Master Thesis

Cost Optimization in Hybrid Multi-Cloud Architectures: A Study of Cloud Pricing Models, Sky Computing, and API-Driven Cost Control

Submitted by :	Shiva Kumar Biru
First examiner:	Prof. Henry-Nobert Cocos, M.Sc.
Second examiner:	Prof. Dr. Christian Baun
Date of start:	26.02.2025
Date of submission:	27.07.2025

Statement

I confirm that I, Shiva kumar Biru have written this thesis on my own. All sources used are duly referenced, and any content taken directly or indirectly from published or unpublished sources is properly cited. The figures and drawings included are my own creations or are appropriately referenced. Furthermore, I affirm that this work has not been submitted in the same or a similar form to any other examination board.

27-07-2025

Date, signature of the student

Acknowledgement

I would like to express my deepest gratitude to Professor Henry-Nobert Cocos, for his invaluable guidance, insightful feedback, and unwavering support throughout this project. His expertise and encouragement have been instrumental in shaping my ideas and bringing this project to reality. From the early conceptual stages to the final implementation, his mentorship has been a cornerstone of this thesis. I am especially thankful for his patience, suggestions. This work would not have reached its current form without his constant encouragement and supervision. I would also like to thank Prof. Dr. Christian Baun, my second examiner, for being part of the evaluation process and for his time and consideration.

ABSTRACT

In today's business world, companies are continuously seeking methods to reduce costs. Cloud computing has emerged as a valuable solution, particularly for small and medium-sized organizations (SMEs), because it provides flexible and scalable IT resources. However, as companies increasingly adopt hybrid and multi-cloud strategies using services from multiple cloud providers managing and optimizing costs across these platforms has become more complex. This thesis explores how various cloud pricing models, cost optimization techniques, cost control APIs, and the concept of Sky Computing can support cost efficiency in such environments.

The study begins with an in-depth comparison of pricing models from Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), focusing on compute instance pricing structures. It then investigates best practices for cost optimization and provides an overview of the Cost control APIs available from each provider to support real-time pricing access and visibility.

The research introduces a prototype that automatically retrieves service pricing data from cloud provider APIs. This prototype serves as a proof of concept for implementing cost analysis tools that can operate across cloud providers.

Table of Contents

Abbreviations

AWS	Amazon Web Services
GCP	Google Cloud Platform
API	Application Programming Interface
GUI	Graphical User Interface
VCI	Virtual Compute Instance
VM	Virtual Machine
SLR	Systematic Literature Review
PICOC	Population, Intervention, Comparison, Outcome, Context
QA	Quality Assessment
RI	Reserved Instance
CUD	Committed Use Discount
EC2	Elastic Compute Cloud
SKU	Stock Keeping Unit
ML	Machine Learning
KVM	Keyboard, Video, Mouse
PDU	Power Distribution Unit
CLI	Command Line Interface
DX	Direct Expansion
ALB	Application Load Balancer
SDK	Software Development Kit

List of Figures

List of Tables

List of Listings

1 Introduction

This chapter introduces the thesis by highlighting the importance of cost-effective and scalable cloud strategies in today's technology environment. It explains the motivation behind the study, emphasizing the complexity organizations face in managing cloud expenditures across hybrid and multi-cloud environments. The chapter outlines the research questions that guide the investigation, explains the methodology applied, and provides a roadmap for the chapters that follow.

1.1 Motivation

Moving to the cloud isn't just about setting up virtual machines or launching applications. For many organizations, the real challenge is ensuring their cloud setup is reliable, scalable, and cost-effective. As cloud usage increases, companies are choosing hybrid and multi-cloud setups. They combine on-premises systems with services from major providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). This combination helps improve flexibility, reduce dependence on a single vendor, and utilize various cloud features more effectively.

Cloud computing is considered a straightforward and scalable alternative to traditional IT infrastructure. However, as cloud usage grows, cost management becomes increasingly complex. Each cloud provider offers various pricing options, such as On-Demand, Reserved Instances, or Spot pricing. Each option has its own terms, discount plans, and billing practices. It is crucial to select the right model for each workload to avoid overspending and to maximize value.

Cost management becomes even more challenging when resources are spread across multiple platforms. In addition to compute and storage pricing, organizations face hidden costs, such as data ingress and egress fees, cross-region transfers, and unexpected charges for API calls or idle resources. Companies using hybrid architectures must also consider ongoing expenses related to on-premise infrastructure. These include power, cooling, hardware depreciation, and maintenance. Although these costs often go unnoticed, they can have a significant impact on the overall IT budget.

This is where cost optimization strategies, careful planning, and the use of automated tools and Cost control APIs become essential . These tools help track usage, analyze spending patterns, and implement cost-saving measures effectively across different cloud providers.

Sky Computing is gaining increasing attention as it aims to break down the barriers between cloud providers. This model enables running workloads on the most cost-efficient platform at any time, independent of the cloud provider. While this enhances cost-efficiency and flexibility, it also introduces new complexities in orchestration and control.

This thesis analyses these challenges in detail. It examines the pricing models of AWS, Azure, and GCP. It explores how resources are allocated efficiently in real-world architectures. It assesses

tools and APIs available for controlling cloud costs. It also considers how newer technologies such as Sky Computing, contribute to cost optimization.

The study incorporates these insights into the design and development of a working prototype. It aims to help organizations make smarter, more cost-aware decisions when using the cloud. The prototype allows users to compare costs across AWS, Azure, and GCP. This helps them understand pricing differences and choose the best options for their workloads. This approach not only strengthens the research but also provides practical value for cloud planning and cost optimization in the real world.

1.2 Goal of this Thesis

The thesis applies a Systematic Literature Review (SLR), which involves the collection and analysis of peer-reviewed articles, technical documentation, and whitepapers related to cloud pricing models and cost optimization techniques. The review follows a structured protocol, which includes the PICOC framework, a set of well-defined research questions, detailed search strategies, and inclusion and exclusion criteria, as well as a data extraction and data synthesis process. Additionally, the thesis provides a technical overview of cost control APIs, derived from the official documentation of each cloud provider. Furthermore, it presents an exploratory review of recent research and position papers on Sky Computing, reflecting its growing relevance in cloud architecture and cost optimization.

Through this process, the study aims to understand existing pricing models, practical cost control measures, and emerging possibilities for cost-aware infrastructure planning. To guide the investigation, the following research questions are formulated:

RQ1: What pricing models are available and what hidden costs are involved?
RQ2: How can cloud tools and APIs help automate and control costs?
RQ3: How can Sky Computing influence cloud cost strategies?

By answering these questions, the thesis aims to provide a comprehensive view of current practices, identify gaps and challenges, and offer insights. These insights are further supported by a prototype tool that enables users to compare pricing across AWS, Azure, and GCP, allowing for more informed decisions in cloud infrastructure planning and budgeting.

To ensure a focused and manageable scope, this thesis limits its analysis to compute resources, specifically virtual machine offerings such as AWS EC2, Azure Virtual Machines, and Google Compute Engine. While cloud platforms offer a broad range of services including storage, networking, and managed databases, these are excluded from the study. Compute services are chosen as the focus because they represent a significant portion of cloud expenditures and form the basis of most cloud-based workloads.

1.3 Outline

This chapter presents the motivation for the study, the research need, and the overall goal. Chapter 2 provides an overview of cloud computing, followed by the evolution toward hybrid and multi-

cloud architectures. It discusses the limitations encountered at each stage and introduces Sky Computing as a response to cross-cloud cost and control challenges. Chapter 3 outlines the methodology, focusing on the Systematic Literature Review (SLR) approach. It explains the PICOC framework, search strategy, and data synthesis methods used in the study. Chapter 4 presents an analysis and discussion of selected literature, evaluating cloud pricing models, cost optimization practices, and identifying gaps in the existing literature. Chapter 5 reviews related work addressing the gaps identified in the literature analysis

Chapter 6 provides a technical overview of cost control APIs offered by the three major cloud platforms, AWS, Azure, and GCP. Chapter 7 introduces the concept of Sky Computing and its potential impact on cost optimization. Chapter 8 describes the implementation of a prototype tool for comparing cloud costs across AWS, Azure, and GCP, and demonstrates its application in practical scenarios. Finally, Chapter 9 concludes the thesis with a summary of key findings, and Chapter 10 discusses directions for future research.

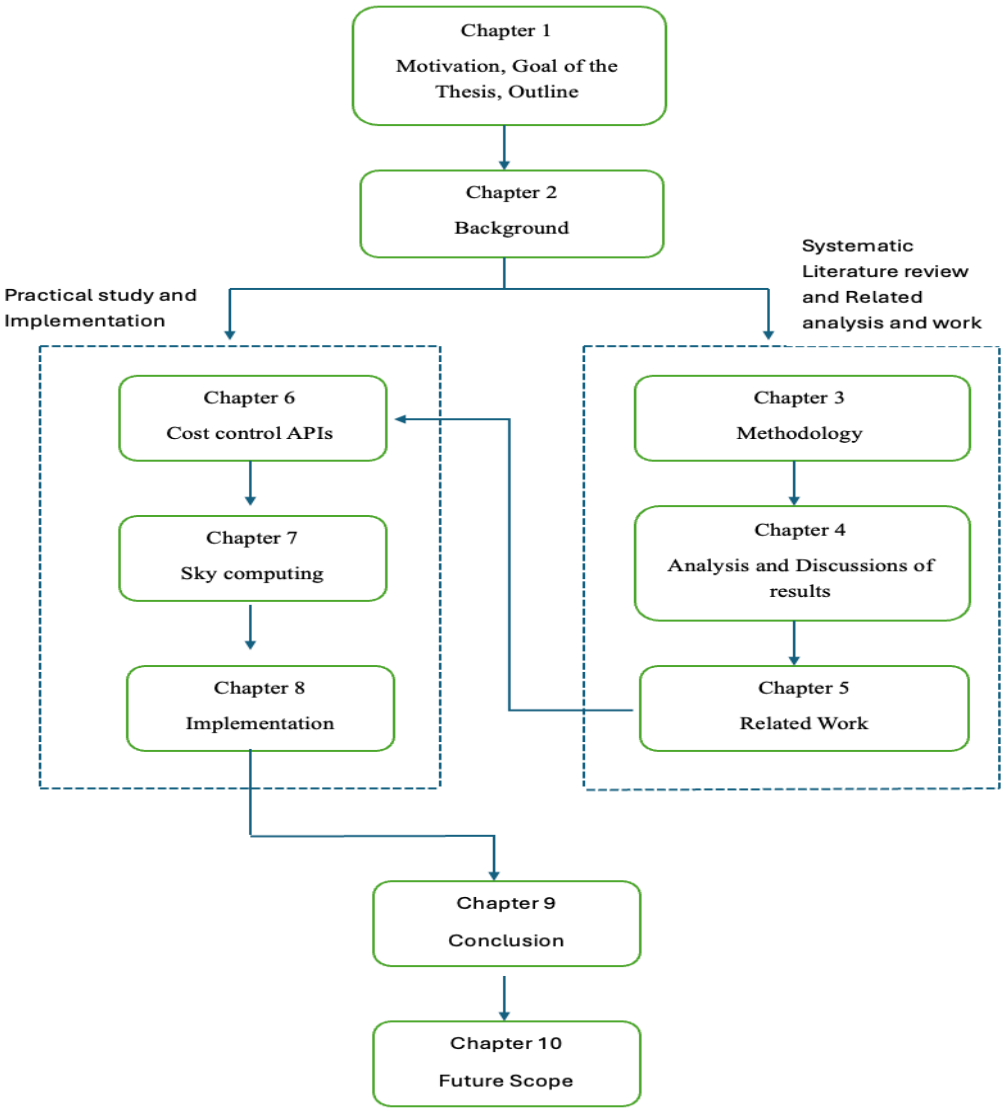

Figure 1.1 : Thesis organization flowchart

2 Background

To clearly define the focus of this study, this section begins by explaining the fundamental concepts of cloud computing, hybrid cloud, multi-cloud, and the emerging idea of Sky Computing. This section outlines their evolution and the challenges they introduce.

2.1 Cloud Computing Overview

Cloud computing emerges as a transformative paradigm in modern computing, fundamentally reshaping how organizations provision and manage IT resources. The shift from traditional on-premises infrastructure to cloud computing is driven by the limitations and challenges associated with maintaining on-prem data centers, particularly the high capital expenditure(CapEx), underutilization of resources, and the complexity of scaling hardware and software components to meet dynamic business demands [1].

Organizations with on-premises systems often struggle to achieve optimal resource utilization, despite investing a significant amount in IT infrastructure. The static nature of such setups leads to over-provisioning or under-provisioning of resources, coupled with delayed responsiveness to evolving business requirements. This inefficiency, along with mounting maintenance costs and the demand for continuous technological upgrades, compels businesses to seek alternative models that offer scalability, agility, and cost-efficiency [1], [2].

Cloud computing helps tackle these challenges by offering instant network access to a shared pool of customizable computing resources such as servers, storage, applications, and services that can be quickly allocated or released with minimal administrative effort [1]. According to the National Institute of Standards and Technology (NIST), cloud computing is defined by five key features: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [1]. These features enable organizations to transition from fixed-capacity systems to scalable environments that can adapt to changing workload needs.

Elasticity is a key benefit of cloud computing, enabling resources to scale dynamically in response to demand. Unlike traditional systems that require overprovisioning, cloud platforms offer pay-as-you-go flexibility, reducing costs and avoiding idle capacity. This responsiveness also prevents service disruptions during usage spikes, as seen when Animoto scaled from 50 to 3,500 servers in three days [3].

Beyond cost savings, elasticity shifts infrastructure risk to the cloud provider, freeing organizations from fixed investments and enabling them to benefit from continuous improvements in pricing and performance [3].

Furthermore, cloud services can be tailored via three primary models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), each catering to different layers of the IT stack [1], [2]. Deployment can be tailored based on organizational needs using private, public, hybrid, or community cloud models.

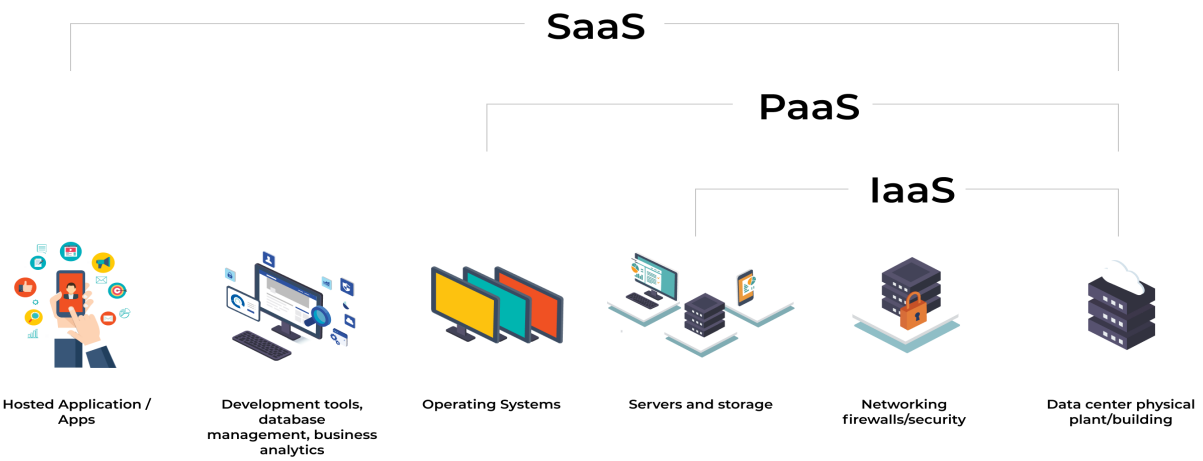

Figure 2.1: IaaS, PaaS, and SaaS each manage different elements of cloud computing

Source: [4]

The financial benefits of moving to the cloud are significant. By shifting from capital expenditure (CapEx)- heavy investments to an operational expenditure (OpEx) model, enterprises can reduce upfront costs and pay only for the resources they use [2], [5]. Additionally, cloud computing enhances reliability, disaster recovery capabilities, and service availability, which are crucial in today's data-driven environments [2], [6].

In summary, the shift from on-premises IT to cloud computing is driven by the need for flexibility, cost savings, and scalable performance.

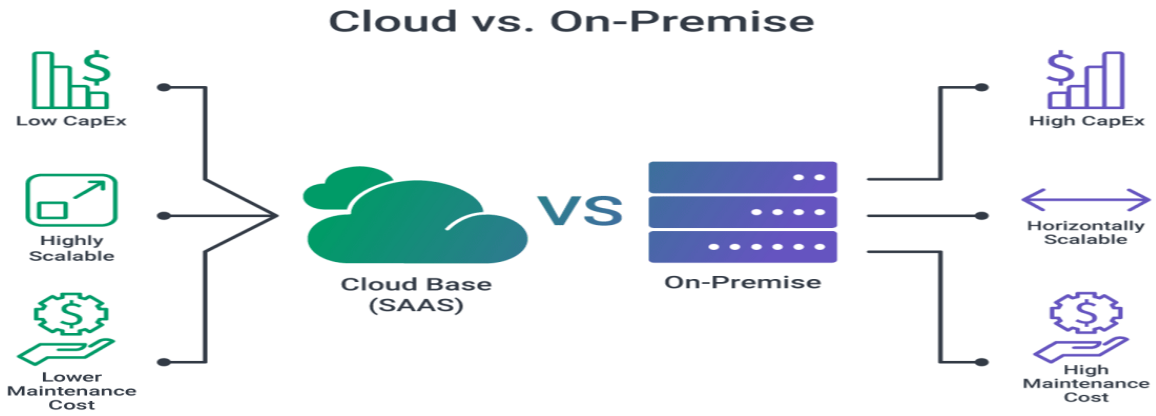

Figure 2.2: Visual Comparison of On-Premise and Cloud Infrastructure

Source: [7]

2.2 Hybrid Cloud Computing

Cloud computing enables organizations to utilize IT resources, such as servers and storage, over the internet, offering flexibility and cost savings. However, many businesses face challenges such as vendor lock-in, hidden costs, and issues with data privacy and regulatory compliance. These challenges lead to the rise of hybrid cloud computing, which combines both public and private Infrastructure models to offer a more flexible and secure solution.

