AI-Enabled Inter cloud Broker for Autonomous Multi-Cloud Service Orchestration

Dr.Farheen Mohammed
Assistant Professor, Bapatla Engineering College, Bapatla, A.P, India

Abstract- To address growing governmental regulations concerning data locality and to mitigate risks from cloud service outages, users increasingly demand the flexibility to seamlessly migrate workloads across multiple cloud platforms. Rather than relying on broad and uniform standardization across providers, this paper introduces an alternative approach: the creation of a fine-grained, two-sided market **facilitated by an** intercloud broker**. Such a broker enables users to perceive the cloud environment not as a set of isolated and incompatible platforms, but as a cohesive and interoperable** "Sky of Computing." **The paper presents the** design and implementation **of an intercloud broker system called** SkyPilot**, evaluates its performance and advantages, and shares insights from its deployment in real-world scenarios.**

INTRODUCTION

Today's digital infrastructure is built upon three foundational pillars: the Internet, which ensures end-to-end network connectivity; cellular networks, which offer near-universal user access through increasingly advanced mobile devices; and cloud computing, which provides scalable and on-demand computational resources.

While these ecosystems may appear similar in function, they differ significantly in terms of interoperability among providers. Both the Internet and cellular networks were designed from the ground up to support universal accessibility. Achieving this required the establishment of comprehensive industry standards and widespread adoption of interconnection agreements—such as Internet peering and cellular roaming—resulting in globally federated systems composed of competitive but cooperative providers.

In contrast, cloud computing evolved differently. Originating as a more flexible alternative to dedicated, on-premises data centers, the cloud ecosystem did not prioritize interconnectivity. Instead, early cloud providers focused on differentiating their platforms. Although most cloud platforms rely on common abstractions like virtual machines (VMs), containers, and function-as-a-service (FaaS), their orchestration interfaces—used to manage these services—were initially highly fragmented. Over time, however, these interfaces have begun to converge, becoming more aligned across providers.

INTER CLOUD BROKER SETUP

Catalog. The catalog records the instances and services available in each cloud, detailed locations that offer them, and the APIs to allocate, shut down, and access them. It also stores the long-term prices for on-demand VMs, data storage, egress, and services (typically these prices do not change for months).
The catalog can provide filtering and searching functionalities. The catalog can be based on information published by the clouds, listed by a third party, or collected by the broker.

Tracker. This component tracks spot prices (which can change more frequently, e.g., hourly or daily) as well as resource availability across clouds and their locations.

Optimizer. The optimizer takes as inputs (1) the application's DAG and its requirements, and (2) the instance and service availability as well as their prices provided by the catalog and tracker, and then computes an optimal placement of the tasks. Upon resource availability and price changes, the optimizer may perform re-optimization.

Provisioner. This component manages resources by allocating the resources required to run the execution plan provided by the optimizer, and freeing them when each task exits. To handle unpredictable capacity and user quota errors, the provisioner implements

automatic failover, where it asks the optimizer for a new placement plan if the provision fails. Failures are also reported to the tracker.

Executor. The executor manages the application by packaging each application's tasks
and running them on the resources allocated by the provisioner.
In the future, we imagine intercloud brokers will offer more sophisticated services such as troubleshooting across clouds, providing more detailed performance measurements for specific applications on each cloud, the equivalent of spot-pricing but across clouds, reselling services at lower than listed prices (similar to the travel industry), and advanced configuration features for security and/or networking.
Furthermore, we expect a commercial broker to provide billing support to enable a user to have a single account with the provider of the intercloud broker, which then pays for the services rendered by each cloud on behalf of the user and charges the user back. In our current deployment, our users have direct accounts with the three major clouds, so this functionality is not needed.

AI ARCHITECTURE

Our architecture focuses on how we can accelerate the practical use of sky computing by integrating AI with the intercloud broker available in the sky computing architecture.
We go through different stages in our architecture to achieve it which include data collection and pre-processing, Model development, Deploying the model and integrating it with the intercloud broker and confirm if the broker is taking the optimal solution. The goal is to ensure that the intercloud broker autonomously makes optimal decisions by considering factors such as real-time traffic analysis, adaptive load balancing, and resource optimization for various services across multiple clouds.

Data pre-processing:
At first we collect the data regarding the network traffic (data flow, bandwidth, routing information across different clouds), Resource usage data (data regarding CPU, memory and storage utilization) and Performance metrics (Latency, throughput and system uptime of multiple services throughout different cloud environments).
Now first we analyse the data for network traffic using different techniques like principal component analysis (PCA) and generative adversarial networks (GANs) to identify the traffic patterns allowing faster processing and creating custom traffic patterns so our ai can learn and predict future traffic patterns respectively. Next we analyze data concerning resource usage by the means of federated learning through which our model can learn about resource usage of services of different clouds locally without the need to transfer data to different clouds ensuring data privacy. At last we analyze data with respect to performance metrics by normalizing, aggregating and comparing the latency and throughput of different services from multiple clouds due to factors like accessibility of availability zone with regards to the specific services. Here we also use technique of auto encoders which briefs the performance metric data, identifies anomalies and removes the noise from data.

