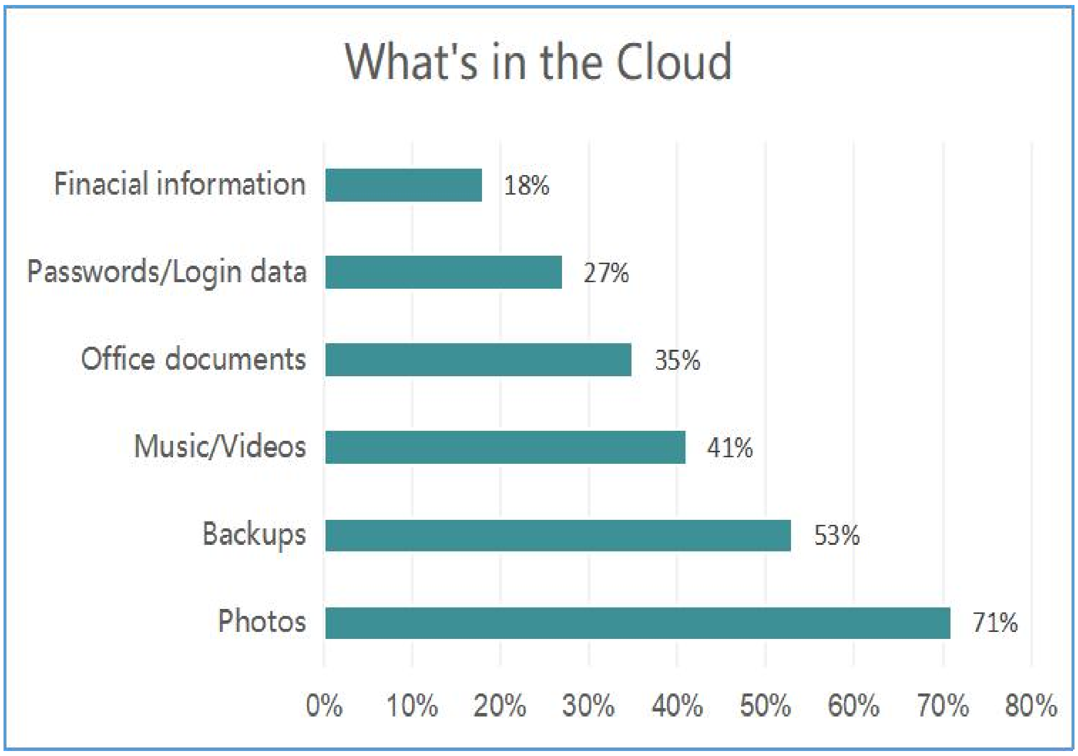


Figure 3. What's in the cloud?

Overall, the Intercloud Brokerage Service plays a crucial role in Sky Computing. It not only helps users effectively manage resources and schedule tasks between multiple cloud services, but also helps users handle compatibility issues, solve data gravity issues, and improve their user experience. In the future, with the further development of cloud computing technology, the role of Intercloud Broker Service will become more critical. It will play a more significant role in improving the efficiency and quality of cloud services, reducing the cost of cloud services, and protecting the security and privacy of user data.

7.5 The Architecture of Intercloud Broker Service

After recent years of development, the mainstream architecture of Intercloud Brokerage Service has gradually been finalized. Based on my understanding, I will introduce what I believe is the most mainstream architecture of Intercloud Brokerage Service. The architecture of the Intercloud Broker Service mainly includes the following key components:

User interface: This is the entry point for users to interact with the Intercloud Broker Service. Users can submit tasks, request resources, and obtain task execution results through this interface. This interface typically provides a set of APIs that allow users to interact with services through programming or scripting.

Catalog: The service catalog is an essential component of the Intercloud Brokerage Service. It provides a centralized location for storing and managing information about all available cloud services. The service catalog needs to interact with the APIs of various cloud service providers to obtain and update service information.

Task scheduler: This component is responsible for receiving user task requests and scheduling them to the appropriate cloud service provider for execution. The task scheduler needs to consider various factors, such as task requirements, cloud service provider capabilities, and current resource usage, to determine how to allocate tasks.

Optimizer: The optimizer plays a crucial role in the Intercloud Brokerage Service architecture. It is responsible for optimizing resource allocation and task scheduling to improve overall performance and efficiency. The optimizer needs to consider various factors, such as task requirements, resource availability, and current system load, to determine how to allocate resources and schedule tasks optimally.

Resource Manager: This component is responsible for managing all available cloud resources. It needs to interact with the APIs of various cloud service providers to obtain resource status information and submit resource requests. The resource manager also needs to handle resource allocation and release, as well as resource migration.

Data Manager: This component is responsible for managing user data. This includes data storage, data migration, and data protection. The data manager needs to interact with the data services of various cloud service providers to achieve data storage and migration.

Security Manager: This component is responsible for protecting the security of user data and tasks. This includes data encryption, access control, and security auditing. The security manager needs to interact with the security services of various cloud service providers to achieve security protection.

The relationships and workflow between the various components in the architecture of the Intercloud Broker Service are shown below, and Figure 4 shows its flowchart.

1. Users submit tasks and request resources through the user interface. These requests include the task's requirements, such as the required computing resources, task priority, and task execution time.

2. After receiving a user's request, the service directory will query its database to find cloud services that meet the user's needs. The service catalog will consider various factors, such as cloud service capabilities, prices, and service quality, to find the most suitable cloud service.

3. After receiving the results of the service directory, the optimizer will further optimize it. The optimizer will consider the current system state, such as resource usage, system load, and requests from other users, to determine how to allocate resources and schedule tasks.

4. After receiving the results of the optimizer, the task scheduler will schedule the task to the selected cloud service for execution. The task scheduler also monitors the execution status of tasks to ensure their successful execution.

5. The resource manager is responsible for managing all cloud resources. When the task scheduler schedules a task, the resource manager will allocate the required resources. When the task is completed, the resource manager will release the resources.

6. The data manager is responsible for managing user data. When a user submits a task, the data manager stores the user's data. When the task is completed, the data manager will return the results to the user.

7. The security manager is responsible for protecting the security of user data and tasks. The security manager encrypts user data to prevent unauthorized users from accessing it. The security manager also monitors the system's security status to avoid security threats.

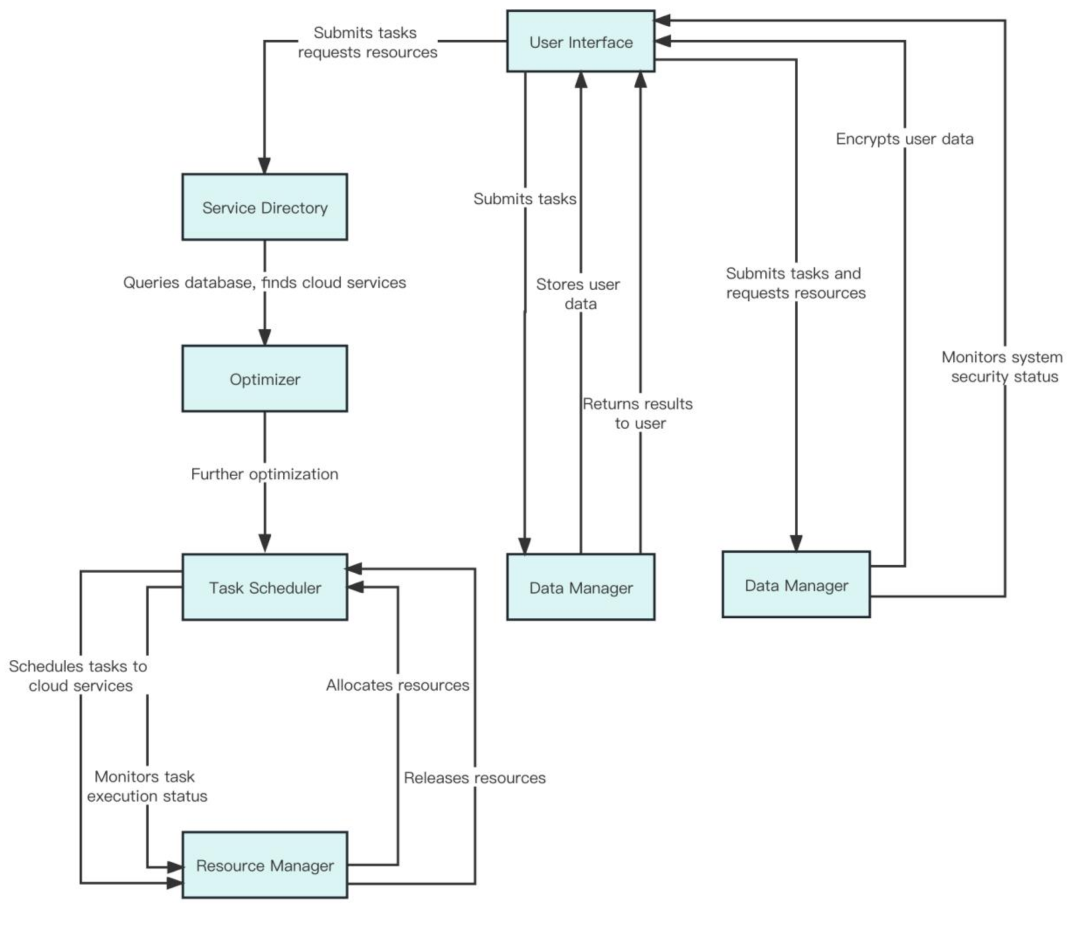


Figure 4. The flowchart of the intercloud brokerage service

The resource manager is responsible for managing all cloud resources. When the task scheduler schedules a task, the resource manager will allocate the required resources. When the task is completed, the resource manager will release the resources.

8. Analysis of the implemented Intercloud Broker Service - SkyPilot

After years of development, the Intercloud Brokerage Service evolved from a concept to a service theoretical framework, which was then truly implemented by UCSB's RISELab. This is undoubtedly a major breakthrough for Sky Computing. In this section, we will explore and analyze the completed Sky Computing-based Intercloud Brokerage Service project, SkyPilot, to understand its working principle and implementation details to help us further understand the Intercloud Brokerage Service.

8.1 Definition of SkyPilot

SkyPilot is essentially an Intercloud Broker service specifically designed for use in Sky Computing environments. In the Sky Computing environment, we face multiple cloud service providers, each with their own APIs, services, and resource management methods. Although this diversity gives users more choices, it also challenges management and coordination. This is where SkyPilot comes into play.