In a hybrid cloud setup, critical and sensitive workloads are maintained on private infrastructure, while less sensitive tasks utilize the public cloud. This provides organizations with the best of both worlds: scalability and low cost from the public cloud, combined with control and security from the private infrastructure . As Azumah et al. [8] note, a hybrid cloud is often adopted to balance cost, performance, and data protection, especially when public or private clouds alone cannot meet all needs.

For example, large companies like Facebook shift some services from public clouds back to private data centers to better control costs and performance [8]. A hybrid cloud enables this by allowing data and applications to move between environments as needed.

Gorantla et al. [9] explain that a hybrid cloud is especially useful in industries like e-commerce, where there are high data demands and strict regulations. It helps manage traffic spikes, protect customer data, and meet compliance requirements.

Studies also show that public clouds may not always be reliable or compliant enough for every business use case. Babu et al. [8] and Huang et al. [9] describe how hybrid cloud helps solve these limitations by allowing customized deployment and better monitoring. Azumah et al. [8] further confirm that many companies opt for hybrid setups to address performance issues and mitigate growing infrastructure costs.

Zhou et al. [12] also demonstrate that a hybrid cloud can reduce costs while maintaining good performance by enabling businesses to allocate workloads between public and private clouds based on location and requirements.

In summary, hybrid cloud computing offers a flexible approach that enables organizations to remain agile, reduce risk, and meet evolving IT demands. It addresses the weaknesses of using only public or private cloud and supports a wide range of business needs.

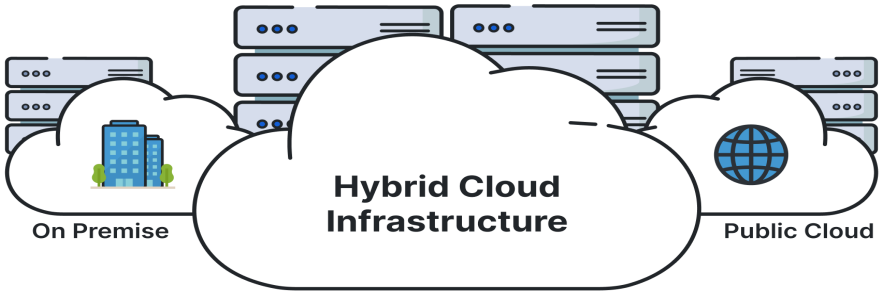

Figure 2.3: Representation of Hybrid Cloud Model

Source: [13]

2.3 Multi-Cloud Computing

Traditionally, many organizations adopt a single-cloud or hybrid cloud architecture due to ease of integration, simplified vendor management, and reduced initial complexity. However, over time, real-world operational challenges with these models prompt a shift towards multi-cloud computing, an approach where services from more than one cloud provider are used. This shift is driven by the need to overcome critical limitations in existing architectures.

One primary concern with single-cloud environments is vendor lock-in. As discussed by Pellegrini et al. [14], many cloud service providers (CSPs) rely on proprietary APIs and data formats, making application and data migration difficult and costly. Even in hybrid cloud environments, where some flexibility exists, organizations often remain dependent on a single cloud provider or their internal infrastructure, which limits portability and increases long-term operational risk.

Another key limitation is the vulnerability of centralized architectures in disaster recovery (DR) scenarios. Alshammari et al. [15] highlight that a major outage affecting a CSP's data center can lead to substantial downtime or data loss. While hybrid clouds may incorporate on-premise backups, they often lack cross-provider redundancy, which is important for maintaining service continuity in the face of natural disasters or cyber-attacks.

Cost efficiency and service-level agreement (SLA) fulfilment also pose challenges in single or hybrid cloud setups. As Bellur et al. [16] point out, relying on a single provider restricts organizations from leveraging diverse pricing models and performance optimizations available across the cloud ecosystem. Applications with strict latency requirements, such as video streaming or online gaming, may suffer if resources are not optimally located near end-users. The inability to dynamically shift workloads to better-performing or lower-cost providers limits both cost savings and user experience.

Multi-cloud computing directly addresses these limitations. It is defined as the use of multiple cloud computing services within a single, heterogeneous architecture. Each cloud may come from a different provider and serve other parts of an application or distinct workloads. This distributed setup enhances architectural flexibility, operational resilience, and performance efficiency.

By spreading workloads across multiple CSPs, multi-cloud architectures significantly reduce the risks associated with vendor lock-in. Pellegrini et al. [14] suggest that containerization technologies, such as Docker, and orchestration platforms, like Kubernetes, can abstract the underlying infrastructure, enabling provider-agnostic deployment of applications and services. This abstraction improves portability and reduces switching costs.

Multi-cloud environments also strengthen disaster recovery by replicating services and data across geographically diverse and independently operated infrastructures. Alshammari et al. [15] note that this design ensures continuity even if one provider fails, as services can be rerouted to others without compromising uptime.

In terms of cost and SLA management, multi-cloud systems enable intelligent workload placement based on cost efficiency, latency, and performance metrics. Bellur et al.[16] Develop a deployment strategy that considers multiple pricing models and resource constraints, achieving up to 22% savings in Total Infrastructure Cost (TIC) compared to single-provider deployments. Such strategies also help ensure SLA compliance by minimizing latency through geolocation-aware deployment.

In summary, the move to multi cloud computing represents a strategic evolution in enterprise cloud adoption. By addressing the challenges of earlier cloud models, such as vendor lock-in, limited backup options, and high costs, multi-cloud offers a flexible, reliable, and cost-effective solution for modern digital infrastructure.

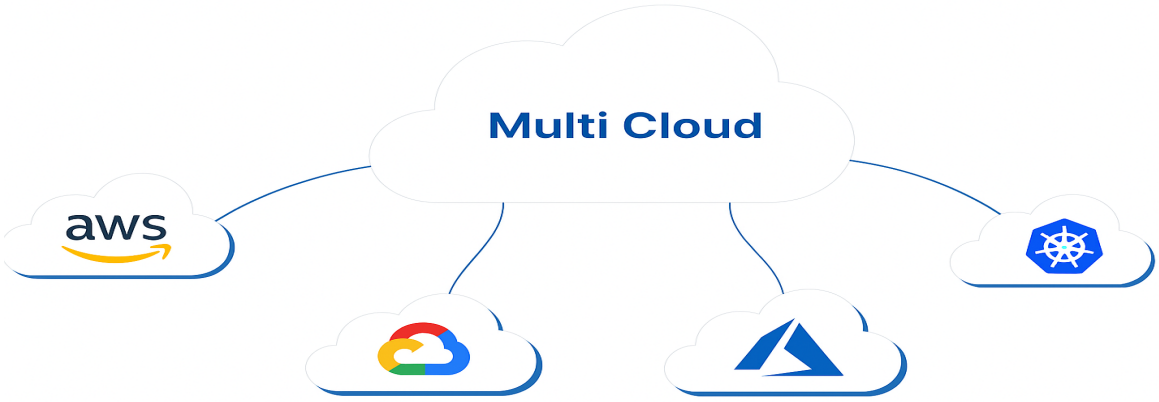

Figure 2.4: Representation of Multi-Cloud Mode

Source: [17]

2.4 Sky Computing

While the multi-cloud approach offers benefits such as flexibility and cost optimization, it also presents several practical challenges that organizations continue to struggle with. Managing resources across different cloud providers can become complex due to differences in APIs, security models, and deployment tools, resulting in inefficiencies and increased operational effort. Monitoring becomes particularly difficult as most tools are provider-specific and don't offer a unified view, making it hard to detect issues or maintain high availability in real-time. Additionally, the shortage of skilled professionals who can effectively manage these diverse environments adds another layer of difficulty. These ongoing issues prompt researchers and practitioners to explore new paradigms that can simplify and standardize cloud usage, paving the way for the emergence of Sky Computing as a more seamless alternative [18].

Sky computing is a proposed evolution of cloud computing that focuses on creating a unified and interoperable environment across multiple cloud providers. The goal is to enable applications to run seamlessly across different clouds without requiring significant modifications, thereby simplifying multi-cloud usage for developers and organizations.

Current cloud platforms often differ in their interfaces, services, and pricing models. These differences make it difficult to move applications or data between providers. Sky computing aims to address these challenges. The vision of Sky computing is inspired by the development of the Internet, where different networks are connected through standard protocols and mutual agreements. Similarly, Sky computing aims to make cloud services more accessible, flexible, and efficient by enabling collaboration among different cloud platforms [19].

It is managed through an intercloud broker. Instead of users directly interacting with a specific cloud provider (such as AWS, or Azure, or GCP), they send their job and its requirements to this broker. The broker then selects the most suitable cloud platforms to run various aspects of the job and oversees the entire process from start to finish. Think of it as a smart middle layer that enhances cloud usage by making it more flexible, efficient, and automated.

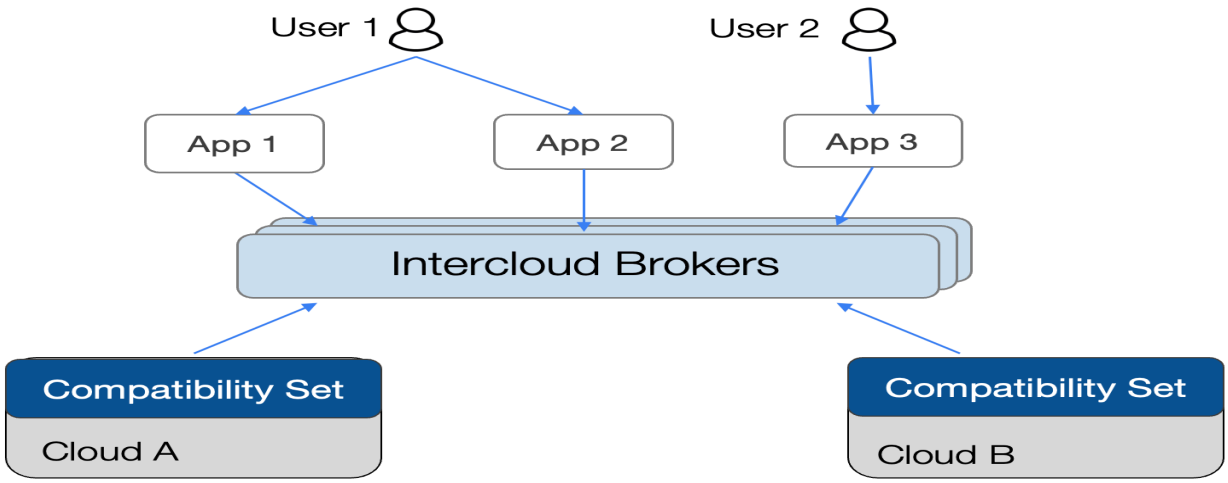

Figure 2.5: Sky computing intercloud broker

3 Methodology

This chapter describes the steps followed to carry out the Systematic Literature Review (SLR) for this thesis. The research questions addressed, the search strategy for relevant studies, and the main components of the review process are presented. The inclusion and exclusion criteria, along with the data collection and analysis methods, are also described.

3.1 Conducting a Systematic Literature Review

This thesis adopts the methodology outlined by Carrera-Rivera et al. [21], which provides a structured approach to identifying, selecting, assessing, and synthesizing relevant research. The methodology includes key phases such as planning the review, defining research questions using the PICOC framework, executing a structured search strategy, applying inclusion/exclusion and quality criteria, extracting data, and synthesizing the findings to answer predefined research questions.

3.1.1 Planning the Review

The Initial phase of the SLR involves developing a detailed review protocol. This protocol defines the procedures, tools, and criteria used throughout the review process. It serves as a foundation for ensuring transparency, consistency, and replicability.

Key steps in the planning phase include:

- **Definition of PICOC and Synonyms**: The research questions are broken down using the PICOC framework (Population, Intervention, Comparison, Outcome, Context). Relevant keywords and their synonyms are used to construct effective search queries.
- **Formulation of Research Questions**: Research questions are designed to address the objectives of the study and are aligned with the components of PICOC.
- **Selection of Digital Libraries**: The study utilizes digital databases such as IEEE Xplore, ACM Digital Library to ensure coverage of high quality literature in the domain.
- **Definition of Inclusion and Exclusion Criteria**: Articles are filtered based on publication date, language, type of source, accessibility, and relevance to the research questions.
- **Quality Assessment (QA) Checklist**: A checklist-based scoring system is implemented to assess the credibility and rigor of each selected study.
- **Design of the Data Extraction Form**: A structured form is developed to capture essential information from each study, including research type, authorship, publication year, identified research gaps, and key findings.

3.1.2 Conducting the Review

With the protocol in place, the review process begins by constructing search strings based on PICOC elements and their synonyms, using Boolean operators (AND/OR) to ensure comprehensive retrieval of relevant literature.

The following steps are then performed:

- **Literature Search and Retrieval**: Searches are conducted across selected databases using carefully constructed queries.
- **Study Selection and Refinement**: Duplicate entries are removed, followed by title and abstract screening based on inclusion/exclusion criteria. Studies that pass the initial screening undergo full-text review and quality assessment.
- **Data Extraction**: Using the predefined form, data is systematically extracted from all qualifying studies.

3.2 Detailed Methodology Implementation

The methodology follows a structured SLR approach using a predefined review protocol. Research questions are framed using the PICOC model and searched across IEEE Xplore, ACM, and Semantic Scholar. Criteria for inclusion, exclusion, and quality assessment are applied to filter and evaluate studies. The following section presents a detailed implementation of the systematic review methodology.

3.2.1 Review Protocol

The review protocol is designed to ensure transparency, consistency, and reproducibility.

3.2.2 PICOC Framework

Table 3.1: PICOC Criteria

PICOC Element	Definition
Population	Cloud pricing models and cost control APIs across AWS, GCP, and Azure. hybrid cloud infrastructure, multi-cloud platforms, sky computing.
Intervention	Analysis of pricing strategies, cost optimization techniques, cost control APIs, Sky computing, platform evaluations.
Comparison	Comparative analysis of pricing models and cost control APIs between major cloud providers (AWS, Azure, GCP)
Outcome	Insight into Cost analysis, Cost control API, Platforms
Context	Cloud adoption in business and technical environments, especially in contexts involving hybrid infrastructure, multi-cloud strategy, and emerging technologies like Sky computing.

3.2.3 Research Questions (RQs)

- **RQ1:** What cloud pricing models are offered by AWS, Azure, and GCP, and how do they compare to on-premises infrastructure in terms of total and hidden costs?
- **RQ2:** What are the best practices for cost optimization in hybrid cloud infrastructures, and how can cloud provider cost control APIs (AWS, Azure, GCP) be leveraged to implement and automate?
- **RQ3:** How can Sky Computing influence cloud cost strategies?

3.2.4 Search Strategy

A structured search strategy is developed using relevant keywords and Boolean operators. The following digital libraries are used: IEEE Xplore, ACM Digital Library, and Semantic Scholar. For a more detailed search in electronic libraries, a search string is defined, according to the scope of this study, as follows

("pricing model" OR "cost model*" OR "billing model*" OR "cloud pricing model*") AND ("AWS" OR "Amazon Web Services" OR "Azure" OR "Microsoft Azure" OR "GCP" OR "Google Cloud Platform")* ***AND*** *(("cost optimization" OR "cost efficiency" OR "cost control") AND ("hybrid cloud" OR "hybrid infrastructure") AND ("best practices" OR "guidelines" OR "strategies")).*

3.2.5 Selection Criteria

To ensure the relevance and quality of the selected literature, a set of inclusion and exclusion criteria is defined in alignment with the research objectives and the PICOC framework. These criteria are used during the study selection phase to filter out irrelevant or low quality sources. Inclusion criteria(I) and exclusion criteria(E) stated below:

- **I1 Published Works:** Peer-reviewed journal articles and conference papers published between **2019 and 2025**, ensuring recent and relevant advancements in hybrid infrastructure optimization, sky computing, multi-cloud platforms

  Exception**:** For cloud pricing models, studies from **2010 to 2025** are considered, acknowledging that public cloud services and their associated pricing frameworks emerged during this period.

- **I2 Language:** Studies must be written in English to maintain consistency in analysis and interpretation.
- **I3 Cloud Providers and Platforms:** Research must specifically focus on AWS, Microsoft Azure, GCP, or explicitly address hybrid, multi-cloud, or sky computing platforms and technologies.
- **I4 Pricing and Cost Models:** Selected studies must address cloud pricing mechanisms, billing structures, cost estimation strategies, economic models, financial aspects of cloud services, or investigate cost optimization best practices in hybrid and multi-cloud infrastructures.
- **I5 Sky Computing:** Studies investigating the sky computing and cost control, as well as those analysing sky computing technologies.
- **E1 Scope Mismatch:** Studies not explicitly focused on AWS, Azure, GCP, hybrid infrastructure, multi-cloud technologies, sky computing technologies, or those broadly addressing cloud computing without specific reference to pricing, cost models, or infrastructure optimization.
- **E2 Content Type:** Non-peer-reviewed works, such as blog posts, editorials, informal communications, duplicate studies, magazines or studies with inaccessible full texts.
- **E3 Language:** Articles not written in English.