MODEL DEVELOPMENT

We train the data of traffic analysis by using algorithms like convolutional recurrent neural networks (CRNNs) and Graph neural networks (GNNs). CRNNs can record both spatial and temporal patterns in traffic data. RNNs are used to understand and predict the future traffic on different routes provided by different services and identify the service providing least latency and CNNs are used for recognizing patterns in data packets. GNNs are used in sky computing where each service of different cloud providers are represented as nodes and communication paths as edges and our model can understand the best route by analysing this graph allowing better routing decisions and avoiding any congestion.

For load balancing and resource optimization we prefer using multi-agent learning where every cloud is an agent which optimize load balancing for particular service and compare with agents for similar services provided by them, comparing the number of availability zones and resources like memory and storage used the best service is identified. Algorithms used here are Deep Q-learning used by agents to improve performance locally and proximal policy optimization to explore techniques that uses efficient

resources without negatively affecting the overall system. Hierarchical reinforcement learning is very helpful in large distributed system like sky computing through which our model can assign specific tasks to individual cloud instances based on real-time resource availability. HRL leads to optimized resource allocation and task scheduling.

INTEGRATION OF AI IN INTER CLOUD BROKER

The information present in catalog about available instances, API's and prices for all services across various cloud platforms is pre-processed thoroughly using the techniques mentioned in previous section.After all the data related to availibility of resources like cpu, memory storage, API's and best prices among similar services across clouds, the data is sent to optimizer which will plan the resource allocation for a task it takes input as a dag from the customer. Optimizer takes decision of choosing service from the perticular cloud based on the best prices available, providing the service with low latency. We integrate our model training step in optimizer as we train the pre-processed data received from catalog.

Here we apply different algorithms based on which factor we are applying to as discussed in the previous section. By taking the results of the data trained the optimizer makes accurate prediction and chooses the most optimized services considering low latency and best price. The plan created with the help of AI algorithms is sent to the provisioner which allocates the resources as per the plan put forward by optimizer. Now the executor packages the applications tasks and completes the execution based on the allotted instances and resources like memory by the provisioner.

CONCLUSION

This paper describes the design, implementation, applications, and early deployment of an intercloud broker, SkyPilot. SkyPilot enables users to seamlessly run their batch jobs across clouds to minimize cost and/or delay. We see this as the first step towards a paradigm we call Sky Computing, which we hope will transform the cloud computing ecosystem to better meet user needs. Acknowledgements. We thank the NSDI reviewers and our shepherd, Paolo Costa, for their valuable feedback. This work is in part supported by NSF CISE Expeditions Award CCF1730628 and gifts from Astronomer, Google, IBM, Intel, Lacework, Microsoft, Nexla, Samsung SDS, Uber, and VMware.

REFERENCES

[1] Akka. https://akka.io/.
[2] Amazon customer reviews dataset. https://s3.amazonaws.com/amazon-reviews-pds/readme.html.
[3] Amazon Elastic Kubernetes Service. https://aws.amazon.com/eks/.
[4] Amazon EMR. https://aws.amazon.com/emr/.
[5] Anthos. https://cloud.google.com/anthos.
[6] Apache Airflow. https://airflow.apache.org/.
[7] Apache Cassandra. https://cassandra.apache.org/.
[8] Apache jclouds. https://jclouds.apache.org/.
[9] Apache Kafka. https://kafka.apache.org/.
[10] Apache Libcloud. https://libcloud.apache.org/.
[11] Application versions in Amazon EMR 6.x releases.https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-app-versions-6.x.html.
[12] Artificial Intelligence: From the Public Cloud to the Device Edge. https://www.equinix.com/resources/whitepapers/nvidia-distributed-aicloud-infrastructure-edge.
[13] AWS and Arm. https://www.arm.com/why-arm/partnerecosystem/aws.
[14] AWS Graviton Processor. https://aws.amazon.com/ec2/ graviton/.
[15] AWS Inferentia. https://aws.amazon.com/machinelearning/inferentia/.
[16] Azure confidential computing. https://azure.microsoft.com/en-us/solutions/confidentialcompute/.
[17] Azure HDInsight. https://azure.microsoft.com /enus/services/hdinsight/.
[18] Azure Kubernetes Service. https://azure.microsoft.com/en-us/services/Kubernetes service/.
[19] Bandwidth Alliance. https://www.cloudflare.com/ bandwidth-alliance/.
[20] BlobFuse - A Microsoft supported Azure Storage FUSE driver. https://github.com/Azure/azure-storagefuse.

[21] Carbon free energy for Google Cloud regions. https://cloud.google.com/sustainability/region-carbon.
[22] Cerebras. https://cerebras.net/.
[23] Cloud TPU. https://cloud.google.com/tpu.
[24] Cloudflare R2. https://www.cloudflare.com /products/ r2/.
[25] Common Object Request Broker Architecture (CORBA). https://www.omg.org/spec/CORBA.