The primary responsibility of SkyPilot is coordinating and managing among these cloud service providers to achieve optimal allocation and use of resources. Its goal is to enable users to easily use multiple cloud services, just like using a single cloud service. To achieve this goal, SkyPilot needs to deal with various complex problems, such as resource allocation, task scheduling, Data migration, etc.

Simply put, SkyPilot is like an air traffic commander, scheduling and managing multiple cloud services to ensure that each task can be executed on the most suitable cloud service and that all cloud services can be effectively utilized. Through this approach, SkyPilot enables the concept of Sky Computing, which integrates multiple cloud services to form a large-scale virtual computing environment, providing more powerful and flexible computing capabilities.

8.2 The architecture and workflow of SkyPilot

The architecture of SkyPilot is designed to achieve effective resource management and task scheduling among multiple cloud services. It consists of the following main components:

(1) Application API: This is the user interface of SkyPilot, through which users submit tasks and request resources. This interface needs to be designed simple enough for users to use easily and also powerful enough to support various complex tasks and requests. It allows users to define applications in the form of coarse-grained tasks. These tasks can be of various types, including data processing, model training or High-performance computing. The task starts executing after all tasks that provide its input are completed, and each task is self-contained, including its executable file and all library dependencies. At the same time, these tasks specify their input and output locations through cloud Object storage URIs and even provide estimates of their input and output sizes to help the optimizer estimate the cost of data transmission across the cloud.

Regarding the specific use of the Application API, users should specify the resources required for each task. For ease of understanding and use, users should encode these resources in the form of labels. If users want to specify a CPU count of 8 and an accelerator of Nvidia v100, they should specify "CPU: 8, accelerator: Nvidia v100" as follows. After specifying, the optimizer will use these resource tags to search for available resource candidates in the catalog for each task. SkyPilot also provides sufficient humanized options, allowing users to clearly specify what resources they need and the number of resources they need. For example, users can specify the number of instances they need through "numnodes: n." If not specified, the system will default to 1.

At the same time, SkyPilot will also optimize the running process based on the user's expected running time, that is, optimize the DAG. If the user specifies a short runtime, the optimizer may choose to sacrifice costs to accelerate the task. If the user does not specify a runtime, the optimizer will default to selecting the resource with the lowest hourly price to execute the task. The following is a usage example, which includes two steps: *Data cleansing and data analysis*.

```
# A simple application: clean -> analyze.
with Dag() as dag:
    clean = Task('clean', run='clean.py', arg='--data=$INPUT[0] --cleaned_data=$OUTPUT[0]')
        .set_input('s3://raw-data', size=200 * GB)
        # '?': saves to the cloud this op ends up running on.
        .set_output('?://cleaned-data', size=150 * GB)
        # Required resources. A set ({}) means pick any Resources.
        .set_resources({
            Resources(cpu=8),
            Resources(ram=32 * GB)})
        # A partial function: Resources -> time.
        .set_time_estimator(clean_time_estimator_fn)
    analyze = Task('analyze', run='analyze.py', arg='--data=$INPUT[0]')
        .set_input(clean.output(0))
        .set_resources({
            Resources(cpu=16),
            Resources(ram=64 * GB)})
        .set_time_estimator(analyze_time_estimator_fn)
    # Connect the tasks.
    clean >> analyz
```

List 1. A usage example

In this example, we have defined a Data cleansing task and a data analysis task. The Data cleansing task reads the original data, performs Data cleansing, and writes the cleaned data to output. The data analysis task receives the cleaned data as input and performs data analysis. These two tasks are encapsulated in a Dag object, passed to the optimizer to output the execution plan, and then passed to the provider and executor. This example demonstrates how to use SkyPilot's API to define an application that includes two tasks, as well as how to specify the input, output, required resources, and time estimator for the task.

(2) Catalog: Catalog is an essential component of SkyPilot. It contains detailed information about all available cloud services, which may include service capabilities, prices, service quality, etc. When a user submits a task or requests resources, the Catalog will use this information to find the cloud service that best meets the user's needs.

In the framework of SkyPilot, the role of Catalog is to serve as a central information base to store and manage the information of all cloud services. This information can assist other components of SkyPilot, such as optimizers and providers, in better resource management and task scheduling. For example, the optimizer can use the information in the catalog to select the optimal cloud service, and the provider can use the information in the catalog to configure and allocate resources.

Overall, the Catalog is a critical component of SkyPilot's implementation of cloud service management and task scheduling. It provides detailed and accurate cloud service information, enabling SkyPilot to manage resources and schedule tasks across multiple cloud services effectively.

(3) Optimizer: The optimizer is a critical component of SkyPilot, whose main task is to allocate cloud resources, locations, and hardware configurations for each task in the user-submitted requirements to better adapt to the user's needs, such as minimizing cost or time. To achieve this goal, it will select the optimal allocation by filtering the candidate resources provided by the Catalog and solving integer linear programming (ILP).

To screen candidate resources, the author proposes a new concept: clustering. Simply put, it is a tuple composed of user needs, namely cloud resources, instance types, and zones that meet user needs. For example, if the optimizer finds a resource accelerator='nvidia v100 'that meets the user's needs after filtering the resources provided by CataLog, it can be mapped to the cluster<AWS, us west-2a, p3.2x>or<Azure, westus2-1, NC6s_ V3>. The reason why the ILP method is used to select the optimal allocation is that in SkyPilot, the optimizer calculates and executes tasks at the zone level instead of the region level, which can avoid situations where different zones within the same region may have different instance types and prices.

Next, the optimizer describes the Assignment problem as a 0-1 ILP. This is achieved by establishing a model for the execution time and cost of each task on each feasible cluster, and then finding an optimal task to cluster mapping to minimize total cost or time.

Minimizing the total cost: the optimizer needs to consider two types of costs: the total cost of executing all tasks and the total cost of data transmission. The optimizer finds an optimal task to cluster mapping by solving a 0-1 ILP to minimize the sum of these two costs. These two costs can be expressed by the following integer Linear programming (ILP) formula:

$$Min_s \sum_{v \in V} s_v^T c_v + \sum_{(u,v) \in E} s_u^T Q_{uv} s_v$$

The meaning of this formula is as follows:

$\sum_{v \in V} s_v^T c_v$: This section represents the total cost of executing all tasks. For each task v, we have a vector s_v, which is a one-hot vector used to select a feasible cluster for task v. c_v is a vector that represents the cost of running task v on each feasible cluster. $s_v^T c_v$ is the cost of task v on the selected cluster, and then we sum all tasks to obtain the total cost of executing all tasks.

$\sum_{(u,v) \in E} s_u^T Q_{uv} s_v$: This part represents the total cost of data transmission. For each pair of dependent tasks (u, v), we have a matrix Q_{uv} whose (i, j) elements represent the data transmission cost when the parent task u maps to the i-th cluster of C_u and the subtask v maps to the jth cluster of C_v. So, $s_u^T Q_{uv} s_v$ is the cost of data transmission from task u to task v, and then we sum all dependent task pairs to obtain the total cost of data transmission.

So, this formula aims to find a task to cluster mapping that minimizes the total cost of executing all tasks and the total cost of data transmission. This problem can be solved by solving a 0-1 ILP.

Minimizing the total time: SkyPilot's optimizer aims to find a task to cluster mapping that minimizes the running time of the entire Task Dependency Graph (DAG). This time is Co-determination determined by the execution time of all tasks and the data transmission time between tasks. This goal can be expressed by the following integer Linear programming (ILP) formula:

$$Min_s f_{sink}$$

$$f_v \geq f_u + s_u^T P_{uv} s_v + s_v^T t_v \; \forall (u,v) \in E$$

$Min_s f_{sink}$: This section indicates that our goal is to minimize the runtime of DAG. Here, the runtime of a DAG is defined as the completion time of its "sink point" (i.e. the last completed task in the DAG), denoted as f_{sink}.

$f_v \geq f_u + s_u^T P_{uv} s_v + s_v^T t_v \ \forall (u, v) \in E$: This is the constraint condition for optimization. For each pair of dependent tasks (u, v) in DAG, the completion time f_v of task v must be no earlier than the completion time f_u of its parent task u, the data transmission time $s_u^T P_{uv} s_v$ from parent task u to task v, and the execution time $s_v^T t_v$ of task v on the selected cluster. This constraint ensures that the execution order of tasks satisfies their dependencies.

In this formula, $s_u^T P_{uv} s_v$ represents the data transmission time from task u to task v. This time is determined by which cluster task u and task v are executed on, respectively. If task u and task v are executed on the same cluster, then this time is 0; If they execute on different clusters, then this time is the time it takes for data to be transferred from the cluster of task u to the cluster of task v.

$s_v^T t_v$ represents the execution time of task v on the selected cluster. This time is Co-determination determined by the specific execution of task v and the computing power of the selected cluster.

By solving this 0-1 ILP problem, we can find a task to cluster mapping that minimizes the running time of the entire DAG.

(4) Provisioner: In SkyPilot, the primary responsibility of the Provisioner is to allocate and configure various computing resources required for each task based on the output of the optimizer. These computing resources may come from different cloud service providers, such as Amazon AWS, Google Cloud Platform, Microsoft Azure, etc. The workflow of the Provisioner is roughly as follows:

1. Receive the output of the optimizer, which includes which type of instance of which cloud service provider, region, and type each task should run on.

2. For each task, the Provisioner will request and configure the required computing resources through the API of the cloud service provider. This may include starting a new instance, configuring the network and storage, installing the necessary software, etc.

3. The Provisioner will continuously monitor the status of these computing resources to ensure their normal operation. If any errors occur, such as instance

crashes, network interruptions, etc., the Provisioner will attempt to fix these errors or request new computing resources if necessary.

4. After a task is completed, the Provisioner will be responsible for cleaning up and releasing the computing resources used by the task to reduce unnecessary costs.

Through this approach, the Provisioner can effectively manage and schedule computing resources, enabling users' tasks to run in the optimal environment while minimizing the complexity of underlying resource management that users need to care about.

(5) Executor: In SkyPilot, Executor is responsible for executing user tasks on the computing resources allocated and configured by the Provisioner. As mentioned earlier, these tasks are typically expressed in the form of a coarse-grained Task Dependency Graph (DAG), where each task can start executing after completing all its input tasks. Its workflow is as follows:

1. Receive the output of the optimizer: The Executor first receives the output of the optimizer, which includes the type of instance in which the cloud service provider, region, and region each task should run on, as well as the execution order of these tasks.