- **E4 Other Providers:** Papers exclusively discussing cloud providers or technologies other than AWS, Azure, GCP, hybrid/multi-cloud, sky computing.

Table 3.2 presents the results of applying the inclusion and exclusion criteria.

Table 3.2:Inclusion and Exclusion Criteria Results

Databases	Retrieved	Excluded	Screened	Selected
IEEE Xplore	918	883	35	10
Semantic Scholar	464	417	48	14
ACM	967	947	22	11

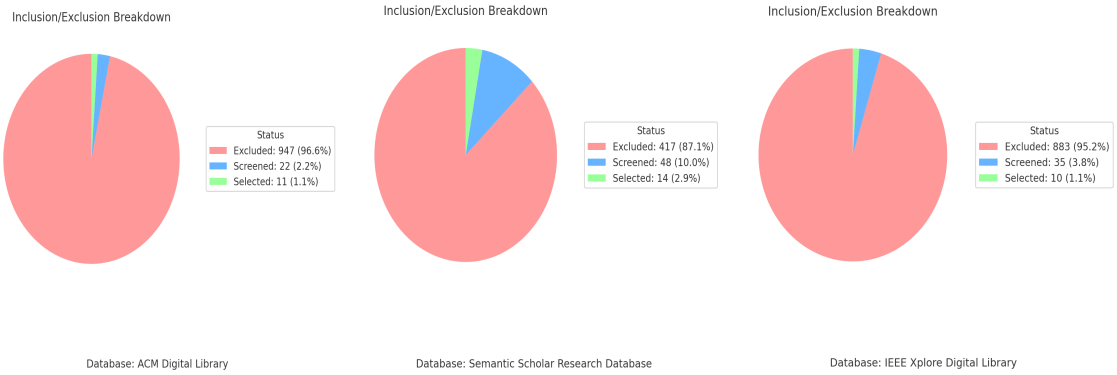

Figure 3.1: Inclusion/Exclusion Breakdown by Database

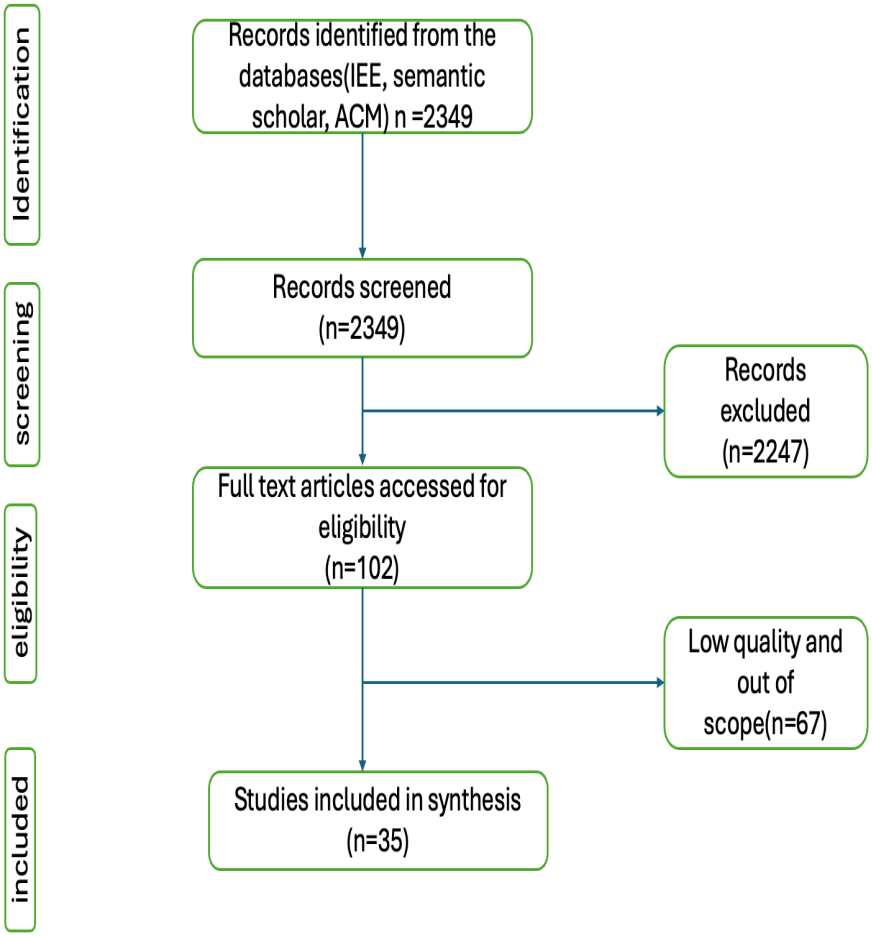

Figure 3.2: PRISMA Flow Diagram of Study Selection Process

3.2.6 Quality Assessment (QA)

To ensure the credibility, rigor, and relevance of the primary studies included in this Systematic Literature Review (SLR), a detailed quality assessment is conducted. This process is essential for enhancing the reliability of the synthesis. The quality assessment is based on four main criteria used to evaluate empirical studies:

1. **Reporting** – Is the study clearly communicated?
2. **Rigor** – Are the research design and methods appropriately executed?
3. **Credibility** – Are the findings reliable and supported by evidence?
4. **Relevance** – Is the study applicable to real-world hybrid cloud cost optimization scenarios?

These four aspects represent the best practices suggested in the review by Zhou et al. [22]. They noted that SLRs in software engineering focus on rigor and credibility but often overlook relevance. This study aims to avoid that gap.

Table 3.3: Quality Assessment Criteria for Selected Studies

Aspect	Quality Criterion (Assessment Question)	Question ID (Qn)
Reporting	Is there a clear statement of the research aims/goals/objectives?	Q1
	Is the context of the research adequately described?	Q2
	Is the reporting clear, coherent, and well-structured?	Q3
Rigor	Are the research methods clearly defined and appropriate to the research aims?	Q4
	Are the data collection and analysis methods adequately described and justified?	Q5
Credibility	Are the findings clearly stated and supported by evidence?	Q6
	Are limitations, threats to validity, or potential biases discussed?	Q7
Relevance	Is the study of value to research and/or practice (i.e., does it have practical or theoretical relevance)?	Q8

A scoring scheme is applied to each quality criterion:

- **1 point** → Fully covered
- **0.5 point** → Partially covered
- **0 points** → Not covered

Each selected study is evaluated using a set of quality assessment questions, and the individual scores are summed to calculate a total score for every paper. A threshold score of 7 serves to determine inclusion eligibility. Studies that meet or exceed this threshold are considered methodologically sound and advanced to the data extraction phase. For instance, the study titled *"A Survey of Cloud Computing Variable Pricing Models"* achieves a total score of 8 and meets the inclusion threshold for the final analysis.

Table 3.4: Quality Assessment Scorecard for Individual Study

Paper ID	Paper Title	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total Score	Verdict
P1	A Survey of Cloud Computing Variable Pricing Models	1	1	1	1	1	1	0.5	0.5	7	Included

The complete results of the quality assessment, including the individual scores and inclusion/exclusion decisions for all reviewed studies, are presented in Table 12.1 in the Appendix**,** titled *"Quality Assessment Scores and Inclusion Results"*.

3.2.7 Data Extraction

After selecting the studies through the quality assessment process, the next step involves systematically extracting key information from each paper to support the analysis phase. To ensure consistency and avoid missing any relevant details, a structured Data Extraction Form is developed and applied uniformly across all included studies.

Each article is carefully reviewed, and relevant data is recorded under pre-defined categories in the form. These categories are designed to capture essential aspects of each study in a clear and comparable manner. Specifically, the following fields are included in the form:

- **Paper ID**: Unique identifier assigned to each study.
- **Title**: Full title of the publication.
- **Authors and Year**: Authors of the study and the year it was published.
- **Key Findings**: Summary of the main results, insights, and contributions reported in the study related to the topics.
- **Gaps Found**: Any limitations, shortcomings, or research gaps identified by the authors or noted during the review.

Table 3.5 presents an example entry from the Data Extraction Form:

Table 3.5: Sample Data Extraction Entry for Selected Study

Paper ID	Title	Authors	Year	Key Findings	Gaps Found
P1	A Survey of Cloud Computing Variable Pricing Models	Sahar Arshad, Saeed Ullah, Shoab Ahmed Khan, M. Daud Awan, M. Sikandar Hayat Khayal	2014	1. Dynamic pricing models like Spot instances help reduce costs for low-priority jobs. 2. Mixed pricing strategies (spot + on-demand) are recommended for balancing cost and reliability.	1. Focus is more on providers profitability, less on user-side cost savings. 2. Real market-driven dynamic pricing strategies are not deeply explored.

The full data extraction results for all included studies are presented in Table 12.2 in the Appendix**,** titled *"Data Extraction Form"*. This table provides a comprehensive view of the data gathered from each study and serves as the foundation for synthesizing the results in later stages of the review.

3.2.8 Reporting the Review

As mentioned above, three databases containing cloud pricing model papers are examined using the defined search string described in Section 3.2.4. During the search, attention is given to the article title, abstract, and, in some cases, the conclusion. The decision to include or exclude a paper is made based on the defined inclusion and exclusion criteria as outlined in Section 3.2.5

3.2.9 Data Collection

A total of 2,349 records are initially retrieved from three major academic databases: IEEE Xplore, Semantic Scholar, and the ACM Digital Library. The PRISMA flow diagram (Figure: 3.2) illustrates that an initial screening based on titles and abstracts results in the exclusion of 2,247 records due to irrelevance or being out of scope. This leaves 102 articles for full-text eligibility assessment. Of these, 67 papers are excluded based on quality or relevance, resulting in 35 studies being included for synthesis.

The detailed contribution of each database is shown in Table 3.2. Specifically, 918 records are retrieved from IEEE Xplore, with 10 selected; Semantic Scholar provided 464 records, from which 14 are included and ACM contributed 967 records, resulting in 11 selected studies. Following this selection, a quality assessment is conducted as outlined in Section 3.2.6, reducing the pool to 19 high-quality papers. These studies subsequently undergo structured data extraction using the format described in Section 3.2.7, forming the foundation for the synthesis and analysis phase.

3.2.10 Data Synthesis

All papers are read and analysed. Data is extracted from each study to identify recurring patterns, compare findings, and derive meaningful insights aligned with the research question. The results are synthesized and discussed in detail in Chapter 4.

4 Analysis and Discussion of Results

4.1 Analysis of pricing models

This section presents the synthesized findings of the six selected studies on cloud pricing models. The review identifies major pricing strategies and research gaps. The analysis follows the SLR method using the PICOC framework.

4.1.1 Classification of Pricing Models

The studies reveal major categories of cloud pricing models. Kansal et al. [23] present a comprehensive overview of cloud service pricing models. He divides cloud pricing into three main models: pay-per-use, subscription-based, and hybrid models. Pay-per-use models charge users based on actual consumption, as seen in services like AWS EC2 Subscription-based models involve pre-booking resources for a fixed period, often at a lower cost, regardless of utilization. Hybrid models combine both approaches by allowing reserved resources with the flexibility to scale on-demand at additional per-use costs. The authors state that no single pricing model fits all scenarios. This demonstrates the need for flexible pricing strategies.

Murthy et al. [24] examine how pricing structures change with usage patterns. The authors identify and compare three principal pricing models: Linear**,** Variations of Linear**,** and Step models. In the Linear model**,** price scales directly with resource consumption. The Variations of the Linear model**,** used by Amazon EC2 introduce modifications such as upfront subscription fees (reducing hourly rates) or tiered discounts based on usage duration or thresholds. In the Step model, unit price decreases stepwise as usage crosses predefined thresholds. This highlights that Amazon's reserved instances are more cost-effective for long-term use**,** whereas on-demand instances are better for short-term needs**.**

Arshad et al. [25] categorize cloud pricing models into fixed (e.g., pay-as-you-go, subscription) and dynamic (e.g., spot pricing, auctions). They argue that fixed pricing leads to resource wastage and unfair costs, while dynamic models adjust prices based on market demand, improving efficiency.

Lee [26] views cloud pricing models as diverse and strategically significant. The paper identifies several prevalent models such as pay-as-you-go, subscription, tiered, spot, and hybrid pricing each with distinct advantages. Lee highlights that while dynamic pricing like spot instances offers flexibility and cost efficiency, it introduces complexity and risk of interruption. Conversely, subscription models provide predictable revenue for providers and cost savings for customers but require careful discounting to balance profitability and demand.

Choudhary et al. [27] analyse the pricing models of major cloud service providers AWS, Oracle Compute, Microsoft Azure, Google Cloud Platform, and IBM Cloud. The authors note that all providers follow pay-as-you-go principles but differ in billing granularity and discounting. AWS offers minute- or second-based pricing with options such as Dedicated**,** Spot**,** On-Demand**,** and Reserved Instances**.** Azure provides monthly and prepaid options including Reserved and Spot VMs**,** while Google Cloud combines hourly tiered pricing with subscription and hybrid models

Rohatgi at al. [28] analyses the pricing models of Microsoft Azure and Google Cloud Platform (GCP), emphasizing their flexibility and suitability for different workloads. Azure offers a Pay-as-you-go model for dynamic usage and Reserved Instances for one- or three-year commitments, providing significant cost savings for predictable workloads. GCP, on the other hand, utilizes an On-demand pricing model with no upfront commitment and offers Sustained Use Discounts**,** which automatically reduce costs for continuously running workloads. Figure 4.1 illustrates the frequency of pricing models found across the selected studies.

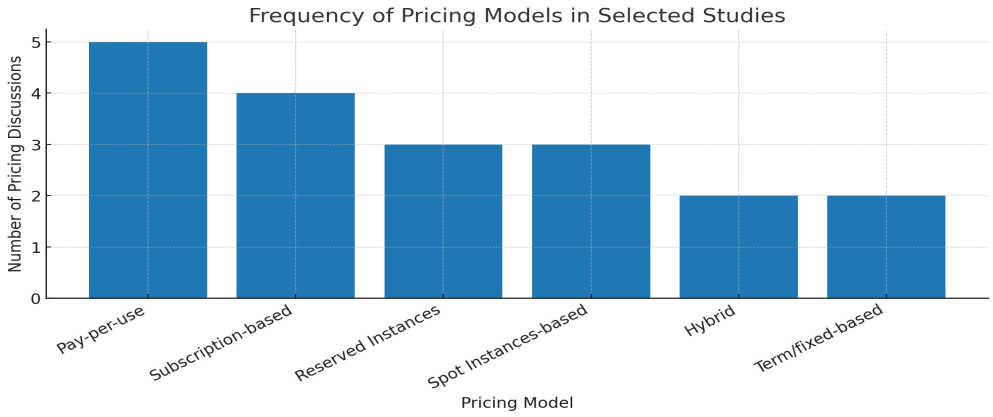

Figure 4.1: Number of Reviewed Papers Supporting Each Pricing models

4.1.2 Research Gaps Identified

1. **Lack of Regional Pricing Comparison**: None of the reviewed studies compare actual cloud pricing across different geographic regions, even though platforms like AWS, Azure, and GCP show significant cost variation by location.
2. **Real-World Cost Evaluation:** Most studies focus on theoretical pricing models without validating them against actual costs obtained from provider pricing calculators.

4.1.3 Summary of SLR Findings

The systematic review shows that cloud pricing models evolve from simple, fixed-rate models like pay-as-you-go and subscriptions to more dynamic setups that adjust to workloads and user behaviour. Static models are easy to understand but struggle to meet changing demands. Dynamic models, like spot pricing, are more efficient but can be risky due to price fluctuations or sudden drops in service. Some researchers recommend hybrid pricing, mixing financial models and algorithms to find a better balance. AWS offers a wide range of options but adds complexity. Azure provides more structured pricing suited for enterprises. GCP focuses on automation and transparency through sustained use discounts.

There is a significant lack of regional pricing comparisons. These insights lay the groundwork for implementation strategies to improve research gaps and pricing transparency through pricing calculators from each provider.

While the literature reveals valuable insights into pricing models, it overlooks two important aspects that are essential for real-world cloud decision-making. Data transfer costs (ingress and egress), and the total cost of ownership (TCO) comparison between cloud and on-premises infrastructure. These factors are very relevant in hybrid and enterprise contexts.

To complement the findings of the SLR, additional analyses are conducted:

- A comparative study of data ingress and egress pricing across AWS, Azure, and GCP.
- An empirical evaluation of the total cost of ownership (TCO) for on-premises versus cloud infrastructure using generalized compute instances.

These practical implementations enhance the relevance of the study by including operational cost considerations that are important to decision-makers. The details and analysis of these implementations are provided in Chapter 5.

4.2 Cost optimization in Hybrid Infrastructure

This section synthesizes the insights gathered from the selected studies on cost optimization in hybrid infrastructures**.**

4.2.1 Types of Cost Optimization Strategies in Hybrid Infrastructures

Across the studies reviewed, six main strategies emerged as the most discussed and widely used methods for managing and reducing costs in hybrid environments.

First, dynamic resource scheduling and provisioning is a common theme. Many papers note that hybrid environments involve fluctuating workloads. To handle this variability, solutions such as auto-scaling, and proactive-reactive scaling algorithms have been widely proposed. These methods ensure that resources are scaled up and down efficiently, reducing unnecessary spending during idle periods [29][30][31][32][33][34].