2. Start cluster: The Executor will start a cluster managed by the Ray framework on the specified computing resources. This cluster comprises a set of collaborative computing nodes, which can be seen as a distributed computing environment.

3. Upload the task and start execution: The Executor will upload the code and data of the task to the cluster, and then start the execution of the task on the cluster. This process may include setting up the running environment, parsing the input and output of tasks, and initiating task execution.

4. Monitor task execution status: The Executor will continuously monitor the execution status of the task, including task runtime, resource usage, output results, etc. If the task is successfully executed, the Executor will collect the output results of the task and pass them on to the next task. If the task execution fails, the Executor will attempt to execute the task again or report an error message.

5. Collect and organize results: After all tasks are completed, the Executor will be responsible for collecting and organizing the output results of all tasks, so that users can easily obtain and use these results.

6. Shut down the cluster and release resources: Finally, the Executor will be responsible for shutting down the cluster and releasing computing resources to reduce unnecessary costs

Overall, Executor enables users to easily run their distributed applications within the SkyPilot architecture by effectively managing and executing their tasks without worrying about underlying resource management and task scheduling issues. Figure 5 shows the workflow diagram of the Executor.

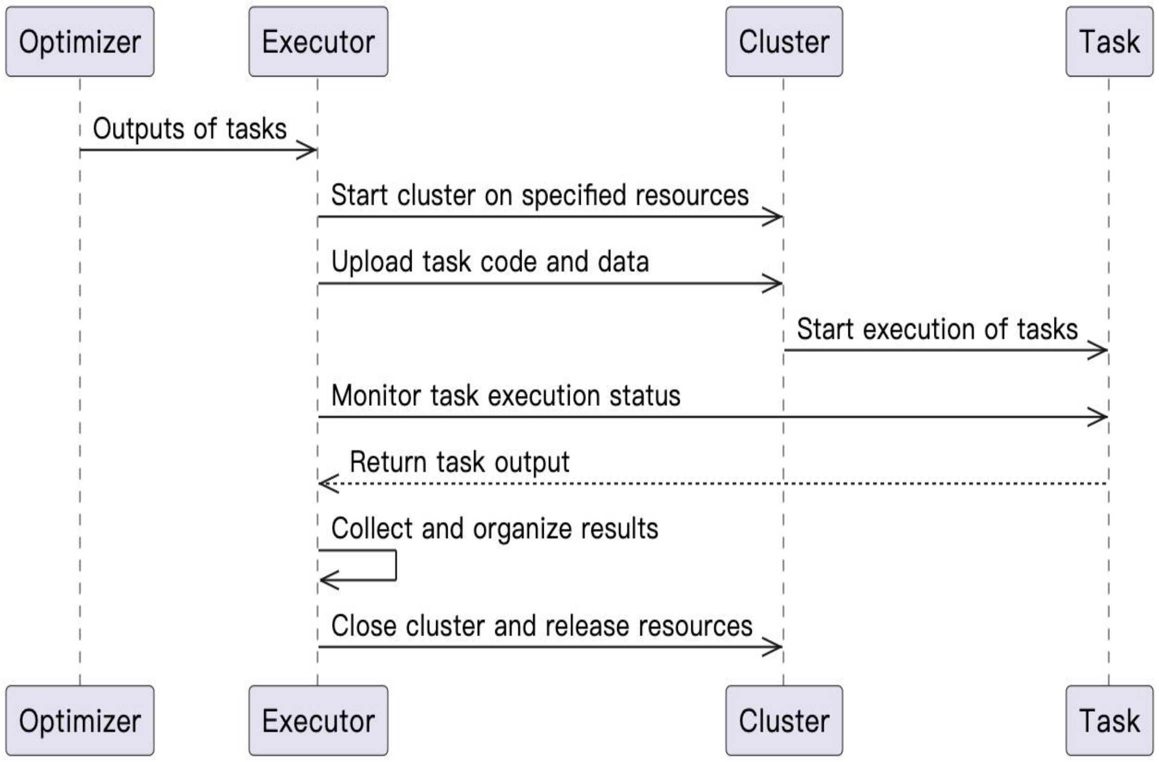

Optimizer
Executor
Cluster
Task
Outputs of tasks
Start cluster on specified resources
Upload task code and data
Start execution of tasks
Monitor task execution status
Return task output
Collect and organize results
Close cluster and release resources
Optimizer
Executor
Cluster
Task

Figure 5. The flowchart of the Executor

(6) Compatibility Set: In the SkyPilot architecture, "Compatibility set" is an essential component. Due to different cloud service providers providing different APIs and service interfaces, compatibility issues may be encountered when running applications in cross-cloud environments. To address this issue, SkyPilot introduced the "Compatibility set" component.

The working principle of the Compatibility Set is that it creates a compatibility set for each cloud service provider, which includes all APIs and service interfaces supported by that cloud service provider. Then, when the application needs to migrate between multiple cloud service providers, the "Compatibility set" checks the compatibility set of the target cloud service provider to see if it supports all the APIs and service interfaces required by the application. If supported, then the application can be smoothly migrated and run on the target cloud service provider. If not supported, the 'Compatibility set' will report an error telling the user that the migration cannot be completed.

Overall, Compatibility Set is a strategy used by SkyPilot to address cross-cloud service compatibility issues. It achieves seamless switching and coordination between different cloud services by using existing cloud services, APIs, and some glue code.

Figure 6 shows the complete workflow of SkyPilot.

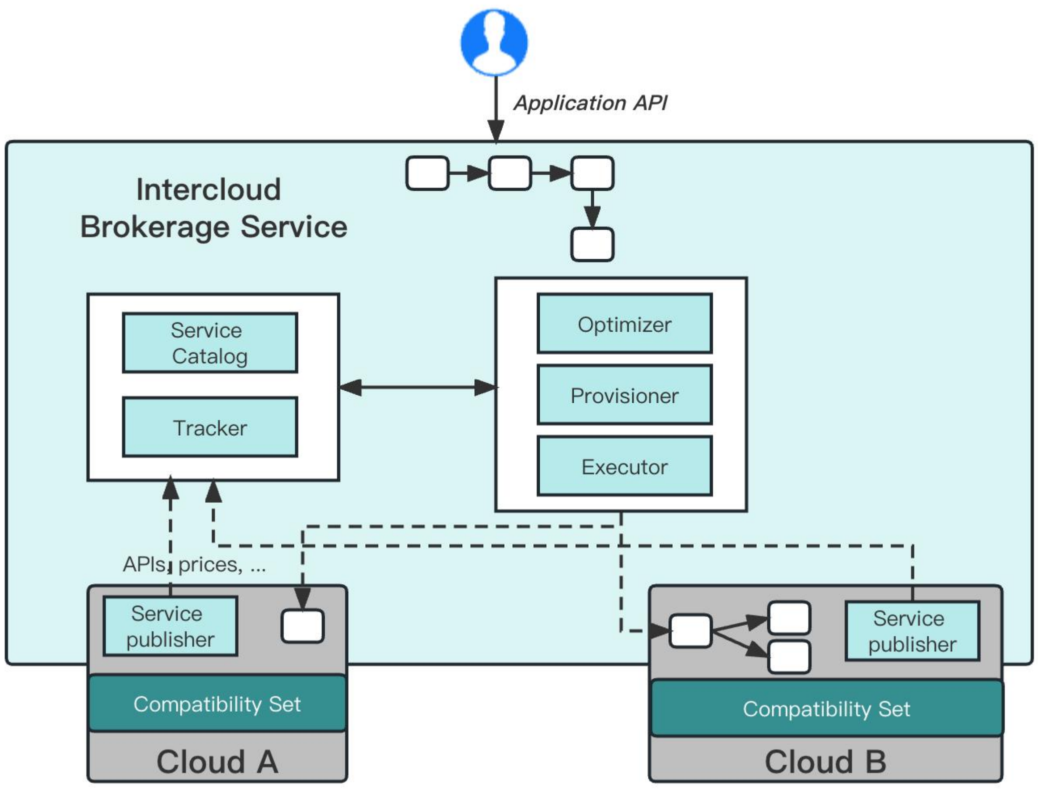


Figure 6. The workflow of SkyPilot

9. Experiment

In this section, we will study and reproduce the ML pipeline experiment mentioned in the paper "The Sky Above The Clouds." Here, I would like to emphasize the importance of the experiment implemented by the original author for Sky Computing, as well as the reason why I chose to replicate this experiment: this experiment is not only a technical demonstration but also an empirical study that proves the value of using the Intercloud Broker Service in practical applications. Using the Intercloud Broker Service can indeed reduce the time and money costs of task operation, And this is also one of the greatest significance of Sky Computing's existence.

9.1 Experimental Purpose

(1) Understanding and validating the original research: Repeating experiments can help us gain a deeper understanding of the methods and results of the original research. By implementing it ourselves, we can better understand each step and detail of the experiment and how these steps affect the final result. In addition, replication experiments can also verify the accuracy of the original research results and whether they can be replicated under different environments and conditions.

(2) Learning and mastering new technologies: Repeating experiments is an excellent way to learn and master new technologies. In this experiment, we need to use technologies such as Opaque and Intel SGX to process data. By using these technologies in practice, we can better understand their working principles and how to apply them in practical problems.

(3) Exploring new possibilities: Reproduction experiments can also provide us a platform to explore new possibilities and improve. For example, we can use different datasets or modify certain experiment steps to see how these changes affect the results. This may bring new discoveries or inspire us to improve the original research.

Overall, our experimental objective is to validate and demonstrate the value and effectiveness of Sky Computing and Intercloud Brokerage Service in practical applications. We showed how to use Intercloud Broker Service to schedule and manage resources among multiple cloud service providers by replicating ML Pipeline experiments, to optimize performance and reduce costs. The final conclusion is that using the Intercloud broker service can indeed reduce the time and monetary cost of task execution.

9.2 Experimental Design

The experiment will be divided into two parts:

(1) Single Cloud Experiment: In the single cloud experiment, we will only conduct our experiment on a separate cloud throughout the entire process and record the time and money spent in each stage. This single cloud experiment is essentially a comparative experiment.

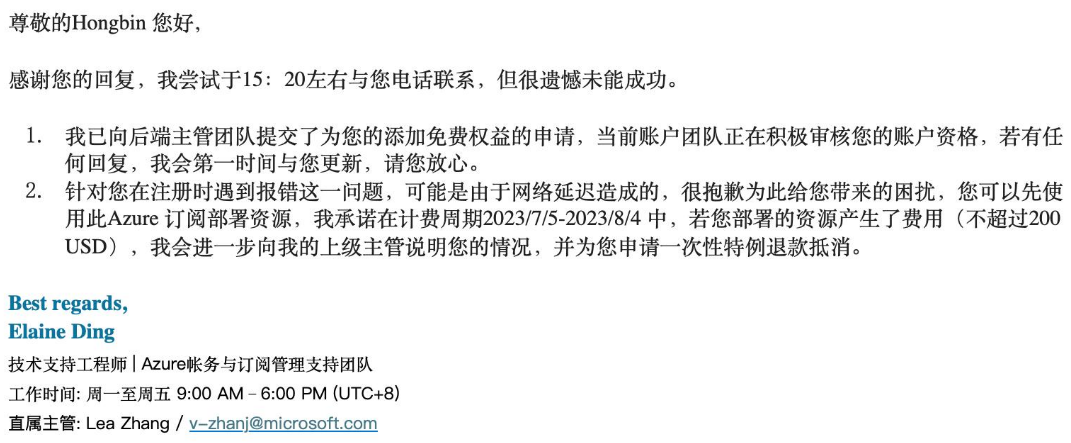
(2) Sky Computing experiment: In this experiment, we will manually simulate the Intercloud Brokerage Service, screen and determine the most suitable cloud for each stage of the ML Pipeline experiment, and record the time and money costs spent in each stage of the experiment.

After the experiment is completed, we will analyze and study the experimental results.

9.3 Single Cloud Experiment

Stage 1: Confidential data processing. At this stage, we will use a VM that supports SGX to run Opaque to process the Amazon customer review dataset. Opaque SQL is a package of Apache Spark SQL that enables encryption for DataFrames using the OpenEnclave framework. Its purpose is to analyze sensitive data in untrusted clouds. Once the content of the DataFrame is encrypted, subsequent operations will run within the hardware enclosure (such as Intel SGX). Due to the fact that only Microsoft Azure currently provides SGX virtual machines for Stage 1, whether it is a single cloud experiment or a Sky Collaboration experiment, we can only choose Azure for the experiment.

(1) Based on the above content, we must first create an Azure account and configure the relevant account information. It should be noted in this step that registering an Azure account requires the use of a credit card, preferably Master or Visa. Meanwhile, at this stage, I encountered a very tricky problem: I found that I could not use campus email when registering for an Azure account, as this would prevent me from receiving the 200USD gift from Azure for new users and from using the various services provided by Azure normally. After encountering this issue, I attempted to contact Azure's technical support personnel for assistance. After I applied for assistance, Azure customer service first promised me that I would receive a quota of 200 USD, as shown in Figure 7. Subsequently, Azure officials requested that I provide various identity information. Surprisingly, Azure officially banned my account after I offered my correct identity information and did not accept my appeal. As shown in Figure 8 below. Ultimately, I had no choice but to register for a new email (not a campus email) and use a new credit card to register for an Azure account. Ultimately, I successfully obtained a free quota of 200 USD and account integrity. As shown in Figure 9.

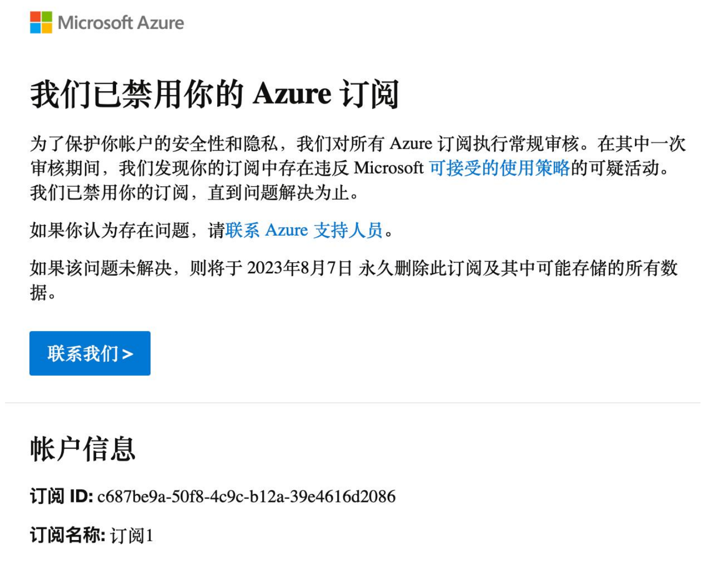

尊敬的Hongbin 您好，

感谢您的回复，我尝试于15：20左右与您电话联系，但很遗憾未能成功。

1. 我已向后端主管团队提交了为您的添加免费权益的申请，当前账户团队正在积极审核您的账户资格，若有任何回复，我会第一时间与您更新，请您放心。
2. 针对您在注册时遇到报错这一问题，可能是由于网络延迟造成的，很抱歉为此给您带来的困扰，您可以先使用此Azure 订阅部署资源，我承诺在计费周期2023/7/5-2023/8/4 中，若您部署的资源产生了费用（不超过200 USD），我会进一步向我的上级主管说明您的情况，并为您申请一次性特例退款抵消。

Best regards,
Elaine Ding
技术支持工程师 | Azure帐务与订阅管理支持团队
工作时间: 周一至周五 9:00 AM – 6:00 PM (UTC+8)
直属主管: Lea Zhang / v-zhanj@microsoft.com

Figure 7. The response of Customer service


Microsoft Azure

我们已禁用你的 Azure 订阅

为了保护你帐户的安全性和隐私，我们对所有 Azure 订阅执行常规审核。在其中一次审核期间，我们发现你的订阅中存在违反 Microsoft 可接受的使用策略的可疑活动。我们已禁用你的订阅，直到问题解决为止。

如果你认为存在问题，请联系 Azure 支持人员。

如果该问题未解决，则将于 2023年8月7日 永久删除此订阅及其中可能存储的所有数据。

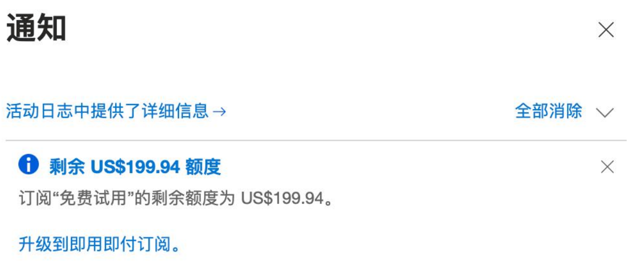
联系我们 >

帐户信息

订阅 ID: c687be9a-50f8-4c9c-b12a-39e4616d2086

订阅名称: 订阅1

Figure 8. My account has been banned


通知

活动日志中提供了详细信息 →

全部消除

剩余 US$199.94 额度

订阅"免费试用"的剩余额度为 US$199.94。

升级到即用即付订阅。

Figure 9. Free quota

(2) After resolving the issue with the Azure account, we need to deploy Opaque on the Azure SGX VM. The official document of Opaque states that "Note that Opaque SQL requires the MC2 Client to run an encrypted query securely." Therefore, we need to install MC2 first. MC2 is a platform for security analysis and machine learning of encrypted data, and running Opaque SQL using MC2 is more secure. But due to the fact that this project has not been maintained for nearly two years, the installation tutorials provided in the official documents are no longer applicable, which has led me to spend much time understanding the architecture of MC2 and how to install it. Finally, the installation was successful, but I also realized there was no need to install MC2 first. You can simply deploy Opaque on the SGX VM, because MC2 is essentially a task dispatch program or management panel, and the actual data processing process is completed by Opaque SQL.

So I abandoned MC2 and started installing Opaque directly, but at this point, there were two prerequisites for successfully deploying Opaque: 1. I needed to use a VM that supports SGX for deployment. To verify this, I attempted to deploy using a VM (Ubuntu 18.04) that does not support SGX, and in the end, an error message as shown in Figure 10 will appear; 2. It is necessary to use Ubuntu 18.04 for deployment. If a version above 18.04 is used, there will be a package conflict in the Ubuntu system kernel, resulting in Opaque SQL's inability to complete compilation, as shown in Figures 11 and 12.

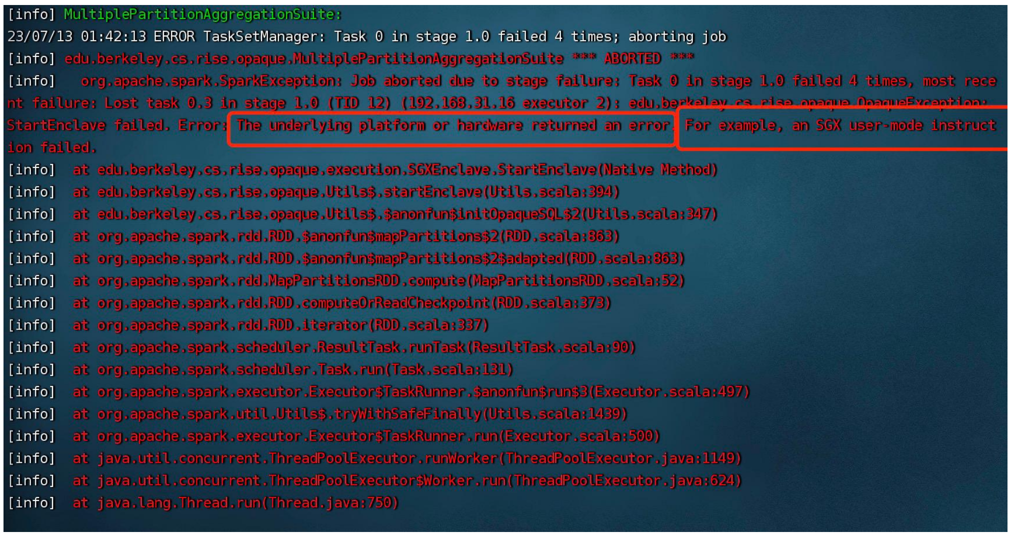

```
[info] MultiplePartitionAggregationSuite:
23/07/13 01:42:13 ERROR TaskSetManager: Task 0 in stage 1.0 failed 4 times; aborting job
[info] edu.berkeley.cs.rise.opaque.MultiplePartitionAggregationSuite *** ABORTED ***
[info]   org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 1.0 failed 4 times, most rece
nt failure: Lost task 0.3 in stage 1.0 (TID 12) (192.168.31.16 executor 2): edu.berkeley.cs.rise.opaque.OpaqueException:
StartEnclave failed. Error: The underlying platform or hardware returned an error. For example, an SGX user-mode instruct
ion failed.
[info]  at edu.berkeley.cs.rise.opaque.execution.SGXEnclave.StartEnclave(Native Method)
[info]  at edu.berkeley.cs.rise.opaque.Utils$.startEnclave(Utils.scala:394)
[info]  at edu.berkeley.cs.rise.opaque.Utils$.$anonfun$initOpaqueSQL$2(Utils.scala:347)
[info]  at org.apache.spark.rdd.RDD.$anonfun$mapPartitions$2(RDD.scala:863)
[info]  at org.apache.spark.rdd.RDD.$anonfun$mapPartitions$2$adapted(RDD.scala:863)
[info]  at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
[info]  at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)
[info]  at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)
[info]  at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
[info]  at org.apache.spark.scheduler.Task.run(Task.scala:131)
[info]  at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:497)
[info]  at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
[info]  at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
[info]  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
[info]  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
[info]  at java.lang.Thread.run(Thread.java:750)
```