Another key focus is on optimizing storage costs. Several studies highlight the importance of smart data management, especially in scenarios where data needs to be frequently accessed or archived. Using tiered storage solutions, such as moving less important data to cold storage, can help balance cost and performance well [35][36][37][38].

Workload placement strategies are also frequently discussed. Researchers suggest that intelligently distributing workloads across public clouds, private clouds, and edge resources can help businesses find the right balance between cost, performance, and security [32][30][34][37].

Additionally, studies highlighted the need to carefully choose instance types and purchase options. Combining reserved instances, on-demand resources, and spot instances based on workload predictability is seen to offer significant savings potential [32][39].

Migration and refactoring strategies also play a key role. Some papers suggest that gradually migrating applications to hybrid environments, or refactoring them to better leverage cloud-native benefits, can help control costs over the long term [40][39].

Finally, the use of AI and automation emerges as a best practice. A few studies explore how machine learning can support smarter decision-making, helping organizations automatically choose the most cost-effective resource allocation strategies in real time [41][34][31][39][37].

Figure 4.2: Number of Reviewed Papers Supporting Each Cost Optimization Strategy

4.2.2 Summary of Findings

Overall, the findings from this review show a growing maturity in hybrid cloud cost optimization strategies. Organizations clearly move beyond basic cost-cutting measures and are now using more advanced approaches that consider workload variability, performance needs, and smart automation.

Dynamic provisioning and autoscaling techniques are essential for keeping resource usage aligned with demand. Effective storage management, through tiering and hybrid storage solutions, offers further opportunities to reduce expenses without compromising on access speed. Meanwhile, combining purchase options especially reserved and spot instances remains a tried and tested method to control cloud compute costs.

However, research does not fully address the challenges that come with hybrid setups. There is still a lack of cost models that look at the whole picture, including security, performance, and data movement between providers. Likewise, while edge computing and AI have potential to create new efficiencies, they remain in the early stages of being integrated into cost optimization plans.

In conclusion, although existing best practices establish a foundational framework, there remains a need for empirical validation. Chapter 5 addresses this by implementing a selected cost optimization strategy in a practical context to assess pricing differences and demonstrate its potential for reducing cloud costs.

5 Related Work

5.1 Analysis of pricing model across the public cloud providers

While cloud pricing models are studied in SLR process, a real-time cloud pricing is very dynamic and affected by several factors. These include geographical region, instance type, commitment level, and provider-specific strategies. There is a clear gap in the empirical comparison of pricing among cloud providers that use official pricing calculators.

This study addresses this gap by conducting a comparative analysis of real-time compute service pricing across the three major cloud providers AWS, Microsoft Azure, and GCP. The focus of this study is on general-purpose compute services, which are often used by small to medium-sized businesses because they provide a good balance of performance and cost efficiency. This study evaluates real-time compute pricing across different providers and regions using official pricing calculators.

In addition to compute pricing, it is also important to analyse data ingress and egress cost evaluations**,** which significantly influence the total cost of ownership but are often overlooked in pricing studies. Furthermore, comparing on-premises and public cloud cost models, especially for general-purpose compute scenarios, offers valuable insights for organizations considering cloud adoption or hybrid infrastructure strategies.

5.1.1 Real Time Pricing Model Comparison Across Cloud Providers

Each cloud provider uses a different naming convention for their compute services EC2 in AWS, Virtual Machines in Azure, and Compute Engine in GCP. To create a common term for this study, the term Virtual Compute Instance (VCI) is used to refer to these services. The focus is on the general-purpose versions of VCIs.

Before comparing prices, it is important to understand how each cloud provider calculates usage time and charges for it:

- **Amazon Web Services (AWS)** also charges for at least 1 minute, and then switches to per-second billing [42].
- **Microsoft Azure**, on the other hand, charges by the second right from the start, even though prices are usually shown per hour or per month. This means charges apply only for the exact duration the virtual machine remains active [43].
- **Google Cloud Platform (GCP)** charges for a minimum of 1 minute. After that, it bills by the second [44].

For this analysis, the general-purpose variants of VCIs are selected:

- **AWS: t3a.2xlarge**
- **Azure: Standard D8as v5**
- **GCP: n2-standard-8**

To examine pricing variability across geographies, two representative regions are selected:

- **Frankfurt, Germany**
- **Virginia, USA**

These regions are supported by all three cloud providers .These instance types and regions form the basis for evaluating real-time pricing strategies. Across all three providers, compute services are priced under comparable models. This study focuses on the following categories:

- **On-Demand/Pay as you go:** Flexible pay-as-you-go pricing without long-term commitment.
- **Reserved / Savings Plans/ Committed use discount:** Discounts provided in exchange for 1-year or 3-year usage commitments.
- **Spot Instances:** Access to unused capacity at lower costs, with potential interruptions.

This comparison explores real-time pricing variations for equivalent VCI configurations across providers and regions. Specifically, it reveals how regional pricing strategies affect cost optimization decisions.

This study presents a detailed comparison of the real-time pricing models offered by AWS, Azure, and GCP for general-purpose virtual compute instances (VCIs). The analysis is based on the standardized instance types and selected regions as described above, which ensure consistency in evaluating pricing differences across providers and geographic markets.

All pricing data is sourced from official pricing calculators for each provider to reflect the most up-to-date and accurate costs:

- AWS Pricing Calculator [45].
- Azure Pricing Calculator [46].
- GCP Pricing Calculator [47].

Azure's monthly prices are normalized to hourly rates by dividing by 730 (the average number of hours in a month) to enable fair comparisons across all providers.

All pricing values presented in this analysis are standardized in US Dollars (USD) to ensure consistency across all cloud providers and regions. This uniform currency treatment allows for a direct and fair comparison of costs. The prices are based on data collected during March–April 2025 using the official pricing calculators and public documentation from AWS, Azure, and GCP.

A. AWS Pricing Analysis

AWS offers a comprehensive set of pricing models structured for various usage needs and commitment levels. These include Savings Plans**,** Reserved Instances (Standard)**,** Reserved Instances (Convertible)**,** On-Demand**,** and Spot Instances. For both Savings Plans and Reserved Instances (Standard and Convertible), users can choose among three payment options**:** No Upfront, Partial Upfront, and All Upfront. Each of these is available in 1-year and 3-year terms. This models

allows users to balance cost and flexibility based on workload predictability and commitment duration.

These pricing models differ significantly in terms of cost structure, flexibility, and ideal use cases:

- **Savings Plans** allow users to commit to a consistent usage level (measured in $/hour) over a 1- or 3-year term, offering up to 72% savings compared to On-Demand pricing. It provide flexible coverage across instance types and regions.
- **Standard Reserved Instances** offer the highest discounts (up to 72%) but require a fixed commitment to a specific instance type and region. They are best suited for predictable, steady-state workloads.
- **Convertible Reserved Instances** provide slightly lower discounts (up to 54%) but allow changes in instance family, operating system, and tenancy over time, offering a balance between savings and flexibility.
- **On-Demand Instances** provide maximum flexibility with no long-term commitment, charging by the second. This model is ideal for unpredictable or short-lived workloads but comes at a higher cost.
- **Spot Instances** offer the steepest discounts (up to 90%) by utilizing unused AWS capacity, but can be interrupted with short notice within 2 minutes, making them suitable only for fault-tolerant applications like batch processing or data analytics.

Prices also change depending on the AWS region. Some regions are more expensive than others because of different costs and demand. The AWS pricing table (Table: 5.1) highlights noticeable differences not only between pricing models but also across regions. Among all options, Spot Instances are the most affordable, priced at $0.1143 per hour in Virginia and $0.1640 in Frankfurt a steep 43.48% increase in Frankfurt. On the other end, On-Demand instances are the most expensive, with rates of $0.3008 in Virginia and $0.3645 in Frankfurt, reflecting a 21.18% regional difference.

For users seeking long-term savings, Savings Plans and Standard Reserved Instances (SRI) offer considerable discounts. Regardless of whether it is a 1-year or 3-year term, or which payment option is chosen, Frankfurt consistently remains about 15% more expensive than Virginia.

Convertible Reserved Instances (CRI), while more flexible, show the biggest pricing gap between the two regions. A 3-year Partial Upfront CRI comes to $0.1386/hr in Virginia, compared to $0.1882/hr in Frankfurt, marking a 35.78% difference. In general, CRIs are 32–36% more expensive in Frankfurt, suggesting that the added flexibility comes with a higher price tag in certain regions.

In conclusion, Frankfurt is consistently more expensive than Virginia across all AWS pricing models, with cost differences ranging from 14% to over 40%, making Virginia the more cost-effective region for AWS deployments.

Table 5.1: AWS Pricing table between two regions

Plan Type	Term	Payment Option	Rate (USD/hr)_Virginia	Rate (USD/hr)_Frankfurt	Price Difference (USD/hr)	Price Difference (%)
Savings Plan	1 Year	No Upfront	0.1886	0.2177	0.0291	15.43
Savings Plan		Partial Upfront	0.1796	0.2074	0.0278	15.48
Savings Plan		All Upfront	0.176	0.2032	0.0272	15.45
Savings Plan	3 Year	No Upfront	0.1299	0.1493	0.0194	14.93
Savings Plan		Partial Upfront	0.1208	0.1382	0.0174	14.40
Savings Plan		All Upfront	0.1131	0.1299	0.0168	14.85
Reserved (SRI)	1 Year	No Upfront	0.1886	0.2177	0.0291	15.42
Reserved (SRI)		Partial Upfront	0.1796	0.2073	0.0277	15.42
Reserved (SRI)		All Upfront	0.1760	0.2032	0.0272	15.45
Reserved (SRI)	3 Year	No Upfront	0.1299	0.1493	0.0194	14.93
Reserved (SRI)		Partial Upfront	0.1204	0.1382	0.0177	14.78
Reserved (SRI)		All Upfront	0.1131	0.1299	0.0167	14.85
Convertible (CRI)	1 Year	No Upfront	0.2170	0.2886	0.0716	32.99
Convertible (CRI)		Partial Upfront	0.2066	0.2730	0.0664	32.13
Convertible (CRI)		All Upfront	0.2025	0.2675	0.0650	32.09
Convertible (CRI)	3 Year	No Upfront	0.1498	0.2033	0.0535	35.71
Convertible (CRI)		Partial Upfront	0.1386	0.1882	0.0496	35.78
Convertible (CRI)		All Upfront	0.1359	0.1845	0.0486	35.76
On-Demand	–	–	0.3008	0.3645	0.0637	21.18
Spot instances		spot	0.1143	0.1640	0.0497	43.48

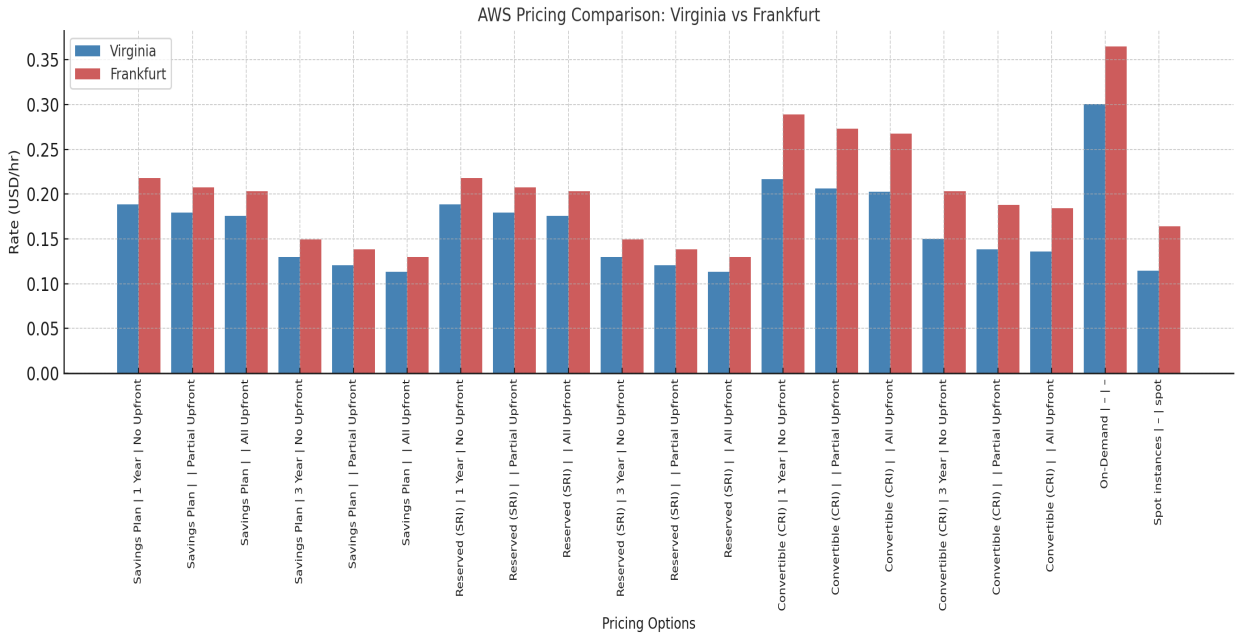

Figure 5.1: AWS pricing comparison between two regions

B. Azure Pricing Analysis

Azure provides a range of pricing models to accommodate different user needs, including Pay-as-you-go**,** Reservations**,** Savings Plans**,** and Spot pricing. The Pay-as-you-go model offers flexibility without long-term commitment, while Reservation options are available for both 1-year and 3-year terms**,** offering substantial cost reductions for predictable workloads. Azure also features Savings Plans for 1-year and 3-year terms, which further optimize costs based on consistent usage. Additionally, Spot plans allow users to take advantage of unused capacity at significantly reduced rates.

These pricing models varies significantly in terms of cost structure, flexibility, and ideal use cases:

- **Azure Savings Plan for Compute** allows users to commit to a fixed hourly spending rate over a one- or three-year period, offering significant cost savings often up to 65% compared to standard Pay-As-You-Go pricing.
- **Azure Reserved Instances (Standard)** provide the deepest discounts up to 72% in exchange for a fixed commitment to a specific virtual machine type and region over one or three years. This model is ideal for stable, predictable workloads with consistent usage patterns, such as production environments or large-scale databases. While Azure permits limited exchanges and cancellations any unused reserved capacity up to the yearly limit. Unlike AWS, Azure does not offer a fully convertible Reserved Instance option, but some flexibility can be achieved through instance exchange mechanisms.

- **Pay-As-You-Go** model offers the highest level of flexibility by charging users per second based on actual consumption, with no upfront payments or long-term contracts. This model is well-suited for unpredictable or short-lived workloads, development and testing environments, and organizations that prefer to treat compute expenses as operational rather than capital expenditures
- **Azure Spot Virtual Machines** provide the most substantial discounts up to 90% by utilizing Azure's unused capacity. These instances are highly cost-effective for fault-tolerant and interruptible workloads. Spot VMs are interruptible with little notice if Azure needs to reallocate the capacity elsewhere. Pricing operates on a bidding model where users specify the maximum price they are willing to pay.

Similar to AWS, Azure pricing also varies significantly by region due to differences in operational costs and local demand. The pricing comparison between the East US (Virginia) and West Europe (Frankfurt) regions reveals Frankfurt is notably more expensive across all pricing models. For example, Pay-as-you-go plans cost $0.416/hr in Frankfurt versus $0.344/hr in Virginia, a 20.93% increase. Reserved Instances for 1-year and 3-year terms follow a similar pattern, with Frankfurt rates being approximately 20.87% and 20.89% higher respectively.

Azure's Savings Plans show an even more regional cost gap. A 1-year Savings Plan costs $0.3162/hr in Frankfurt compared to $0.2021/hr in Virginia a 56.46% increase. The 3-year plan reflects a 44.99% hike. Interestingly, the Spot pricing model, while being the most cost-effective in both regions, still exhibits a 40% price difference $0.07/hr in Frankfurt versus $0.05/hr in Virginia.

These findings indicate that while Azure offers flexible pricing models to optimize cloud spend, regional pricing differences remain substantial. For cost-effective users, deploying workloads in Virginia proves significantly cheaper, especially under long-term commitments or when leveraging discounted pricing models like Reserved Instances or Savings Plans.