Figure 10

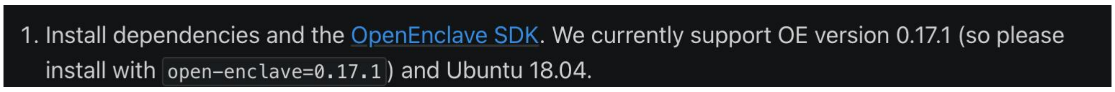

1. Install dependencies and the OpenEnclave SDK. We currently support OE version 0.17.1 (so please install with `open-enclave=0.17.1`) and Ubuntu 18.04.

Figure 11

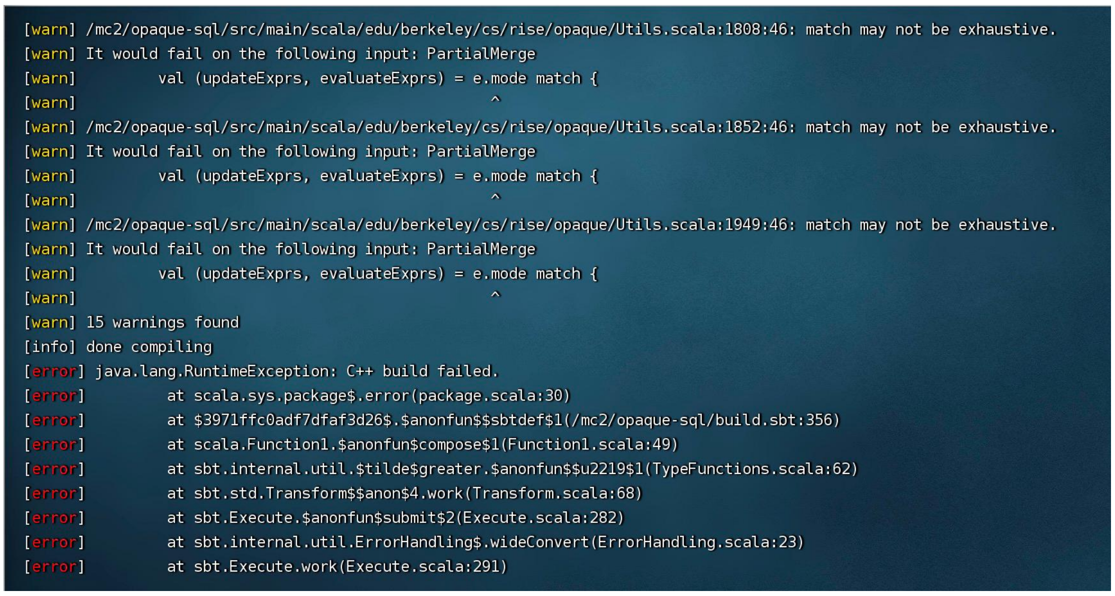
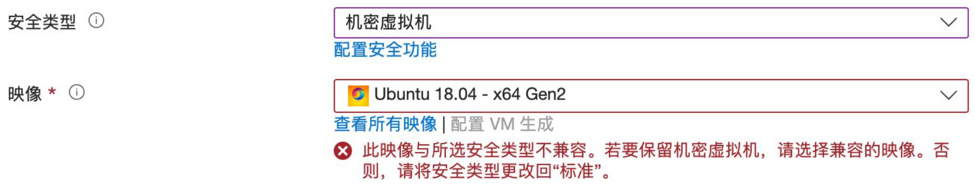

```
[warn] /mc2/opaque-sql/src/main/scala/edu/berkeley/cs/rise/opaque/Utils.scala:1808:46: match may not be exhaustive.
[warn] It would fail on the following input: PartialMerge
[warn]         val (updateExprs, evaluateExprs) = e.mode match {
[warn]                                             ^
[warn] /mc2/opaque-sql/src/main/scala/edu/berkeley/cs/rise/opaque/Utils.scala:1852:46: match may not be exhaustive.
[warn] It would fail on the following input: PartialMerge
[warn]         val (updateExprs, evaluateExprs) = e.mode match {
[warn]                                             ^
[warn] /mc2/opaque-sql/src/main/scala/edu/berkeley/cs/rise/opaque/Utils.scala:1949:46: match may not be exhaustive.
[warn] It would fail on the following input: PartialMerge
[warn]         val (updateExprs, evaluateExprs) = e.mode match {
[warn]                                             ^
[warn] 15 warnings found
[info] done compiling
[error] java.lang.RuntimeException: C++ build failed.
[error]         at scala.sys.package$.error(package.scala:30)
[error]         at $3971ffc0adf7dfaf3d26$.$anonfun$$sbtdef$1(/mc2/opaque-sql/build.sbt:356)
[error]         at scala.Function1.$anonfun$compose$1(Function1.scala:49)
[error]         at sbt.internal.util.$tilde$greater.$anonfun$$u2219$1(TypeFunctions.scala:62)
[error]         at sbt.std.Transform$$anon$4.work(Transform.scala:68)
[error]         at sbt.Execute.$anonfun$submit$2(Execute.scala:282)
[error]         at sbt.internal.util.ErrorHandling$.wideConvert(ErrorHandling.scala:23)
[error]         at sbt.Execute.work(Execute.scala:291)
```

Figure 12

The only cloud service provider that meets all the above requirements is Azure, so I searched for relevant resources in Azure. As a result, due to the outdated version of Ubuntu 18.04, this version stopped providing SGX confidential computing services on May 31, 2023, and version 18.04 is no longer maintained, as shown in Figure 13.


安全类型 ⓘ 机密虚拟机
配置安全功能

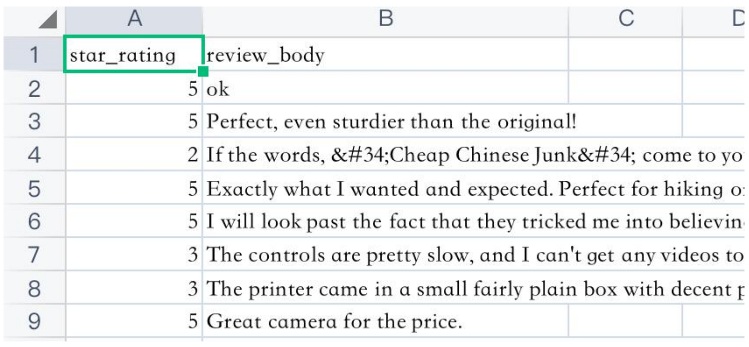
映像 * ⓘ Ubuntu 18.04 - x64 Gen2
查看所有映像 | 配置 VM 生成
此映像与所选安全类型不兼容。若要保留机密虚拟机，请选择兼容的映像。否则，请将安全类型更改回“标准”。

Figure 13

Simply put, both Opaque and MC2 development teams have been disbanded, and the project has ceased maintenance. After all, it's an open-source project, and I understand that. But the inability to solve this problem still makes me feel very frustrated.

(3) After the above argumentation, it is evident that the first stage of the experiment cannot be replicated. However, since the first stage of the experiment can only be conducted on Azure, the time and cost consumed in this stage are the same for both Sky Computing and single-cloud experiments. Therefore, I assume that the time and cost consumed in this stage are T and C, respectively. In the future, T and C can be added when calculating the total time and cost.

Since Opaque cannot be used to process the dataset, I will use Python's Pandas library to process the dataset and obtain the desired result, which only includes user comments and star ratings, as shown in Figure 14.

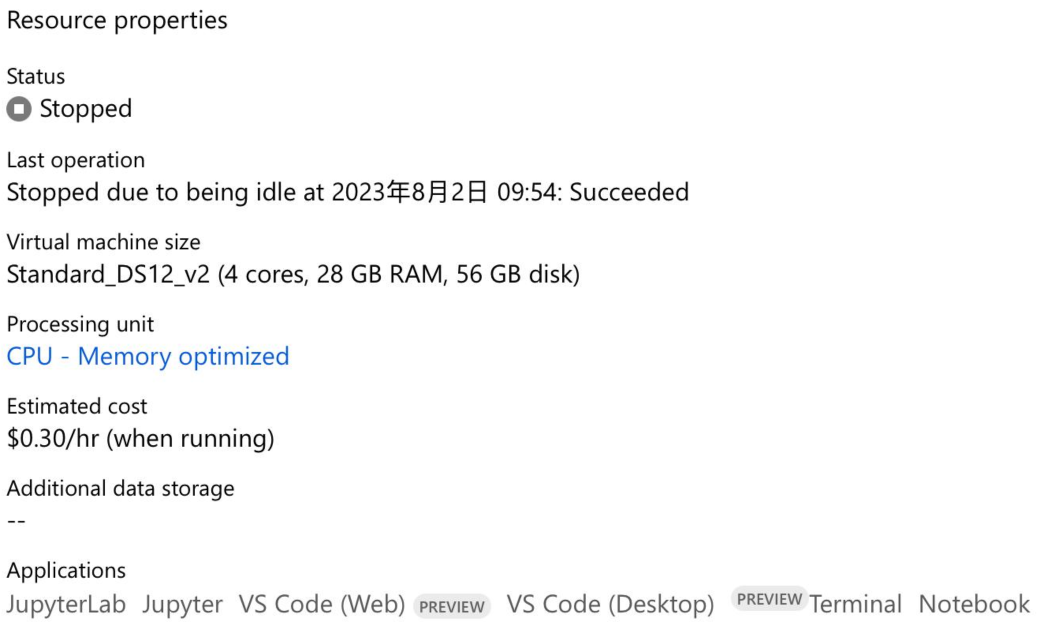


	A	B	C	D
1	star_rating	review_body		
2	5	ok		
3	5	Perfect, even sturdier than the original!		
4	2	If the words, &#34;Cheap Chinese Junk&#34; come to yo		
5	5	Exactly what I wanted and expected. Perfect for hiking o		
6	5	I will look past the fact that they tricked me into believin		
7	3	The controls are pretty slow, and I can't get any videos to		
8	3	The printer came in a small fairly plain box with decent p		
9	5	Great camera for the price.		