Table 5.2: Azure Pricing table between two regions

Plan Type	Term	Payment Option	Monthly Rate (USD)_Virginia	Monthly Rate (USD)_Frankfurt	Rate (USD/hr)_Virginia	Rate (USD/hr)_Frankfurt	Price Difference (USD/hr)	Price Difference (%)
Pay-as-you-go	-	Pay-as-you-go	251.12	303.68	0.344	0.416	0.072	20.93
Reservations	1 Year	Reservations	110.5	133.58	0.1514	0.183	0.0316	20.87
Reservations	3 Years	Reservations	95.42	115.32	0.1307	0.158	0.0273	20.89
Savings Plan	1 Year	Savings Plan	147.54	230.8	0.2021	0.3162	0.1141	56.46
Savings Plan	3 Years	Savings Plan	115.19	167.02	0.1578	0.2288	0.071	44.99
Spot		Spot plan	37.59	51.02	0.05	0.07	0.071	40

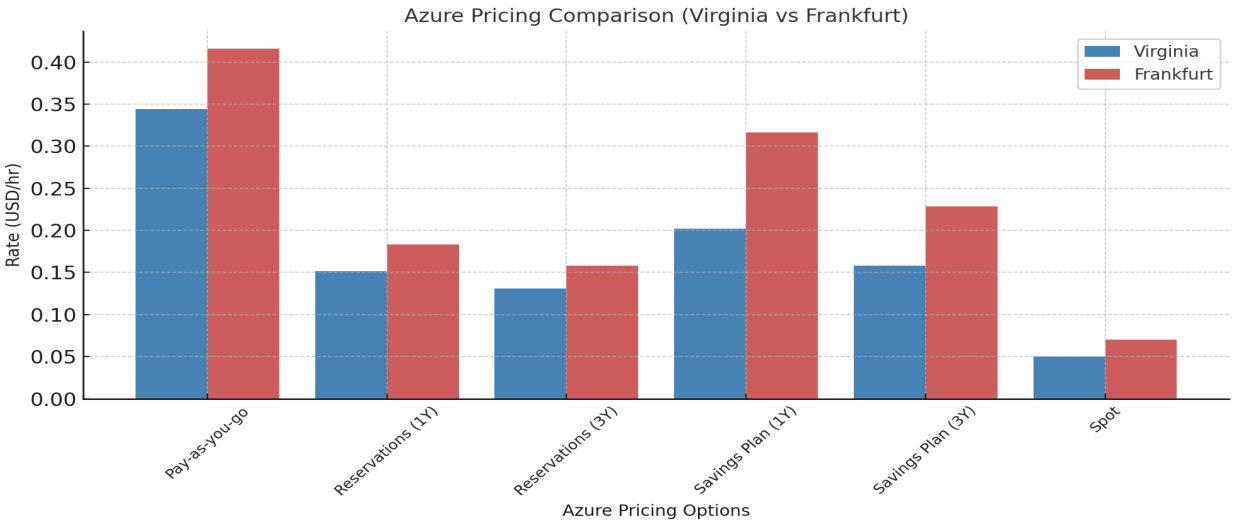

Figure 5.2: Azure pricing comparison between two regions

C. GCP Pricing Analysis

Google Cloud Platform (GCP) supports a set of pricing models focused on simplicity and flexibility. These include Pay-as-you-go**,** Committed Use Discounts (CUDs) for 1-year and 3-year terms**,** and Spot instances (preemptible VMs). The Pay-as-you-go model offers maximum flexibility with no long-term commitment, while Committed Use Discounts provide reduced pricing in exchange for usage commitment over a fixed term. Spot instances are the most cost-effective option, ideal for fault-tolerant or batch workloads.

These pricing models varies significantly in terms of cost structure, flexibility, and ideal use cases:

- **Pay-as-you-go** : Google Cloud's pay-as-you-go pricing model provides full flexibility, with charges based only on actual usage no upfront commitments or long-term contracts required. This makes it a great option for businesses with unpredictable or variable workloads, as it allows them to scale up or down at any time based on demand. That said, this flexibility comes at a cost it's the most expensive option overall compared to other long-term plans.
- **Committed Use Discounts**: reward customers who agree to use specific resources for a period of one or three years. In return, they receive significant price reductions sometimes as high as 57% compared to standard rates. This model is ideal for companies with stable, long-term infrastructure needs.
- **Sustained Use Discounts**: This on the other hand, comes in automatically. If a service is used consistently for a large portion of the month, Google applies discounts of up to 30% without any manual setup. It's a smart way to save money on workloads that run continuously. To get the most value out of these discounts, it's important to monitor usage closely and ensure the commitment is being utilized efficiently.
- **Spot VM**: Spot VMs offer an attractive option. These are heavily discounted virtual machines up to 91% cheaper than regular pricing. Spot VM can be terminated at any time if Google needs the capacity for other tasks.

A closer examination of Google Cloud's pricing across different regions and plan types reveals significant cost variations between Frankfurt (Germany) and Virginia (USA). This variance is especially relevant for organizations aiming to optimize cloud expenditure based on geographical deployment.

Among the available pricing models, spot instances which represent the most cost-effective, preemptible virtual machine option show the most regional difference. In Virginia, the cost is approximately $0.1105 per hour, whereas in Frankfurt, it rises to $0.1447 per hour, marking a 30.95% increase. The pay-as-you-go model, known for its flexibility and lack of commitment, also demonstrates a notable price gap. In Virginia, the pay-as-you-go rate is $0.4374 per hour, while in Frankfurt it is $0.5005 per hour about 14% higher.

For enterprises with predictable long-term workloads, Google Cloud offers committed use discounts. A one-year commitment results in a rate of $0.2619 per hour in Virginia compared to $0.315 in Frankfurt, reflecting a 20.27% difference. With a three-year commitment, the rate is $0.1871 per hour in Virginia and $0.225 in Frankfurt—about 20% higher. These findings suggest that while long-term commitments do offer significant savings over on-demand pricing, the choice of region continues to have a substantial impact on cost.

In summary, this analysis results Frankfurt remains more expensive than Virginia across all Google Cloud pricing models, with regional cost differences ranging from 14% to over 30%. These findings are critical for cloud strategy planning, especially for organizations that have the flexibility to choose between deployment regions.

Table 5.3: GCP Pricing table between two regions

Plan Type(model)	Term	Payment Option	Rate (USD/hr)_Frankfurt	Rate (USD/hr)_Virginia	Price Difference (USD/hr)	Price Difference (%)
Regular(commited use discount)	1 Year	1 Year	0.315	0.2619	0.0531	20.27
Regular(commited use discount)	3 Years	3 Years	0.225	0.1871	0.0379	20.26
Regular		Pay as you go	0.5005	0.4374	0.0631	14.43
Spot(preemptible vm)	-	Spot	0.1447	0.1105	0.0342	30.95

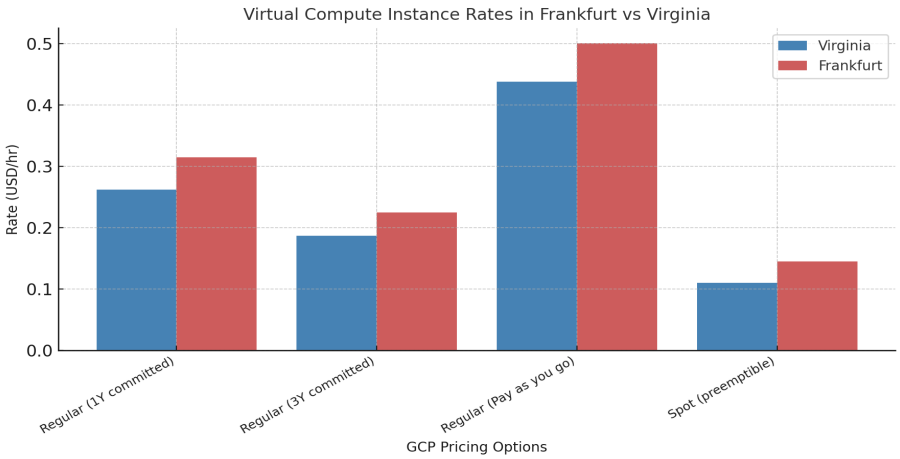

Figure 5.3: GCP pricing comparison between two regions

The pricing models are analysed and their regional differences across AWS, Azure, and GCP, it becomes evident that both the pricing structure and the selected deployment region significantly influence overall cloud costs. Each provider offers a variety of pricing models such as Savings Plans, Reserved Instances, Committed Use Discounts, On demand and Spot pricing tailored for different workload characteristics and budget strategies. However, even with the same pricing model, costs can vary a lot depending on the region.

For example, Frankfurt was consistently more expensive than Virginia across all three cloud providers . Spot instances, which are typically the lowest-cost option, show the greatest regional pricing gaps up to 43% in AWS, 40% in Azure, and nearly 31% in GCP making them highly sensitive to regional selection. Similarly, Pay-as-you-go pricing, while offering flexibility, proved more expensive across all providers in Frankfurt by 14–21%, depending on the provider.

Long-term commitment models, such as AWS Reserved Instances, Azure Reservations, and GCP Committed Use Discounts, did offer significant savings over on-demand rates. However, they too showed consistent regional markups in Frankfurt, typically around 15–20%. Azure's Savings Plans stand out for their steep regional difference up to 56% higher in Frankfurt making region selection even more critical when committing to longer terms.

In terms of pricing model diversity, AWS offered the most extensive range with customizable payment options (No Upfront, Partial Upfront, and All Upfront) across both Standard and Convertible Reserved Instances, allowing greater flexibility. Azure provided structured savings through Reservations and Savings Plans but lacked full convertibility. GCP, while offering fewer pricing tiers, maintained simplicity with strong savings under its Committed Use Discounts and automatic Sustained Use Discounts.

In conclusion, this comparative analysis highlights that optimizing cloud costs requires more than just selecting the cheapest pricing model it demands a strategic evaluation of both the pricing structure and deployment region. While all providers offer substantial savings opportunities, the region-specific pricing variation especially between Frankfurt and Virginia can outweigh the benefits of even the most aggressive discount plans.

5.1.2 Data Ingress and Egress Cost Analysis

In cloud computing, the movement of data both into (ingress) and out of (egress) virtual infrastructure plays a crucial role in determining total operational expenditure. While compute and storage costs often get most of the focus, data transfer pricing can become a hidden but significant expense, particularly in hybrid or multi-cloud deployments where data is frequently exchanged across regions, services, and platforms.

This section presents a comparative analysis of data ingress and egress pricing models across three major cloud providers AWS, Azure and GCP. Two key regions are selected for each provider to ensure geographical diversity: Frankfurt (Europe) and Northern Virginia (USA). This comparison highlights the cost implications of regional cloud architecture choices and helps in making informed decisions for global deployments.

A. **Amazon Web Services (AWS)**

Frankfurt Region (EU-Central-1) : Data from pricing calculator

AWS offers free ingress across all regions, including Frankfurt. However, outbound transfers (egress) vary based on destination [45].

Table 5.4: AWS Ingress and egress costs in Frankfurt Region

Transfer Type	Direction	Destination / Source	Rate	Example Volume	Monthly Cost (1 TB)
Inbound	Into Frankfurt	Internet / Other AWS Regions	Free	1 TB	$0.00
Intra-Region	Within Frankfurt	EC2 ⇄ S3 / Other Services	$0.01 per GB (each way “in +out”)	1 TB	$20.48
Outbound	From Frankfurt	All Other Regions	$0.02 per GB	1 TB	$20.48
Outbound (Asia Pacific)	From Frankfurt	Thailand	$0.08 per GB	1 TB	$81.92
Outbound (Canada/Mexico)	From Frankfurt	Calgary / Mexico	$0.05 per GB	1 TB	$51.20
Outbound (Internet)	From Frankfurt	Internet	$0.05–$0.09 per GB	1 TB	$51.20–$92.16
Outbound (CloudFront)	From Frankfurt	Amazon CloudFront	Free	1 TB	$0.00

Inbound (incoming) data transfers to the AWS Frankfurt region are completely free, no matter if the data comes from the Internet or another AWS region. This makes a good choice for receiving data in global setups. However, if data moves within the region like between EC2 and S3 it costs $0.01 per GB each way, which means $0.02 per GB in total for a round trip. Outbound (outgoing) data transfers have different prices depending on where the data is going. The cheapest option is sending data to other AWS regions, which costs $0.02 per GB. Sending data to the Internet is more expensive, between $0.05 and $0.09 per GB. A great way to save is using Amazon CloudFront

sending data to it is completely free, making it a smart option for delivering content around the world at lower costs.

US East – Northern Virginia Region

Table 5.5: AWS Ingress and egress costs in Virginia Region

Transfer Type	Direction	Source / Destination	Rate	Example Volume	Monthly Cost (1 TB)
Inbound	Into Virginia	Internet / Other AWS Regions	Free	1 TB	$0.00
Intra-Region	Within Virginia	EC2 ⇄ S3 / AZ ⇄ AZ / VPC Peering	$0.01 per GB (each way)	1 TB	$20.48
Outbound	From Virginia	All Other Regions	$0.02 per GB	1 TB	$20.48
Outbound (Asia Pacific)	From Virginia	Thailand	$0.08 per GB	1 TB	$81.92
Outbound (Senegal)	From Virginia	Dakar	$0.05 per GB	1 TB	$51.20
Outbound (US East - Ohio)	From Virginia	US East (Ohio)	$0.01 per GB	1 TB	$10.24
Outbound (US Verizon)	From Virginia	Nashville / Tampa (Verizon edge)	Free	1 TB	$0.00
Outbound (Internet)	From Virginia	Internet	$0.05–$0.09 per GB	1 TB	$51.20–$92.16
Outbound (CloudFront)	From Virginia	Amazon CloudFront	Free	1 TB	$0.00

Inbound data transfers to AWS's North Virginia (US East) region are completely free, whether the data comes from the Internet or other AWS regions. This is part of AWS's standard policy worldwide. Within the same region, moving data for example, between Availability Zones or VPCs is not free. The cost is $0.01 per GB in each direction, totalling approximately $20.48 per terabyte for both sending and receiving data..

When sending data out of the region (egress), the cost depends on where it's going. Transferring data to other AWS regions costs $0.02 per GB, but sending data to the Internet is more expensive between $0.05 and $0.09 per GB. Sending data to services like Amazon CloudFront or certain nearby AWS edge locations is free. Transferring data to a nearby region such as US East (Ohio) is more affordable, costing just $0.01 per GB. This makes it a cost-effective choice for backups or disaster recovery setups.

B. Microsoft Azure

Azure offers free ingress across all regions. However, outbound transfers (egress) vary based on destination [46].

Germany West Central (Frankfurt) : Data from pricing calculator

Table 5.6: Azure Ingress and egress costs in Frankfurt Region

Transfer Type	Direction	From → To Region	Rate per GB	Example Volume	Monthly Cost	Volume (1 TB)	Monthly Cost (1 TB)
Inter-Region (first 5 GB)	Outbound	Any Region	Free	5 GB	$0.00	—	$0.00
Inter-Region	Outbound	Germany → East US	$0.00833	6 GB	$0.05	1000 GB	$49.75
Inter-Region	Outbound	Germany → Switzerland North	$0.0199	6 GB	$0.02	1000 GB	$19.90
Inter-Region	Outbound	Germany → New Zealand North	$0.00833	6 GB	$0.05	1000 GB	$49.75
Internet Egress (first 100 GB)	Outbound	Germany → Internet	Free	100 GB	$0.00	—	$78.30 **(beyond free tier)**

Azure provides free data transfer allowances that enhance affordability for small-scale workloads. Each month, the first 5 GB of data transferred to other Azure regions is free. Similarly, the first 100 GB of outbound internet data per zone is included at no charge, supporting lightweight external workloads. Beyond these thresholds, pricing varies by destination. For example, transfers to geographically closer regions such as Switzerland North are priced lower—approximately $0.0199 per GB—likely due to reduced transit costs. In contrast, data transfers to more distant locations, such as New Zealand or the East US, incur higher rates. Once the 100 GB free internet egress limit is exceeded, the cost rises to $0.0783 per GB, equivalent to $78.30 per terabyte.

East US (Northern Virginia)

Table 5.7: Azure Ingress and egress costs in Virginia region

Transfer Type	Direction	From → To Region	Rate per GB	Example Volume	Monthly Cost	Volume (1 TB)	Monthly Cost (1 TB)
Inter-Region (first 5 GB)	Outbound	Any Region	Free	5 GB	$0.00	—	$0.00
Inter-Region	Outbound	East US → Germany West Central	$0.04975	6 GB	$0.05	1000 GB	$49.75
Inter-Region	Outbound	East US → Switzerland North	$0.04975	6 GB	$0.05	1000 GB	$49.75
Inter-Region	Outbound	East US → New Zealand North	$0.04975	6 GB	$0.05	1000 GB	$49.75

Inter-Region	Outbound	East US → West Central US	$0.01990	6 GB	$0.02	1000 GB	$19.90
Internet Egress (first 100 GB)	Outbound	East US → Internet	Free	100 GB	$0.00	—	$78.30

Each month, the first 5 GB of data transferred to other Azure regions is free, along with 100 GB of complimentary internet egress per region. After that, the costs depend on where the data is going. For example, sending data from East US to regions like Germany, Switzerland, or New Zealand costs around $49.75 per terabyte, while transfers to other US regions like from East US to West Central US-based rates are lower, at approximately $19.90 per terabyte. When the 100 GB free internet limit is exceeded, the cost increases to $78.30 per terabyte for additional outbound traffic. Compared to AWS and GCP, Azure's free data allowances enhance cost-effectiveness for lightweight distributed systems and startups managing basic connectivity requirements.

C. Google Cloud Platform (GCP)

Frankfurt (europe-west3) - costs taken from the google documentation.

GCP offers free ingress across all regions. However, outbound transfers (egress) vary based on destination. GCP pricing varies by network tier, with Premium Tier enabled by default [48].

Table 5.8: GCP Ingress and egress costs in Frankfurt region

Transfer Type	Direction	Source / Destination	Rate	Example Volume	Monthly Cost (1 TB)
Inbound	Into Frankfurt	Internet or any region	Free	1 TB	$0.00
Intra-Zone	Within Zone	Same zone (internal IP)	Free	1 TB	$0.00
Inter-Zone (same region)	Internal	Same region (e.g., zone-a → zone-b)	$0.01 per GiB	1 TB	$10.24
Inter-Region (Premium Tier)	Outbound	Frankfurt → North America	$0.11 per GiB	1 TB	$112.64
Inter-Region (Premium Tier)	Outbound	Frankfurt → Europe	$0.11 per GiB	1 TB	$112.64
Inter-Region (Premium Tier)	Outbound	Frankfurt → Asia	$0.11 per GiB	1 TB	$112.64
Internet Egress (Premium)	Outbound	Frankfurt → North America (1–10 TB)	$0.11 per GiB	1 TB	$112.64
Internet Egress (Standard)	Outbound	Frankfurt → North America (1–10 TB)	$0.065 per GiB	1 TB	$66.56

Google Cloud Platform (GCP) offers flexible data transfer pricing based on the selected network tier, with the Premium Tier set as the default. In the Frankfurt region, all inbound data transfers whether from the Internet or another GCP region are completely free. Similarly, data transfers within the same zone (intra-zone) using internal IPs are also free, which helps reduce costs for tightly coupled services.