Figure 14

Stage 2: Model training stage. At this stage, we will continue to conduct experiments on Azure Cloud. In Stage 1, we have processed the original dataset and obtained a new dataset (which only contains review_body and star_rating columns). In Stage 2, we will use this dataset as the training set, and then fine-tune the BERT model. BERT is a very popular Natural-language understanding model. With this model, we can predict the score of a given comment text.

(1) Determine the services required for training the model: To facilitate statistical experimental data (time and cost) and visualize the experimental process, I have chosen to use Azure Notebooks to train the model, with VM configuration as Standard_ DS12_ V2 (4 cores, 28 GB RAM, 56 GB disks), the estimated cost is 0.30 USD/hour. As shown in Figure 15. Meanwhile, as my subscription is free, Azure only allows me to use the CPU, which means I can only use the CPU for this experiment.



Resource properties

Status
Stopped

Last operation
Stopped due to being idle at 2023年8月2日 09:54: Succeeded

Virtual machine size
Standard_DS12_v2 (4 cores, 28 GB RAM, 56 GB disk)

Processing unit
CPU - Memory optimized

Estimated cost
$0.30/hr (when running)

Additional data storage
--

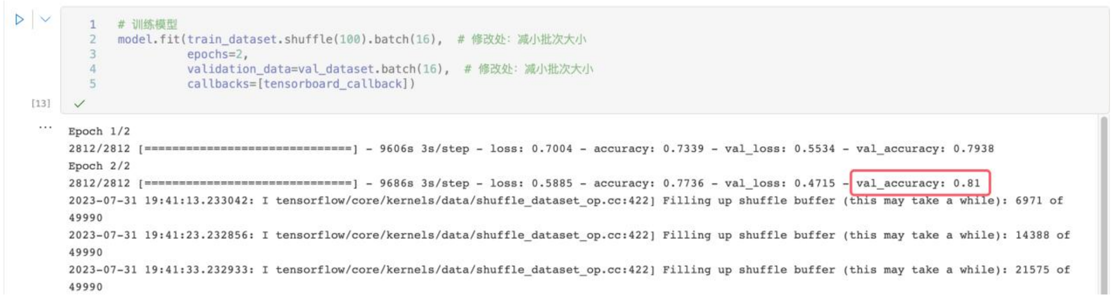
Applications
JupyterLab Jupyter VS Code (Web) PREVIEW VS Code (Desktop) PREVIEW Terminal Notebook

Figure 15

(2) Code: The following is the Pseudocode I used to train this model. Due to the limitation of VM performance, when the data volume exceeds 50000, the VM will crash. Therefore, I will adjust the data volume of the training set to the first 50000. At the same time, I will use 90% of the data in this training set as the training set and the remaining 10% of the data as the verification set to confirm the success of the model training.

```
IMPORT necessary libraries and modules
SET mixed precision policy to 'mixed_float16'

LOAD pre-trained BERT model with 5 output neurons
LOAD pre-trained BERT tokenizer

DEFINE dataset path and chunk size

FUNCTION generate_examples:
    FOR each chunk in dataset:
        FOR each row in chunk:
            CREATE InputExample object
            YIELD InputExample
        ENDFOR
    ENDFOR
END FUNCTION

FUNCTION generate_features:
    FOR each example in generate_examples:
        ENCODE text using tokenizer
        YIELD InputFeatures
    ENDFOR
END FUNCTION

FUNCTION gen:
    FOR each feature in generate_features:
        YIELD features and labels
    ENDFOR
END FUNCTION

CREATE TensorFlow dataset from gen function

CALCULATE dataset size, train size, and validation size
SHUFFLE and SPLIT dataset into training and validation sets

COMPILE model with optimizer, loss function, and accuracy metric

DEFINE TensorBoard log directory
CREATE TensorBoard callback

TRAIN model with training and validation sets, using TensorBoard callback

SAVE trained model to specified directory
```

(3) Running result: The model training was successful through the above code. This experiment took 10 ***hours*** and a total cost of 8 ***USD***. The final prediction accuracy of this model is 81%. As shown in Figure 16.



```
# 训练模型
model.fit(train_dataset.shuffle(100).batch(16),  # 修改处: 减小批次大小
          epochs=2,
          validation_data=val_dataset.batch(16),  # 修改处: 减小批次大小
          callbacks=[tensorboard_callback])
```

```
[13]
Epoch 1/2
2812/2812 [==============================] - 9606s 3s/step - loss: 0.7004 - accuracy: 0.7339 - val_loss: 0.5534 - val_accuracy: 0.7938
Epoch 2/2
2812/2812 [==============================] - 9686s 3s/step - loss: 0.5885 - accuracy: 0.7736 - val_loss: 0.4715 - val_accuracy: 0.81
2023-07-31 19:41:13.233042: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:422] Filling up shuffle buffer (this may take a while): 6971 of
49990
2023-07-31 19:41:23.232856: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:422] Filling up shuffle buffer (this may take a while): 14388 of
49990
2023-07-31 19:41:33.232933: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:422] Filling up shuffle buffer (this may take a while): 21575 of
49990
```

Figure 16

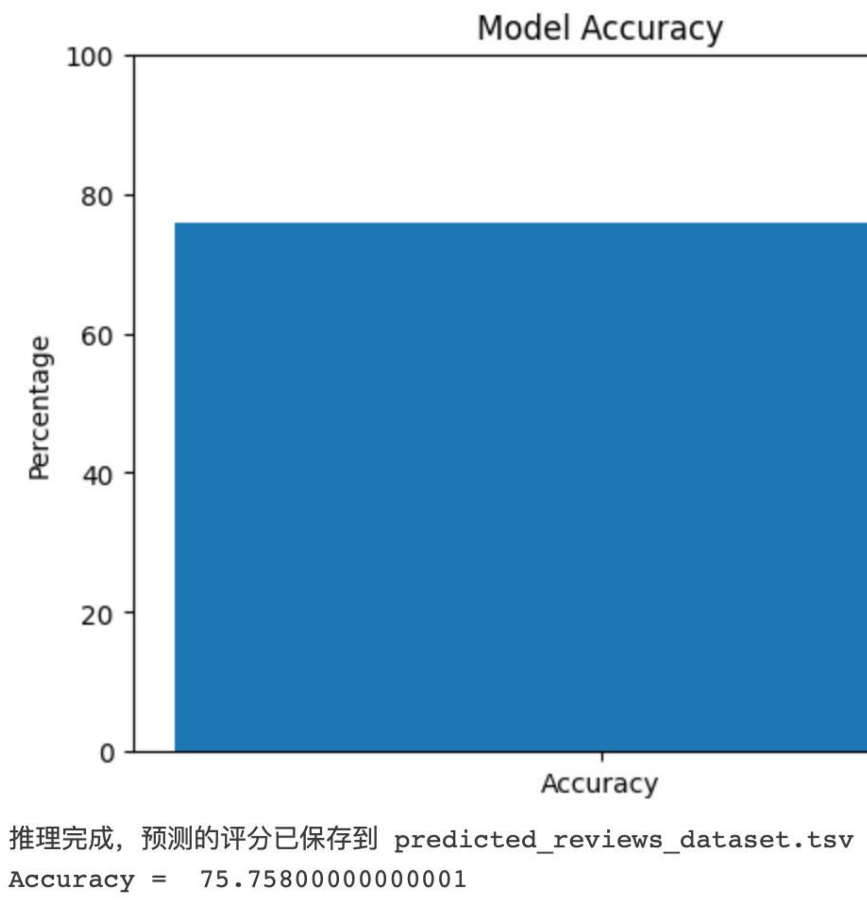
Stage 3: Reasoning stage. Through Stage 2, we have obtained the model we want, and in this stage we will use this model to predict the rating of new 50000 comment content.

(1) Determine the required cloud resources: The resources used in this stage are the same as those used in Stage 2.

(2) Code: The following is Pseudocode. Also, due to the limitation of VM performance, I set the number of new data to 50000.

```
IMPORT necessary libraries and modules
LOAD pre-trained BERT model
DEFINE new reviews dataset path and batch size

FUNCTION batch_inference:
    FOR each batch in new reviews dataset:
        EXTRACT batch reviews and convert to string list
        TOKENIZE text using tokenizer
        MAKE predictions using BERT model
        CONVERT predictions to ratings
        EXTEND predicted ratings list
    ENDFOR
END FUNCTION

CALL batch_inference function
ADD predicted ratings to new reviews dataset
SAVE new reviews dataset with predicted ratings to specified TSV file
CALCULATE accuracy by comparing predicted ratings with actual ratings
CREATE bar plot to visualize accuracy using matplotlib.pyplot
SHOW plot
PRINT inference completion message and accuracy
```

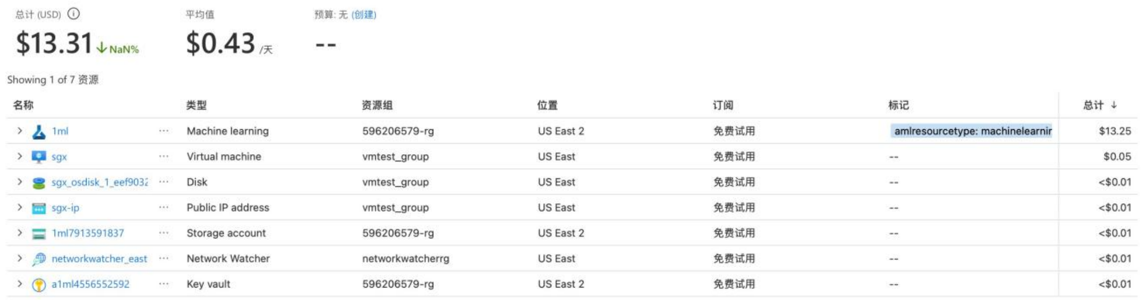
(3) Result: I successfully predicted new data using the above code. This stage takes a total of 7 ***hours*** and costs ***5 USD***. The accuracy of the final prediction is 75.758%, as shown in Figure 17.



```
推理完成，预测的评分已保存到 predicted_reviews_dataset.tsv
Accuracy =  75.75800000000001
```

Figure 17 Accuracy

Stage 4: Cost calculation. The single cloud experiment took a total of (*17+T)* ***hours*** and (*13+C)* ***USD***. As shown in Figure 18.