However, once data moves between zones within the same region (inter-zone), there's a cost of $0.01 per GiB, which totals about $10.24 per terabyte. For outbound transfers to other regions like North America, Europe, or Asia using the Premium Tier, the price is relatively high at $0.11 per GiB, or $112.64 per terabyte.

When it comes to sending data to the Internet, the cost depends on the network tier. Using the Premium Tier, internet egress from Frankfurt to North America costs the same as inter-region transfer—$0.11 per GiB. However, selecting the Standard Tier reduces the cost significantly to $0.065 per GiB, or $66.56 per terabyte. This makes the Standard Tier a more economical choice for large-scale internet transfers, though it may come with slightly lower performance compared to Premium

For data transfers beyond 10 TB, prices decrease, offering economy of scale for high-volume users. The first 200 GiB per month is free across all regions

Northern Virginia (us-east4)

Table 5.9: GCP Ingress and egress costs in Virginia region

Transfer Type	Direction	Source / Destination	Rate	Example Volume	Monthly Cost (1 TB)
Inbound	Into Virginia	Internet or any region	Free	1 TB	$0.00
Intra-Zone	Within Zone	Same zone (internal IP)	Free	1 TB	$0.00
Inter-Zone (same region)	Internal	Same region (e.g., zone-a → zone-b)	$0.01 per GiB	1 TB	$10.24
Inter-Region (Premium Tier)	Outbound	Virginia → Europe	$0.05 per GiB	1 TB	$51.20
Inter-Region (Premium Tier)	Outbound	Virginia → Asia	$0.08 per GiB	1 TB	$81.92
Internet Egress (Premium)	Outbound	Virginia → Europe (1–10 TB)	$0.11 per GiB	1 TB	$112.64
Internet Egress (Standard)	Outbound	Virginia → Europe (1–10 TB)	$0.065 per GiB	1 TB	$66.56

In the US East (Northern Virginia) region, Google Cloud Platform (GCP) maintains its consistent data transfer pricing model, offering several cost-saving opportunities depending on usage and configuration. As with other GCP regions, inbound data transfers whether from the Internet or other GCP regions are completely free. Data moved within the same zone (intra-zone) also remains free when using internal IP addresses.

For inter-zone transfers within the same region, such as between different zones in Virginia, GCP charges $0.01 per GiB, resulting in a monthly cost of approximately $10.24 per terabyte. When sending data to other regions using the Premium Tier, the rates vary by destination. Transfers from Virginia to Europe cost $0.05 per GiB ($51.20 per TB), while transfers to Asia are priced higher at $0.08 per GiB ($81.92 per TB).

Internet egress charges also differ by tier. With the Premium Tier, sending data from Virginia to Europe costs $0.11 per GiB, or $112.64 per TB. Choosing the Standard Tier significantly reduces that cost to $0.065 per GiB, or $66.56 per TB. This makes the Standard Tier a more budget-friendly option for non-critical or bulk data transfers, while the Premium Tier may be preferred for latency-sensitive or performance-focused workloads.

For data transfers beyond 10 TB, prices decrease, offering economy of scale for high-volume users. The first 200 GiB per month is free across all regions

5.1.3 On prem Vs Cloud providers

The server configuration used in this on-premises cost analysis is selected to represent the general-purpose VCIs that major public cloud providers offer, such as AWS, Azure, and GCP. This choice aims to ensure a consistent comparison and allow for a fair evaluation between cloud-hosted virtual machines and physical on-premises compute infrastructure.

The selected on-prem server setup includes a single-node enterprise-grade system built around an Intel® Xeon® E-2468 processor, which features 8 cores and operates at a thermal design power (TDP) of 65W. This is paired with 32 GB of ECC DDR5 RAM, a 2TB SATA hard disk drive, and a 700W Titanium-rated non-redundant power supply unit. The system is a 9U rackmount chassis and includes essential peripherals such as a power distribution unit (PDU), KVM switch, and cabling for deployment in a standard data center environment.

This configuration is chosen to reflect the performance characteristics of the cloud VCIs analysed earlier: the t3a.2xlarge instance in AWS, D8as v5 in Azure, and n2-standard-8 in GCP. All three cloud instances are part of the general-purpose compute category and are selected as reference points to guide the physical infrastructure choice.

It highlights the importance of examining operational metrics that often lack transparency in cloud models. These metrics include power consumption, cooling needs, licensing, and real estate costs all significant expenses for businesses managing their own data centers.

This cost modelling process applies methodologies from two main sources: the first offers a structured way to estimate rated and actual power usage, cooling loads, and related electricity costs [49].The second provides insights on cost modelling, capital versus operational expenditure (CapEx vs OpEx), and total cost of ownership (TCO) estimation over time [50].

A. Scope Limitation

This part of the analysis is intentionally limited to the evaluation of server-level infrastructure that is, the fundamental computing resources and essential physical components required to operate a general-purpose virtual machine within an on-premises environment. Specifically, it includes the physical servers themselves, directly attached storage drives, power supply and cooling systems, as well as rack-mounted facilities and basic environmental control mechanisms such as fans and sensors.

However, this analysis does not extend to the broader IT ecosystem typically found in a production environment. It excludes components such as database systems (e.g., Amazon RDS, Google Cloud SQL), managed storage solutions (like Amazon EBS or Azure Managed Disks), network services including firewalls and load balancers, and software licensing or staffing costs associated with system administration. Furthermore, it does not account for application-level services or platform-specific tools that might be integrated into full-scale cloud or enterprise IT deployments. By narrowing the focus in this way, the study ensures a clear and manageable comparison between the cost of owning and operating physical compute infrastructure and the equivalent cost of VCIs in the public cloud.

B. Capital Expenditures of the IT Infrastructure

The capital expenditures include the physical server with onboard networking and storage, rack-level facilities such as PDU and KVM switch, and basic cooling equipment including fans and environmental sensors.

Table 5.10: Capital Expenditures for On-Premises Server Infrastructure

Category	Quantity	Component Summary	Cost (€)	Cost ($)
Server + Total Storage (Internal) + Networking (Onboard NICs)	1	Intel® Xeon® E-2468, 32GB ECC DDR5, 2TB HDD, 700W PSU	2,293	2,606.82
Facilities per rack	1	Rack, PDU, KVM Switch, Power Cable, Blanking Panel	465	528.41
Cooling Equipment	1	Intake Fan, Rear Exhaust Fan, Temp/Humidity Sensor	421	478.41
Capital Expenditures	–	–	3,179.00	3,613.64

C. Operational Expenditures (OpEx)

According to the methodology outlined by Richard Sawyer, the rated power of the server is taken directly from the PSU specification at 700W. Actual power consumption during normal operations is estimated at 67% of the rated power, yielding a real usage of 469W. The cooling power requirements, particularly under DX cooling systems, are estimated as 125% of the IT load, which amounts to 586W. Using an electricity rate of €0.46 per kWh, the yearly energy costs are calculated. Additionally, real estate and environmental monitoring costs are included in the operational expense.

Table 5.11: Three-Year Operational Expenditures of On-Premises Server

OpEx (3 years)	Power Consumption	Rate	Annual Cost (€)	Annual Cost ($)	Three Year Cost (€)	Three Year Cost ($)
Actual Operating Power	469 Watts/server	€0.46/kWh	€1,889.43	$2,144.81	€5,668.29	$6,434.43
Actual Cooling Power	586 Watts/server	€0.46/kWh	€2,360.13	$2,681.97	€7,080.39	$8,045.92
Real Estate Rent (1 sq.m)	€10/sq.m/month	€10/month	€120	$136.36	€360	$409.09

Total Operating Expenditures	–	–	€4,369.56	$4,963.14	€13,108.29	$14,889.43

D. Cloud Provider Cost Comparison (Frankfurt Region)

This section compares the cost implications of using cloud infrastructure services from AWS, Azure, and GCP over a three-year period in the Frankfurt region. Each provider offers comparable VCIs specifications, managed disks, and data transfer capacities.

Table 5.12: Cloud Provider Pricing Comparison in Frankfurt Region

Category	Detail	AWS	Azure	GCP
Compute	VM Type	t3a.2xlarge	D8as v5	N2-standard-8
	Monthly Cost ($)	95	115	165.44
	3-Year Total ($)	3415	3,650	5,955
Storage	Managed Disk (2 TB)	110	82	82
2 TB	3-Year Total ($)	3960	2952	2952
Transfer Cost	Monthly ($)	20.48	49.75	66.56
1 TB	3-Year Total ($)	737	1791	2396
Total Cost	Monthly Total ($)	225.5	246.75	314
	3-Year Total ($)	8,117	8,883	11,304

E. Cost Summary of On-Premises vs Public Cloud

Table 5.13: Total Cost Summary: On-Premises vs Public Cloud Providers

Period	Initial investment + OpEx (in euros)	AWS pricing	Azure Pricing	GCP Pricing
Start	3,613.64	0	0	0
1st Year	4,963.14	2,705	2961	3768
2nd Year	4,963.14	2,705	2961	3768
3rd Year	4,963.14	2,705	2961	3768
Total	18,503.06	**8,117**	**8883**	**11304**

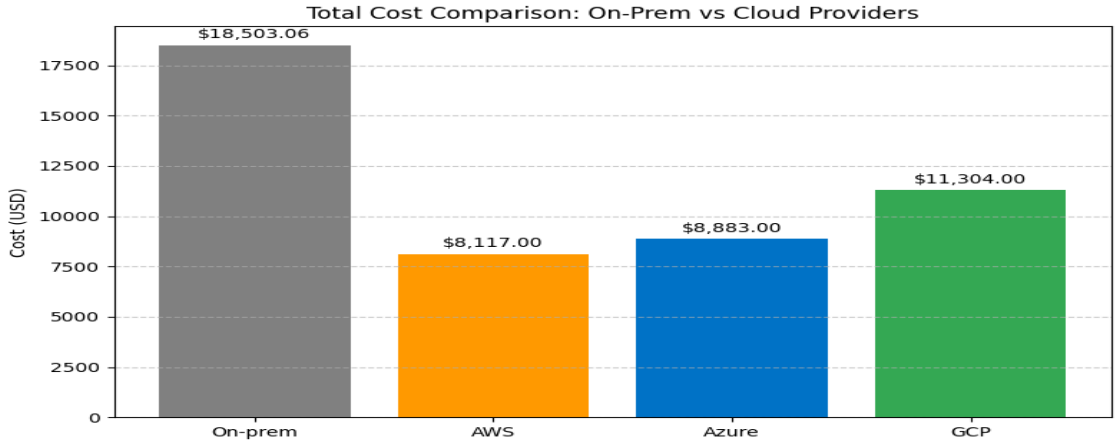

Figure 5.4: Total Cost Summary: On-Premises vs Public Cloud Providers

F. Analysis and Findings of on prem vs cloud

Setting up an on-premises server infrastructure requires a significant initial capital expenditure of $3,613.64, which includes physical server hardware, rack-level facilities, and basic cooling equipment [Table 5.10]. In addition to this upfront investment, the operational expenditures over a three-year period are estimated at $14,889.43, primarily driven by electricity consumption, cooling requirements, and real estate overheads [Table 5.11]. This results in a total cost of ownership (TCO) of approximately $18,503.06 over the three-year period [Table 5.13].
In contrast, cloud computing eliminates the need for capital investment and follows a pay-as-you-go billing model. A cost comparison of major cloud service providers in the Frankfurt region over a three-year term reveals the following:

- Amazon Web Services (AWS) incurs a total cost of $8,117, making it the most cost-effective solution, representing a 56% reduction compared to the on-premises setup.
- Microsoft Azure results in a total expenditure of $8,883, translating to approximately 52% savings relative to the on-prem model.
- Google Cloud Platform (GCP), although the most expensive among the three at $11,304, still offers a 39% cost reduction over on-premises infrastructure [Table 5.12, Table 5.13].

In addition to financial benefits, cloud services offer several operational advantages.

- **Elastic scalability**: resources can be increased or decreased as needed without hardware replacement.
- **No maintenance burden**: providers manage hardware, power, and cooling infrastructure.
- **Reduced staffing requirements**: minimal on-site IT personnel needed.
- **Improved reliability and fault tolerance**, with built-in failover and backup mechanisms.

5.2 Cost Optimization Strategy Implementation

In the SLR, studies emphasize the importance of selecting appropriate instance types and purchase models as a key strategy for cost optimization in cloud and hybrid infrastructures. Specifically, combining Reserved Instances**,** On-Demand**,** and Spot Instances based on workload predictability is consistently highlights as a high-impact approach for reducing overall infrastructure expenditure.

To validate this, a cost analysis is conducted using the official pricing calculators of three major cloud providers: AWS, Azure, and GCP. Although this study did not include a real deployment, the analysis provides useful insights. It shows how choosing the right instances based on workload predictability can lead to significant cost savings in cloud environments.

5.2.1 Use Case: ML-Based Web Application Architecture

The selected architecture represents a realistic and widely adopted deployment pattern for machine learning based web applications. In this use case, the system hosts a machine learning web application with functionalities for both model training and serving predictions through a web interface. The architectural design includes:

- A webserver component accessible to users through a Load Balancer, ensuring high availability and scalability.
- A machine learning model training component**,** which involves periodic execution of compute-intensive training jobs.
- Cloud object storage, used to store trained model artifacts and any supporting datasets or intermediate outputs.

The primary justification for the chosen architecture is based on the operational requirements and behaviour of a typical ML application:

- **Webserver Hosting**: One virtual machine is always running to host the webserver application. A second instance is conditionally used to handle increased traffic, coordinated through a Load Balancer ,ensuring reliability and responsiveness.
- **ML Model Training**: Requires a separate instance due to its high CPU/memory demands.

5.2.2 Two-Phase Deployment Strategy

- **Phase 1**: All resources are provisioned using On-Demand, including those used for model training.
- **Phase 2**: A cost-optimized mix of Reserved(equivalents)**,** On-Demand/Pay-as-you-go/Regular**,** and Spot instances is used to better align infrastructure costs with workload patterns.

This section evaluates how smarter instance selection, based on workload predictability and importance, can lead to substantial cost reductions, without sacrificing system availability or performance. All cost estimates are generated using the pricing calculators [45][46][47].

A. Phase 1: Baseline On-Demand Instance Deployment (Frankfurt Region)

In this phase, all compute resources, including those for the webserver and training workloads, are provisioned using On-Demand instances:

- **AWS**: 3 × t3a.2xlarge EC2 instances behind an ALB (2 for webserver, 1 for training)
- **Azure**: 3 × D8as v5 Pay-as-you-go VMs behind Azure Load Balancer
- **GCP**: 3 × n2-standard-8 Compute Engine behind GCP Load Balancer

The On-Demand deployment model offers several advantages that make it attractive for organizations seeking flexibility and reliability. One of the key benefits is stability, as On-Demand (or Pay-as-you-go) instances are not subject to unexpected interruptions, ensuring consistent performance. This model also promotes simplicity, eliminating the need for upfront commitments or complex bid-based allocation strategies, which can complicate infrastructure planning. Additionally, On-Demand deployments support high availability through load balancers that distribute traffic across multiple availability zones, enhancing fault tolerance and ensuring continuous service delivery. However, this comes at a higher cost, especially when training workloads are compute-intensive and not continuous. Cost estimation using pricing calculators is shown in the below table.

Table 5.14: Baseline On-Demand Deployment Costs

Resource	AWS (t3a.2xlarge)	Azure (D8as v5)	GCP (n2-standard-8)
Compute Instances	3 × t3a.2xlarge	3 × D8as v5	3 × n2-standard-8
Cost/hr (each)	$0.3645	$0.416	$0.5005
Monthly Compute	$262.44 × 3 = $787.32	$303.68 × 3 = $911.04	$360.36 × 3 = $1081.08
Load Balancer	$0.027/hr (~$19.44)	$0.025/hr (~$18.00)	$0.025/hr (~$18.00)
Storage (100 GB)	$0.0245/GB (~$2.45)	$0.0196/GB (~$1.96)	$0.023/GB (~$2.30)
Total Monthly Cost	~$809.21	~$931.00	~$1101.30

B. Phase 2: Mixed Pricing Deployment with Cost Optimization

This strategy introduces cost efficiency by mapping workloads to the most suitable pricing models:

- **1 Reserved Instance**: For the always-on webserver.
- **1 On-Demand Instance**: For burst traffic or failover redundancy.
- **1 Spot Instance**: For non-critical ML training tasks.

The mixed pricing model offers a balanced and cost-effective strategy by combining different cloud instance types based on workload characteristics. Reserved Instances are ideal for stable, always-on workloads, providing predictable cost savings through long-term commitments. On-Demand Instances complement this setup by enabling dynamic scaling and enhancing fault tolerance, making them suitable for variable or unexpected workloads. Spot Instances contribute significant cost reductions up to 90% and are particularly well-suited for flexible, non-critical, and interruptible tasks such as machine learning model training. This hybrid approach maximizes both performance and budget efficiency. Cost estimation using Pricing Calculators

Table 5.15: Baseline Mixed Deployment Costs

Resource	AWS (t3a.2xlarge)	Azure (D8as v5)	GCP(n2-standard-8)
Reserved Instance	1 × t3a.2xlarge (RI) = $0.2177/hr.	1 × Reserved VM = $0.183/hr.	1 × Committed Use = $0.315/hr.
Monthly Cost (RI)	$156.74	$133.58	$226.80
On-Demand/Pay-as-you-go/Regular	1 × OD = $0.3645/hr.	1 × PAYG = $0.416/hr.	1 × Regular= $0.5005/hr.
Monthly Cost (On-Demand/Pay-as-you-go/Regular)	$262.44	$303.68	$360.36
Spot Instance	1 × Spot= ~$0.1246/hr.	1 × Spot = ~$0.07/hr.	1 × Spot = ~$0.1447/hr.
Monthly Cost (Spot)	~$91.85	~$51.02	~$104.18
Load Balancer	ALB – $0.027/hr. ($19.44)	Azure LB = $0.025/hr. ($18.00)	GCP LB – $0.025/hr. ($18.00)
Storage (100 GB)	S3= $0.0245/GB ($2.45)	Blob = $0.0196/GB ($1.96)	Cloud = $0.023/GB ($2.30)
Total Cost	$530.78	$508.00	$711.64

C. Comparative Analysis

A side-by-side evaluation of both deployment phases reveals a clear financial advantage in adopting a mixed pricing strategy:

Table 5.16: Phase-wise Deployment Cost and Savings

Cloud Provider	Phase 1 Cost	Phase 2 Cost	Cost Reduction	% Savings
AWS	$809.21	$530.78	$278.43	34.4%
Azure	$931.00	$508.00	$423.00	45.4%
GCP	$1101.30	$711.64	$389.66	35.4%

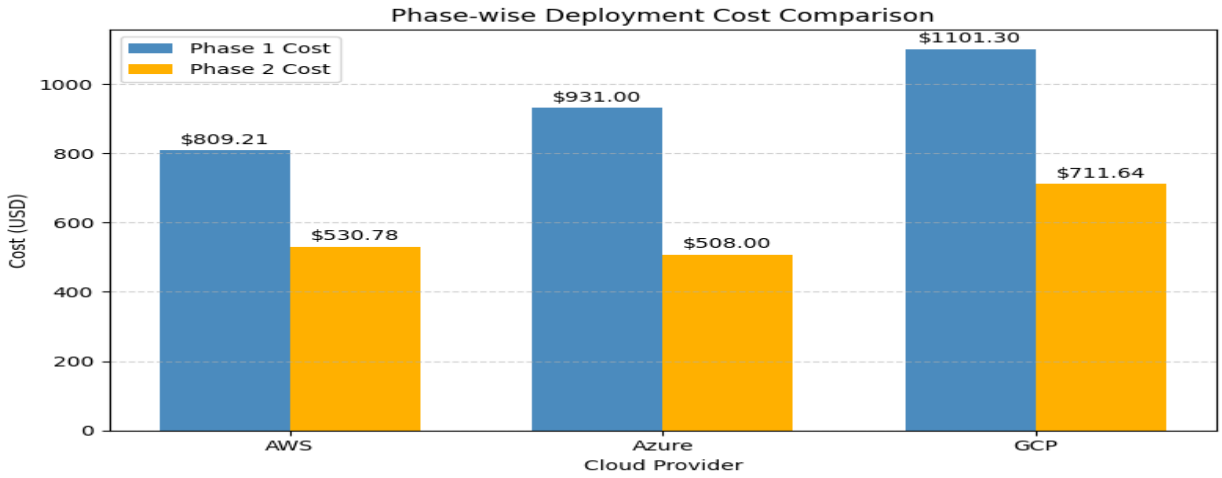

Figure 5.5: Phase-wise Deployment Costs

This approach results in a 30%–45% cost reduction by intelligently mapping predictable workloads to Reserved Instances and leveraging Spot Instances for tasks like training. The On-Demand instance adds flexibility, acting as a buffer for traffic or failures.

6 Cloud Cost Control APIs

Cloud computing offers scalability and flexibility. However, this convenience often leads to unpredictable or high billing. Cloud cost control includes the practices and tools used to monitor, manage, and improve spending in cloud environments. Organizations often face challenges like unclear billing statements, idle or underused resources, and uncontrolled usage that exceeds budgets.

Given the dynamic and elastic nature of cloud services, manual tracking of usage and enforcing budgets is insufficient. Cloud environments need automated solutions that can monitor usage in real time and manage costs proactively.

Application Programming Interfaces (APIs) address the limitations of manual cost management by offering programmable, scalable access to billing and usage data . Through APIs, organizations can:

- Automate cost tracking and reporting in real-time,
- Integrate cloud spend data into dashboards (e.g., Grafana, Power BI),
- Enforce budget thresholds through notifications and corrective actions.

This section looks at how AWS, Azure, and GCP support cost management through their official APIs. To do this, it reviews documentation and publicly available resources from each provider, focusing on how they help users monitor and control cloud spending. All insights are drawn from trusted, official sources to ensure accuracy and relevance.

6.1 AWS Cost Control APIs

AWS provides a set of APIs that enable users to programmatically monitor, forecast, and optimize cloud costs. This analysis is based on a detailed review of over 1,500 pages of official AWS documentation, from which relevant information on cost control APIs was extracted. The following subsections highlight key cost control APIs offered by AWS, their functionalities, supported operations, and usage scenarios [51].

6.1.1 AWS Cost Explorer API

The Cost Explorer API allows users to retrieve historical AWS usage and cost, enabling analysis, forecasting, anomaly detection, and savings recommendations.

Key Actions and Examples

Table 6.1: AWS Cost Explorer API

Action	Description	AWS CLI	Python SDK(Boto3)
GetCostAndUsage	Retrieves historical data on usage and cost by service, account, or tag.	aws ce get-cost-and-usage	client.get_cost_and_usage()

GetCostForecast	Predicts future AWS costs using historical trends.	aws ce get-cost-forecast	client.get_cost_forecast()
GetAnomalies	Identifies unusual spikes or dips in usage or spend.	aws ce get-anomalies	client.get_anomalies()
CreateAnomalyMonitor	Sets rules to track cost anomalies for specific accounts or services.	aws ce create-anomaly-monitor	client.create_anomaly_monitor()
CreateAnomalySubscription	Subscribes users to receive notifications when anomalies are detected.	aws ce create-anomaly-subscription	client.create_anomaly_subscription()
GetReservationCoverage	Shows how much usage is covered by existing Reserved Instances.	aws ce get-reservation-coverage	client.get_reservation_coverage()
GetReservationUtilization	Measures how well your Reserved Instances are being used.	aws ce get-reservation-utilization	client.get_reservation_utilization()
GetRightsizingRecommendation	Suggests better instance types for saving costs.	aws ce get-rightsizing-recommendation	client.get_rightsizing_recommendation()
GetSavingsPlansUtilization	Evaluates the effectiveness of current Savings Plans.	aws ce get-savings-plans-utilization	client.get_savings_plans_utilization()
GetSavingsPlansPurchaseRecommendation	Recommends new Savings Plans to reduce costs.	aws ce get-savings-plans-purchase-recommendation	client.get_savings_plans_purchase_recommendation()

```
import boto3
client = boto3.client('ce')  # 'ce'is Cost Explorer
response = client.get_cost_and_usage(
    TimePeriod={'Start': '2025-06-01', 'End': '2025-06-30'},
    Granularity='DAILY',
    Metrics=['UnblendedCost']
)
```

Listing 6.1: Python example using Boto3 to retrieve AWS Cost Explorer data

Use Cases:

1. Cost and usage trend analysis
2. Resource rightsizing
3. Forecasting cloud spend
4. Anomaly detection and alerting

Endpoint:

```
https://ce.us-east-1.amazonaws.com
```

6.1.2 AWS Budgets API

The Budgets API allows users to set financial limits through cost, usage, RI, or Savings Plans budgets. It also supports automatic actions like applying IAM policies or deactivating resources.

Key Actions

Table 6.2: AWS Budgets API

Action	Description	AWS CLI	Python (boto3)
CreateBudget	Define budgets for cost, usage, RI, or Savings Plans.	aws budgets create-budget --account-id <id> --budget file://budget.json	client.create_budget(AccountId='123456789012', Budget={...})
DescribeBudget	View current budget configurations.	aws budgets describe-budget --account-id <id> --budget-name <name>	client.describe_budget(AccountId='123456789012', BudgetName='MyBudget')
UpdateBudget	Modify thresholds or parameters for existing budgets.	aws budgets update-budget --account-id <id> --budget file://budget.json	client.update_budget(AccountId='123456789012', NewBudget={...})
CreateBudgetAction	Set automatic responses (e.g., alerts or resource shutdown) when a budget is exceeded.	aws budgets create-budget-action --account-id <id> --budget-name <name> --action file://action.json	client.create_budget_action(AccountId='123456789012', BudgetName='MyBudget', NotificationType='ACTUAL', ActionType='APPLY_IAM_POLICY', ActionThreshold={...}, Subscribers=[...])

```
import boto3
client = boto3.client('budgets')

response = client.create_budget(
    AccountId='123456789012',
    Budget={
        'BudgetName': 'MyBudget',
        'BudgetLimit': {'Amount': '1000', 'Unit': 'USD'},
        'TimeUnit': 'MONTHLY',
        'BudgetType': 'COST'
    }
)
```

Listing 6.2: Python example using Boto3 to retrieve Budget API

Use Cases:

1. Set and manage cost limits
2. Get automated alerts and take actions when budgets are reached

3. Control spending by team or project

Endpoint:

```
https://budgets.amazonaws.com
```

6.1.3 AWS Cost Optimization Hub API

The CUR API creates detailed CSV reports for cost allocation, and third-party integrations.

Key Actions and Examples

Table 6.3: AWS Cost Optimization Hub API

Action	Description	AWS CLI (via aws cost-optimization-hub)	Python (boto3)
GetRecommendation	Retrieve targeted cost-saving suggestions (e.g., idle resources).	aws cost-optimization-hub get-recommendation --resource-id <id>	client.get_recommendation(ResourceId='i-12345678')
ListRecommendations	List multiple optimization opportunities across services.	aws cost-optimization-hub list-recommendations	client.list_recommendations()

```
import boto3
client = boto3.client('cost-optimization-hub')
response = client.list_recommendations()
```

Listing 6.3: Python example using Boto3 to retrieve Cost Optimization Hub API

Use Cases:

1. Organization-wide cost-saving insights
2. FinOps visibility and reporting

Endpoint:

```
https://cost-optimization-hub.us-east-1.amazonaws.com
```

6.1.4 AWS Cost and Usage Report (CUR) API

The CUR API generates high-granularity CSV reports for cost allocation, reconciliation, and third-party integrations.

Key Actions and Examples

Table 6.4: AWS Cost and Usage Report (CUR) API

Action	Description	AWS CLI	Python (boto3)
PutReportDefinition	Define what cost and usage data should be included in your reports.	aws cur put-report-definition --report-definition file://report.json	client.put_report_definition(ReportDefinition={...})
DescribeReportDefinitions	View configurations for existing usage reports.	aws cur describe-report-definitions	client.describe_report_definitions()
DeleteReportDefinition	Remove obsolete or unused report definitions.	aws cur delete-report-definition --report-name <name>	client.delete_report_definition(ReportName='MyReport')

```
import boto3
client = boto3.client('cur')

response = client.put_report_definition(
    ReportDefinition={
        'ReportName': 'MyCUR',
        'TimeUnit': 'DAILY',
        'Format': 'textORcsv',
        'Compression': 'GZIP',
        'S3Bucket': 'my-bucket',
        'S3Prefix': 'reports/',
        'S3Region': 'us-east-1',
        'AdditionalSchemaElements': ['RESOURCES']
    }
)
```

Listing 6.4: Python example using Boto3 to retrieve Cost and Usage Report (CUR) API

Use Cases:

1. Check and review billing data to make sure it's correct.
2. Connect with reporting or analytics tools
3. Create detailed cost reports for teams or departments

Endpoint:

```
https://cur.us-east-1.amazonaws.com
```

6.1.5 AWS Price List API

The Price List API offers real-time pricing details for AWS services including EC2, RDS, and storage

Key Actions

Table 6.5: AWS Price List API

Action	Description	AWS CLI	Python (boto3)
GetProducts	Query real-time pricing for specific SKUs, regions, or services.	aws pricing get-products --service-code AmazonEC2 --filters Type=TERM_MATCH,Field=location,Value=Frankfurt	client.get_products(ServiceCode='AmazonEC2', Filters=[...])
DescribeServices	List AWS services available in pricing data.	aws pricing describe-services	client.describe_services()
GetAttributeValues	Get filterable attributes like instance type or OS.	aws pricing get-attribute-values --service-code AmazonEC2 --attribute-name instanceType	client.get_attribute_values(ServiceCode='AmazonEC2', AttributeName='instanceType')

```
import boto3
client = boto3.client('pricing', region_name='us-east-1')

response = client.get_products(
    ServiceCode='AmazonEC2',
    Filters=[
        {'Type': 'TERM_MATCH', 'Field': 'location', 'Value': 'Frankfurt'}
    ]
)
```

Listing 6.5: Python example using Boto3 to retrieve Price List API

Use Cases

1. Real-time price lookup
2. Build cost estimation tools
3. Track pricing trends for services

Endpoint:

```
https://api.pricing.us-east-1.amazonaws.com
```

6.1.6 Summary of AWS Cost APIs

Table 6.6: Summary of AWS Cost APIs

API Name	Purpose	Key Use Cases
Cost Explorer API	Analyze, forecast, and optimize historical usage and spend	Trend analysis, forecasting, anomaly detection
Budgets API	Define budgets and trigger alerts or actions on threshold breach	Enforce limits, automate cost responses
Cost Optimization Hub API	Centralize and aggregate cost-saving recommendations	Multi-account FinOps insights
Cost and Usage Report API	Export detailed usage and billing data for analytics	BI integration, auditing, reporting
Price List API	Retrieve real-time pricing data by filters	Build calculators, dynamic cost models

6.1.7 Developer Access Methods

These APIs can be accessed via:

- AWS CLI
- SDKs in Python (boto3), Java, JavaScript, Go, .NET, PHP, Ruby, and Kotlin
- IAM permissions for programmatic usage

6.2 Azure Cost Control APIs

This section provides a structured analysis of Azure's APIs relevant to cost control, categorized based on their functional objectives such as billing, pricing, budgeting, and reservation optimization [52].

6.2.1 Cost Details APIs

Tracking actual consumption and cost is the foundation of cloud financial operations. Azure provides APIs to retrieve detailed billing records and export reports for automated analysis. These APIs help teams understand current spend and establish reporting pipelines..

A. Exports API

- **Purpose**: This API allows users to automatically export their cost and usage data to Azure Blob Storage on a recurring basis daily, weekly, or monthly.
- **Importance**: For large companies using many cloud services, it becomes difficult to track costs manually. This API provides an automated solution to push detailed cost reports to storage, which can then be analyzed using business intelligence tools like Power BI or Excel.
- **Benefits**:
    - Suitable for enterprise environments managing large volumes of data

    - Enables scheduled, hands-off reporting
    - Easy to integrate with dashboards or analytics tools
- **Common Use Cases**:
    - Feeding cost data into daily reporting dashboards
    - Tracking monthly spend across multiple departments or subscriptions

B. Generate Cost Details API

- **Purpose**: This API provides a way to download cost details on-demand for a specified date range.
- **Importance**: If quick analysis is needed recent costs without waiting for scheduled reports, this API is suitable for quick checks or project-specific budgeting.
- **Benefits**:
    - Provides rapid and configurable access to cost data
    - Useful for teams that want insights without setting up automation
- **Common Use Cases**:
    - Manually downloading a week's worth of cost data for review
    - Tracking project-specific expenses on demand

6.2.2 Pricing APIs

Understanding how much services cost is essential before deploying them. Azure provides pricing APIs to retrieve real-time pricing information.

A. Azure Retail Prices API

- **Purpose**: This API gives access to Azure's public pay-as-you-go pricing data, also known as retail prices.
- **Importance**: Before using any resource, it is useful for estimating how much it costs. This API can be used to build calculators or simulate billing for planned deployments.
- **Benefits**:
    - Real-time pricing for all Azure services
    - Helps developers and finance teams predict expenses before provisioning resources
- **Common Use Cases**:
    - Powering internal pricing dashboards
    - Estimating deployment costs based on current prices

B. Price Sheet API

- **Purpose**: This API returns the custom pricing that organizations may have as part of their enterprise agreements with Microsoft.
- **Importance**: Many large companies negotiate special prices. This API ensures that those discounts are factored into financial forecasts and billing estimates.
- **Benefits**:
    - Enables accurate cost estimation based on organization-specific contracts
    - Supports large-scale financial planning
- **Common Use Cases**:
    - Calculating invoices using enterprise pricing

- Cross-checking billed amounts against agreed-upon rates

6.2.3 Budgets and Alerts APIs

Budgeting is essential for avoiding unexpected costs on monthly cloud bills. Azure provides APIs that enable the definition of spending limits and the generation of notifications when those limits are approached or exceeded.