总计 (USD) $13.31 ↓NaN%　　平均值 $0.43 /天　　预算: 无 (创建) --

Showing 1 of 7 资源

名称	类型	资源组	位置	订阅	标记	总计 ↓
1ml	Machine learning	596206579-rg	US East 2	免费试用	amlresourcetype: machinelearnir	$13.25
sgx	Virtual machine	vmtest_group	US East	免费试用	--	$0.05
sgx_osdisk_1_eef9032	Disk	vmtest_group	US East	免费试用	--	<$0.01
sgx-ip	Public IP address	vmtest_group	US East	免费试用	--	<$0.01
1ml7913591837	Storage account	596206579-rg	US East 2	免费试用	--	<$0.01
networkwatcher_east	Network Watcher	networkwatcherrg	US East	免费试用	--	<$0.01
a1ml4556552592	Key vault	596206579-rg	US East 2	免费试用	--	<$0.01

Figure 18 Time And Cost

9.4 Sky Computing Experiment

Stage 1: The Stage 1 of the Sky Computing experiment is identical to the Stage 1 of the single cloud experiment. So this stage takes T hours and costs C USD.

Stage 2: At this stage, we will use Google's TPU VM on the Google Cloud Platform (GCP) to train our model.

(1) Account registration: Firstly, we need to register for GCP. It should be noted that registering for GCP also requires a credit card. After successful registration, GCP will give the new user 2340 USD as a gift, as shown in Figure 19.

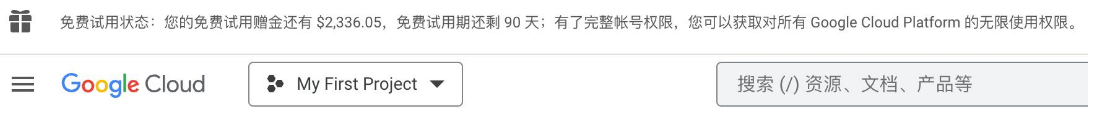
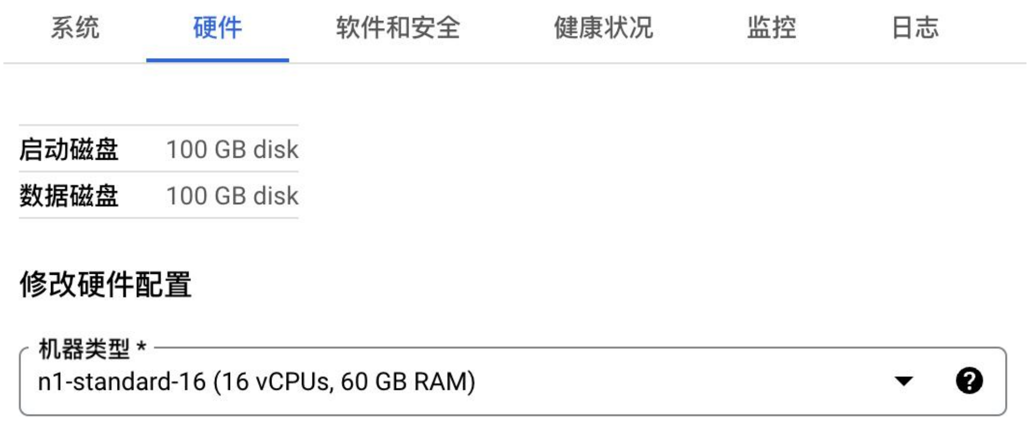

免费试用状态：您的免费试用赠金还有 $2,336.05，免费试用期还剩 90 天；有了完整帐号权限，您可以获取对所有 Google Cloud Platform 的无限使用权限。

Figure 19 Free quota

(2) Determine the cloud resources that need to be used: As mentioned earlier, we need to use GCP's TPU VM to train the model. However, due to the inability of the free account to use TPU resources and the pursuit of lower monetary costs, I chose to use this configured VM to train the model: n1-standard-16 (16 vCPUs, 60 GB RAM), with an estimated cost of 0.632 USD/hour. As shown in Figure 20.



启动磁盘	100 GB disk
数据磁盘	100 GB disk

修改硬件配置

机器类型 *: n1-standard-16 (16 vCPUs, 60 GB RAM)

Figure 20 The Configuration of virtual machines

(3) Code: The training code, dataset, and training set are the same as those used in the single cloud experiment, so they will not be repeated here.

(4) Result: The operation was successful. This stage takes a total of ***5.5 hours*** and costs ***4 USD***. The prediction accuracy of this model is 77.19%. As shown in Figure 21.

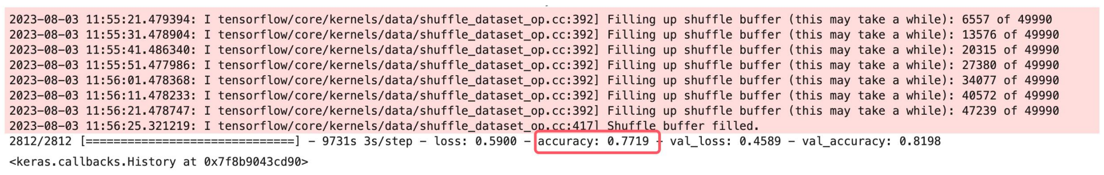

```
2023-08-03 11:55:21.479394: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 6557 of 49990
2023-08-03 11:55:31.478904: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 13576 of 49990
2023-08-03 11:55:41.486340: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 20315 of 49990
2023-08-03 11:55:51.477986: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 27380 of 49990
2023-08-03 11:56:01.478368: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 34077 of 49990
2023-08-03 11:56:11.478233: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 40572 of 49990
2023-08-03 11:56:21.478747: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:392] Filling up shuffle buffer (this may take a while): 47239 of 49990
2023-08-03 11:56:25.321219: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:417] Shuffle buffer filled.
2812/2812 [==============================] - 9731s 3s/step - loss: 0.5900 - accuracy: 0.7719 - val_loss: 0.4589 - val_accuracy: 0.8198
<keras.callbacks.History at 0x7f8b9043cd90>
```

Figure 21 The Running results

Stage 3: In this stage, we will conduct the inference phase of the ML Pipeline experiment on AWS, which is to use the model trained in Stage2 to predict the star rating of 50000 new customer comments.

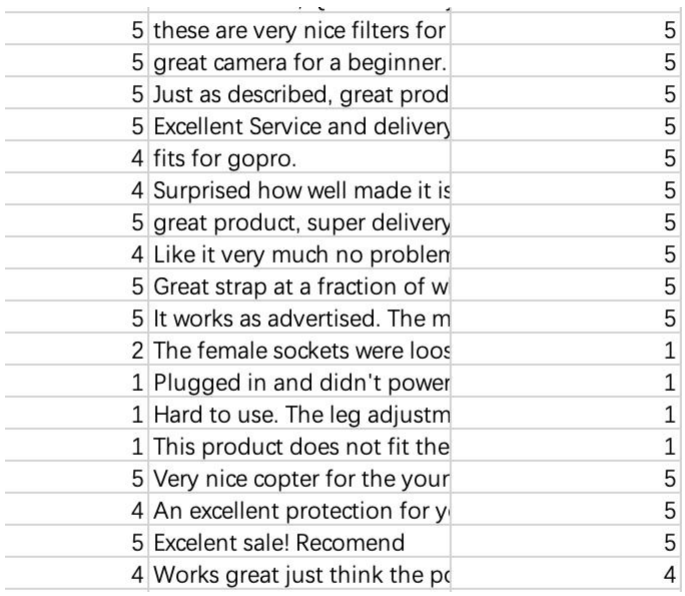
(1) Determine the required cloud resources: We will use the inference function in Amazon SageMaker to complete our Stage 3. Amazon SageMaker is a fully hosted

machine learning service that allows users to easily build, train, and deploy machine learning models. As shown in Figure 22.

Figure 22 Amazon SageMaker

(2) Code: This section is the same as Stage 2 in the single cloud experiment, so it will not be repeated here.

(3) Result: The experiment was successful. This stage takes a total of ***3 hours*** and costs ***2 USD***. The accuracy of the final prediction was 76.174%, and a dataset file with predicted star columns was output. As shown in Figure 23.



5	these are very nice filters for	5
5	great camera for a beginner.	5
5	Just as described, great prod	5
5	Excellent Service and delivery	5
4	fits for gopro.	5
4	Surprised how well made it is	5
5	great product, super delivery	5
4	Like it very much no problen	5
5	Great strap at a fraction of w	5
5	It works as advertised. The m	5
2	The female sockets were loos	1
1	Plugged in and didn't power	1
1	Hard to use. The leg adjustm	1
1	This product does not fit the	1
5	Very nice copter for the your	5
4	An excellent protection for y	5
5	Excelent sale! Recomend	5
4	Works great just think the pc	4

Figure 23 Predicted Star Rating

Stage 4: Cost calculation. This Sky Computing experiment took a total of ***(8.5+T) hours*** and ***(6+C) USD***.

9.5 Analysis of experimental results

Based on the experimental results, we can conclude that the single cloud experiment took (17+T) hours and cost (13+C) USD, while the Sky Computing experiment took (8.5+T) hours and cost (6+C) USD. The single cloud experiment costs 8.5 hours and 7 USD more than the Sky Computing experiment. Obviously, the Sky Computing experiment saves more time and cost than the single cloud experiment. As shown in Table 3:

Stage	Single Cloud Time(Hours)	Sky Computing Time(Hours)	Time Reduction	Single Cloud Cost(USD)	Sky Computing Cost(USD)	Cost Reduction
Data Processing	T	T		C	C	
Training	10	5.5	45%	8	4	50%
Inference	7	3	57.14%	5	2	60%
Total	17+T	8.5+T	50%	13+C	6+C	53.85%

Table 3 Comparison between Single Cloud Experiment and Sky Computing Experiment

Therefore, we can conclude that using the Intercloud broker service can reduce the time and financial cost of task execution, and the experimental replication is successful.

Through this replication experiment, if Sky Computing can be implemented, it will solve a huge pain point for users to use cloud services: the learning cost and usage threshold of using different cloud computing platforms. I think one of the most significant difficulties in this experiment is that I have to learn how different cloud service platforms are used and how to find the cloud services or resources I need from various cloud platforms. Because the names of the same cloud service may vary across different cloud platforms, for example, for machine learning, Azure refers to it as Azure Machine Learning, GCP refers to it as Vertex AI, AWS refers to it as Amazon SageMaker and their usage methods may vary. Therefore, whenever I switch platforms, I need to spend much time finding these services and learning how to use them. In addition, every time I switch platforms, I need to register a new account and bind it to my credit card, and check if there are any abnormal deductions on my credit card. This undoubtedly dramatically increases the learning cost and usage threshold for users. If the user is a beginner, then using a cloud platform is undoubtedly difficult.

Sky Computing does not have the above issues, as implementing the Intercloud Broker Service eliminates the need for users to register multiple cloud platform accounts or learn how different platforms are used. All of these tasks will be completed by Intercloud Broker, and users only need to clarify their needs with Intercloud Broker. And this is also the fundamental difference between Sky Computing and multi-cloud computing.

10. The Future of Intercloud Brokerage Service

With the increasing complexity and diversity of cloud services, users need a tool that can help them find the cloud service that best meets their needs among numerous services. And this is the value of Intercloud Brokerage Service. It can help users choose between cloud service providers and find the most suitable service. In addition, the Intercloud Brokerage Service can also help users manage and monitor the applications they deploy on different cloud services, making it easier for users to manage their cloud resources. Therefore, the future of Intercloud Brokerage Service is very promising. This section will analyze the future development trends of IBS from its advantages and challenges and provide my own suggestions for it.

10.1 Advantages

(1) Flexibility: As an intermediary for cloud services, Intercloud Brokerage Service can choose cloud services among multiple cloud providers, providing users with the most suitable cloud services for their needs. This flexibility allows users to select the optimal cloud service based on their own needs rather than being limited to a single cloud provider.

(2) Optimize resource utilization and cost: Through IBS, users can choose the most suitable cloud service based on their own needs and budget. For example, if a user requires a large amount of computing resources but does not require long-term storage, they can choose cloud services that provide a large amount of computing resources but have lower storage costs. This approach can help users optimize resource utilization and reduce costs.

(3) Provide unified management and monitoring: IBS can help users manage and monitor applications and resources on different cloud services in a unified manner. This allows users to manage their cloud resources more conveniently without needing separate management and monitoring on each cloud service.

(4) Improving the availability and reliability of cloud services: Through IBS, users can switch between multiple cloud services to cope with the failure of a single cloud service. For example, if a cloud service fails, users can quickly switch to another cloud service through IBS to ensure the continuous operation of the application. This approach can improve the availability and reliability of cloud services.

(5) Promoting competition and innovation in cloud services: IBS can promote competition among cloud service providers by providing a choice of multiple cloud service providers. This competition can drive cloud service providers to offer better services, lower prices and thus promote the development and innovation of cloud services.

10.2 Challenges

(1) Standardization: Currently, different cloud providers may use different technologies and standards, making it difficult to interoperate between different cloud services. Therefore, there is a need for a unified standard for interoperability between different cloud services. But at the same time, IBS can only advocate for some to follow this unified standard, as doing so may hinder technological innovation and progress.

(2) Security: Operating between multiple cloud services may pose security issues. For example, user data may be transferred between multiple cloud services, which may increase the risk of Data breach.

(3) Performance: Operating between multiple cloud services may affect the application's performance. For example, if an application requires data transmission between multiple cloud services, this may increase latency and affect the application's performance.

10.3 Suggestions

(1) Enhancing security: The security of cloud services is undoubtedly one of the most concerning issues for users. As an intermediary for cloud services, IBS needs to provide strong security guarantees to protect users' data and applications from attacks. For example, security features such as data encryption, access control, and auditing can be provided.

(2) Provide more cloud service options: Currently, Intercloud Brokerage Service is mainly focused on several major cloud service providers, but there are many other cloud service providers, such as IBM Cloud, Oracle Cloud, etc., who also provide excellent cloud services. IBS can consider increasing the support of these cloud service providers to provide users with more choices.

(3) Provide better performance optimization: The Intercloud Brokerage Service can provide users with the optimal choice of cloud services through optimization algorithms. For example, factors such as user application type, data location, and network conditions can be considered to select the most suitable cloud service.

(4) Providing better cloud service integration: Cloud service integration is a critical way to improve the efficiency of cloud service usage. Intercloud Broker Service can provide better cloud service integration, helping users integrate different cloud services together and achieve more complex applications. For example, computing, storage, database, and analysis services can be integrated to provide a one-stop cloud service solution.

11. Conclusion and Future Work

Through this independent project, I have gained a deeper understanding and understanding of the cutting-edge research field of Sky Computing. At the same time, I have also felt how important the Intercloud Brokerage Service is to Sky Computing. In my opinion, if Sky Computing is a ship, then the Intercloud Brokerage Service is the propeller of that ship. If Sky Computing loses the Intercloud Brokerage Service, So Sky Computing will permanently stagnate. At the same time, I replicated the experiments in the paper and successfully verified the optimization effect of Intercloud Brokerage Service on time and money costs. This independent project is full of challenges for me because it is my first time conducting such a research project.

Sky Computing is also a very cutting-edge and unfamiliar field for me. But in the end, I overcame all the difficulties and challenges to complete this project, and I believe it will benefit me for a lifetime.

In the future, I will continue to explore the field of Sky Computing. Due to the relatively short duration of this independent project, I could not further investigate the code implementation of the Intercloud Broker Service, which is a regrettable thing for me. Therefore, I will continue to explore it and try to implement some of its functions in code myself. Although the Sky Computing field has made significant progress in recent years, it is still quite far from truly transitioning from the cloud computing era to the Sky Computing era. I will continue to pay attention to and explore the Sky Computing field and contribute my own efforts to it.

12. References

[1] Ion Stoica and Scott Shenker. 2021. From cloud computing to sky computing. In Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS '21). Association for Computing Machinery, New York, NY, USA, 26 – 32. https://doi.org/10.1145/3458336.3465301

[2] Chasins, S., Cheung, A., Crooks, N., Ghodsi, A., Goldberg, K., Gonzalez, J. E., ... & Stoica, I. (2022). The sky above the clouds. arXiv preprint arXiv:2205.07147. https://arxiv.org/abs/2205.07147

[3] Yang, Z., Wu, Z., Luo, M., Chiang, W. L., Bhardwaj, R., Kwon, W., ... & Stoica, I. (2023). {SkyPilot}: An Intercloud Broker for Sky Computing. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23) (pp. 437-455). https://www.usenix.org/conference/nsdi23/presentation/yang-zongheng

[4] Keahey, Katarzyna, et al. "Sky computing." IEEE Internet Computing 13.5 (2009): 43-51. https://ieeexplore.ieee.org/abstract/document/5226615

[4] Monteiro, André, Cláudio Teixeira, and Joaquim Sousa Pinto. "Sky Computing: exploring the aggregated Cloud resources." Cluster Computing 20.1 (2017): 621-631. https://link.springer.com/article/10.1007/s10586-017-0727-5

13. Project Activity Log

The following is the activity log of this independent project:

14/06: I first met with Professor Wang Wei, the project manager, and introduced my ideas to him. I discussed possible project themes and techniques that may be used and established the project theme.

15/06: Search for information to gain a preliminary understanding of the basic concepts and historical development of Sky Computing and complete the proposal.

16/06: Submit Proposal and project application to management, waiting for approval.

17/06-25/06: The project application has been approved, and this independent project has officially started. I carefully studied the three papers recommended by Professor Wang Wei and delved into the relevant technologies and developments in the field of Sky Computing.

26/06-30/06: Starting to establish the experimental process and expected experimental results for this independent project.

01/07-04/07: The second and third meetings were held to discuss some issues during the experimental process of this project and establish the expected experimental results of this project.

05/07-12/07: Start the first phase of the experiment, the data processing phase, to understand the use of the Azure platform and solve the problems of account registration and Azure platform binding credit cards.

13/07-27/07: Completed the experiment's first phase and completed the final report's first seven chapters.

28/07: A fourth meeting was held to report on the completion progress of the project and discuss some issues encountered in the experiment.

29/07-05/08: Completed the remaining experiments and final project report.

06/08-07/08: All project materials were organized, the format of the final report was adjusted, and the content of the final report was checked for any issues.

08/08: Complete this independent project and submit relevant documents.

14. Meeting minutes

(1) First project meeting

Date: 14/06; Duration: approximately 30 minutes; Meeting content: The central theme of this project has been determined in this meeting. At the beginning of the meeting, I reported my situation and future career plans to Professor Wang and discussed my understanding of the development of cloud computing related technologies. In the end, Professor Wang recommended the research direction of Sky Computing to me, introduced relevant concepts in this field, and recommended some papers and projects related to this field. Finally, we have determined that the theme of this independent project is Sky Computing.

(2) Second project meeting

Date: 01/07; Duration: approximately 30 minutes; Meeting content: This meeting mainly revolves around the baseline of this independent project. As the theme of this project is a very cutting-edge direction, there is minimal reference available, and many concepts and technologies need to be searched for and learned by oneself. Therefore, establishing a baseline is a relatively difficult task. But through this meeting, I have a clear goal for the work direction of this independent project, and I have some ideas for establishing a baseline, which was verified after the meeting.

(3) Third project meeting

Recently: 04/07; Duration: approximately 40 minutes; Meeting content: The baseline of this independent project was determined in this meeting. After the discussion in the previous meeting, I conducted many research and experiments and finally decided on this project's baseline and expected results. I also reported at this meeting. At the same time, I also reported on my recent work content, including a summary of the papers provided by the professor and my ideas. I also compared these papers and finally raised some issues encountered in the experiment.

(4) Fourth project meeting

Date: 28/07; Duration: approximately 20 minutes; Meeting content: This meeting mainly reported on the completion progress and some problems encountered in this independent project. After more than a month's experiment, the project is nearly completed. I met many interesting problems during the experiment. I focused on the problem of Opaque, an Open-source software that needs to be used in the data processing phase. Because the development team of this software has basically stopped maintaining it, it cannot be used properly, which also prevents my experiment from continuing. Then I reported on my solution and some gains and presented the project's results so far.