A. Budgets API

- **Purpose**: This API lets users define cost or usage budgets and track progress against those limits.
- **Importance**: Without set budgets, costs can grow silently. This API helps set boundaries and monitor how much is spent in real time.
- **Benefits**:
  - Create budgets for resources, resource groups, or subscriptions
  - Automate spending oversight
- **Common Use Cases**:
  - Setting monthly budgets for different departments
  - Monitoring team or project-level cloud spends

B. Alerts API

- **Purpose**: Works with the Budgets API to send alerts when spending limits are reached.
- **Importance**: Budgets are only useful if someone knows they have been exceeded. This API ensures that alerts are sent to the right people or systems in real time.
- **Benefits**:
  - Keeps teams informed when spending gets close to or exceeds limits
  - Supports automation of corrective actions
- **Common Use Cases**:
  - Sending emails or messages to cloud admins on overspend
  - Triggering automated scripts to shut down non-critical services when budgets are breached

6.2.4 Cost Optimization APIs (Reservation Management)

Azure offers discounted pricing for long-term service commitments, such as one or three years. These are referred to as Reserved Instances, and the associated APIs enable the management and optimization of such purchases.

A. Reservation Recommendations API

- **Purpose**: This API suggests Azure services or resources that are suitable for reservation to reduce costs.
- **Importance**: Rather than consistently incurring higher on-demand rates, this API recommends long-term reservation options based on observed usage patterns.
- **Benefits**:

    - Increases cost efficiency
    - Encourages proactive cost planning
- **Common Use Cases**:
    - Recommending reservations for virtual machines based on past usage
    - Financial planning for upcoming quarters

B. Reservation Recommendation Details API

- **Purpose**: Provides deeper insights to simulate the financial impact of accepting a reservation recommendation.
- **Importance**: Before committing to a long-term plan, it is important to evaluate the potential savings and risks. This API helps to facilitates informed decision-making.
- **Benefits**:
    - Allows for informed reservation decisions
    - Supports detailed financial simulations
- **Common Use Cases**:
    - Evaluating how much can be saved by reserving a database or VM
    - Comparing different reservation durations (1 year vs. 3 years)

6.2.5 API Endpoint Summary Table

Table 6.7: Azure Summary of API Endpoints

API	**Operation**	**HTTP Method**	**API Endpoint**
Exports API	Create Or Update	PUT	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/exports/{exportName}?api-version=2025-03-01
Exports API	List	GET	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/exports?api-version=2025-03-01
Exports API	Delete	DELETE	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/exports/{exportName}?api-version=2025-03-01
Exports API	Execute	POST	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/exports/{exportName}/run?api-version=2025-03-01
Exports API	Get	GET	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/exports/{exportName}?api-version=2025-03-01
Generate Cost Details API	Create	POST	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/generateCostDetailsReport?api-version=2025-03-01
Generate Cost Details API	Get	GET	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/costDetailsOperationResults/{operationId}?api-version=2025-03-01
Azure Retail Prices API	GetAzureRetailPrices	GET	https://prices.azure.com/api/retail/prices
Price Sheet API	Get	GET	https://management.azure.com/subscriptions/{subscriptionId}/providers/Microsoft.Consumption/pricesheets/default?api-version=2024-08-01

Price Sheet API	GetByBillingPeriod	GET	https://management.azure.com/subscriptions/{subscriptionId}/providers/Microsoft.Billing/billingPeriods/{billingPeriodName}/providers/Microsoft.Consumption/pricesheets/default?api-version=2024-08-01
Budgets API	Create	PUT	https://management.azure.com/{scope}/providers/Microsoft.Consumption/budgets/{budgetName}?api-version=2024-08-01
Budgets API	Update	PUT	https://management.azure.com/{scope}/providers/Microsoft.Consumption/budgets/{budgetName}?api-version=2024-08-01
Budgets API	Get	GET	https://management.azure.com/{scope}/providers/Microsoft.Consumption/budgets?api-version=2024-08-01
Budgets API	List	GET	https://management.azure.com/{scope}/providers/Microsoft.Consumption/budgets/{budgetName}?api-version=2024-08-01
Budgets API	Delete	DELETE	https://management.azure.com/{scope}/providers/Microsoft.Consumption/budgets/{budgetName}?api-version=2024-08-01
Alerts API	Dismiss	PATCH	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/alerts/{alertId}?api-version=2025-03-01
Alerts API	Get	GET	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/alerts/{alertId}?api-version=2025-03-01
Alerts API	List	GET	https://management.azure.com/{scope}/providers/Microsoft.CostManagement/alerts?api-version=2025-03-01
Alerts API	Get	GET	https://management.azure.com/providers/Microsoft.CostManagement/{externalCloudProviderType}/{externalCloudProviderId}/alerts?api-version=2025-03-01
Reservation Recommendations API	List	GET	https://management.azure.com/{resourceScope}/providers/Microsoft.Consumption/reservationRecommendations?api-version=2024-08-01
Reservation Recommendation Details API	Get	GET	https://management.azure.com/{resourceScope}/providers/Microsoft.Consumption/reservationRecommendationDetails?api-version=2024-08-01&scope={scope}&region={region}&term={term}&lookBackPeriod={lookBackPeriod}&product={product}

6.2.6 Developer Tools and SDK Access

To work programmatically with Azure Cost APIs, developers can use official SDKs and command-line tools [53].

- **Azure Consumption CLI**
  - Tool: az consumption
  - Installation: az extension add --name consumption
- **Azure Consumption Python SDK**
  - Package**:** azure-mgmt-consumption
  - Installation**:** pip install azure-mgmt-consumption
- **Azure Consumption Node.js SDK**
  - Package**:** @azure/arm-consumption
  - Installation**:** npm install @azure/arm-consumption
- **Azure Consumption Ruby SDK**
  - Gem: azure_mgmt_consumption
  - Installation**:** gem install azure_mgmt_consumption

6.2.7 CLI Access for Cost Control APIs

Table 6.8: Azure CLI Coverage for Cost control APIs

API Category	Operation	Azure CLI Command
Exports API	Create or update	az costmanagement export create --name --scope --storage-account-id --storage-container --timeframe {BillingMonthToDate, Custom, MonthToDate, TheLastBillingMonth, TheLastMonth, WeekToDate} [--dataset-configuration] [--recurrence {Annually, Daily, Monthly, Weekly}] [--recurrence-period] [--schedule-status {Active, Inactive}] [--storage-directory] [--time-period] [--type {ActualCost, AmortizedCost, Usage}]
Exports API	List	az costmanagement export list --scope
Exports API	Delete	az costmanagement export delete --name --scope [--yes]
Exports API	Show Export	az costmanagement export show --name --scope
Exports API	Update Export	az costmanagement export update --name --scope [--dataset-configuration] [--recurrence {Annually, Daily, Monthly, Weekly}] [--recurrence-period] [--schedule-status {Active, Inactive}] [--storage-account-id] [--storage-container] [--storage-directory] [--time-period] [--timeframe {BillingMonthToDate, Custom, MonthToDate, TheLastBillingMonth, TheLastMonth, WeekToDate}]
Generate Cost Details API	Create	Not available
Generate Cost Details API	Get Status	Not available
Retail Prices API	Get Retail Prices	Not available
Price Sheet API	Get Price Sheet	az consumption pricesheet show [--billing-period-name] [--ids] [--include-meter-details] [--subscription]
Budgets API	Create or Update	az consumption budget create --amount --budget-name --category {cost, usage} --end-date --start-date --time-grain {annually, monthly, quarterly} [--meter-filter] [--resource-filter] [--resource-group] [--resource-group-filter]
Budgets API	Delete	az consumption budget delete --budget-name [--resource-group]
Budgets API	List	az consumption budget list [--max-items] [--next-token] [--resource-group]
Budgets API	Show	az consumption budget show --budget-name [--resource-group]
Budgets API	Update	az consumption budget update [--add] [--amount] [--budget-name --name] [--category {Cost, Usage}] [--e-tag] [--filters] [--force-string {0, 1, f, false, n, no, t, true, y, yes}] [--ids] [--notifications] [--remove] [--set] [--subscription] [--time-grain {Annually, Monthly, Quarterly}] [--time-period]
Alerts API	Dismiss Alert	Not available
Reservation Recommendations API	List Recommendations	Not available
Reservation Recommendation Details API	Get Details	Not available

6.2.8 Python SDK for Cost Control APIs

The Azure SDK for Python (azure-mgmt-consumption) is a robust library designed to automate key cost-related operations within Azure. It enables developers and administrators to programmatically manage budgeting, monitor expenditures, access detailed usage data, retrieve pricing sheets, and evaluate reservation-based cost optimizations [54].

SDK Structural Overview

Table 6.9: Key Components of the Azure Python SDK

Component	Description
models	Defines data schemas used in API requests/responses (e.g., Budget, UsageDetail)
operations	Contains callable methods via ConsumptionManagementClient for interacting with Azure services

Key SDK Classes – Models and Operations

The SDK groups functionality into model classes**,** which define the structure of cost-related data, and operation classes, which provide methods to interact with Azure's cost management platform.

Table 6.10: Azure SDK Model Classes

Model Class	Purpose
Budget	Defines budget settings (amount, timeframe).
BudgetComparisonExpression	Compares actual vs forecasted spending.
BudgetFilter	Filters budgets by tags, resources, etc.
BudgetTimePeriod	Sets the start and end dates for a budget.
BudgetsListResult	Collection of all budget definitions.
CurrentSpend	Monitors current spending in real-time.
ForecastSpend	Provides projected costs for a given budget period.
Notification	Configures alerts when a budget threshold is exceeded.
UsageDetail	Itemized usage and cost records.
UsageDetailsListResult	List of detailed usage data, useful for analysis.
PriceSheetProperties	Metadata for pricing (used with enterprise accounts).
PriceSheetResult	Full pricing details returned by the Price Sheet API.
ReservationRecommendation	Suggests reserved instances to reduce costs.
ReservationRecommendationDetailsModel	Enables "what-if" analysis to estimate reservation savings.

Table 6.11: Azure SDK Operation Classes

Operations Class	Functionality
BudgetsOperations	Create, update, delete, and retrieve budgets.
UsageDetailsOperations	Retrieve detailed resource usage and associated costs.
PriceSheetOperations	Get pricing information for enterprise agreements.
ReservationRecommendationsOperations	Get suggestions for cost-saving reserved instance purchases.
ReservationRecommendationDetailsOperations	Analyze potential savings from reservation scenarios.
ReservationsSummariesOperations	Summarize how reserved instances are being used.
AggregatedCostOperations	View total spend at the subscription or resource group level.

6.3 GCP Cost Control APIs

Google Cloud provides a suite of APIs to help teams monitor, control, and automate cost governance. These include the Cloud Billing Budget API**,** Cloud Pricing API**,** and Cloud Billing Account and Catalog APIs.

6.3.1 Cloud Billing Budget API

The Cloud Billing Budget API lets users set budgets for their Google Cloud accounts and get alerts when costs go over a set limit. This helps teams manage spending and follow budget rules across services and projects [55].

Purpose: The API allows users to define budgets and receive alerts when actual or forecasted spend exceeds predefined thresholds.

Benefits

- **Flexible Threshold Support:** Budgets can be monitored against actual and forecasted spend.
- **Real-Time Notifications:** Integrates with Pub/Sub to push alerts instantly.
- **Scalable Governance:** Supports centralized control of budgets across projects and services.

Key Use Cases

- **Project-specific Cost Tracking**: Set up budgets tailored to individual projects.
- **Automated Alerts and Controls**: Trigger alerts or actions when budgets are exceeded.
- **Centralized Governance**: Set spending limits for accounts or services.

REST Operations Overview:

Table 6.12: REST Operations of the Cloud Billing Budget API

Operation	HTTP Method	Endpoint	Purpose	Request Example
List Budgets	GET	/v1/billingAccounts/{billing-account-id}/budgets	Retrieves all budgets under a billing account.	curl -X GET -H "Authorization: Bearer $(gcloud auth print-access-token)" -H "x-goog-user-project: {project-id}" "https://billingbudgets.googleapis.com/v1/billingAccounts/{billing-account-id}/budgets"
Get Budget	GET	/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}	Fetches details of a specific budget.	curl -X GET -H "Authorization: Bearer $(gcloud auth print-access-token)" -H "x-goog-user-project: {project-id}" "https://billingbudgets.googleapis.com/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}"
Create Budget	POST	/v1/billingAccounts/{billing-account-id}/budgets	Creates a new budget scoped to a project or service.	curl -X POST -H "Authorization: Bearer $(gcloud auth print-access-token)" -H "x-goog-user-project: {api-user-project-id}" -H "Content-Type: application/json" -d @request.json "https://billingbudgets.googleapis.com/v1/billingAccounts/{billing-account-id}/budgets"
Update Budget	PATCH	/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}	Modifies an existing budget (e.g., scope, threshold, amount).	curl -X PATCH -H "Authorization: Bearer $(gcloud auth print-access-token)" -H "x-goog-user-project: {api-user-project-id}" -H "Content-Type: application/json" -d @request.json "https://billingbudgets.googleapis.com/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}"
Delete Budget	DELETE	/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}	Deletes a budget permanently from the account.	curl -X DELETE -H "Authorization: Bearer $(gcloud auth print-access-token)" -H "x-goog-user-project: {project-id}" "https://billingbudgets.googleapis.com/v1/billingAccounts/{billing-account-id}/budgets/{budget-id}"

Python Client Library (Key Classes)

Table 6.13: Key Python Client Methods for BudgetServiceClient

Class Name	Method	Purpose
BudgetServiceClient	create_budget()	Creates a new budget
	get_budget()	Retrieves an existing budget
	update_budget()	Updates a budget's configuration
	delete_budget()	Deletes a specified budget
	list_budgets()	Lists all budgets under a billing account
	from_service_account_file()	Instantiates client from service account file
	from_service_account_info()	Instantiates client from service account dict

	from_service_account_json()	Instantiates client from JSON credentials
	budget_path()	Builds fully-qualified budget path
	common_billing_account_path()	Builds billing account resource path
	common_project_path()	Builds project resource path
	parse_budget_path()	Parses a full budget path into components
	parse_common_project_path()	Parses a project path into components
	get_transport_class()	Returns transport type for RPC
	get_mtls_endpoint_and_cert_source()	Returns mutual TLS endpoint and cert source (deprecated)
	__exit__()	Cleans up transport resources
BudgetServiceAsyncClient	(All methods same as above, but async)	Supports asynchronous usage via await
ListBudgetsPager	__iter__()	Enables iteration over paged budget results (sync)
ListBudgetsAsyncPager	__aiter__()	Enables iteration over paged budget results (async)

API Access Methods

To interact with the Cloud Billing Budget API, developers can choose between:

- **Client Libraries**:
  Python, Java, Node.js, Go, C#, and C++
- **RESTful HTTP Requests**:
  Suitable for advanced use cases, allowing direct API access via standard HTTP methods (GET, POST, PATCH, DELETE) and JSON.

6.3.2 Cloud Pricing API

The Google Cloud Pricing API is an important tool for cloud cost control, enabling programmatic access to up-to-date pricing information for all Google Cloud Platform (GCP) services. It allows organizations to make informed decisions about resource usage, pricing forecasts, and FinOps automation strategies [56].

Purpose and Strategic Importance

At its core, the Pricing API serves to:

- Provide public and account-specific pricing for GCP services and resources.
- Support both on-demand price queries and contract-based pricing lookups.

By enabling developers and financial teams to programmatically access granular cost details including tiered rates, regional variations, and SKU groupings the API enables smarter infrastructure provisioning and budget forecasting.

REST Operations Overview:

Below is a structured summary of the major REST resources and their corresponding API methods. These enable users to extract pricing data at various levels of specificity and for this no IAM required.

Public REST Endpoints (No IAM roles required)

Table 6.14: GCP Public REST Resources and Methods in the Cloud Pricing API

Resource	Method	Description
v2beta.services	GET /v2beta/services	List all public GCP services
v2beta.skus	GET /v2beta/skus	List all public SKUs across services
v1beta.skus.price	GET /v1beta/skus/*	Get price of a public SKU
v1beta.skus.prices	GET /v1beta/skus/*	List latest public prices for all SKUs
v1beta.skuGroups	GET /v1beta/skuGroups	List public SKU groups
v1beta.skuGroups.skus	GET /v1beta/skuGroups/*/skus	List SKUs in a public SKU group

Billing Account-Specific Endpoints (IAM roles required)

Table 6.15: Billing Account-Specific Pricing API Endpoints and Their Use Cases

Resource	Method	Use Case
v1beta.billingAccounts.services	GET /billingAccounts/*/services	Services visible under a billing account
v1beta.billingAccounts.skus	GET /billingAccounts/*/skus	Account-specific SKU metadata
v1beta.billingAccounts.skus.price	GET /billingAccounts/*/skus/*/price	Price of a SKU in the billing account
v1beta.billingAccounts.skus.prices	GET /billingAccounts/*/skus/*/prices	Tiered or negotiated prices for SKUs
v1beta.billingAccounts.skuGroups	GET /billingAccounts/*/skuGroups	List custom contract SKU groups
v1beta.billingAccounts.skuGroups.skus	GET /billingAccounts/*/skuGroups/*/skus	SKUs under a contract group

REST Resource Structure and Access Scope

The Google Cloud Pricing API is organized into modular REST resources that provide either public or account-specific pricing data. This design supports fine-grained access and flexible integration into custom billing tools, dashboards.

Table 6.16: Pricing API Resource Structure and Access Scope

Resource Name	Scope	Purpose
services / skus	Public	List all services and SKUs
skus.price / skus.prices	Public	Retrieve list prices for individual/all SKUs
billingAccounts.services	Account-specific	Discover services billed to an account
billingAccounts.skus	Account-specific	Access billing-specific SKU metadata

billingAccounts.skus.price	Account-specific	Get custom price per SKU
billingAccounts.skus.prices	Account-specific	Fetch tiered or negotiated rates
billingAccounts.skuGroups	Account-specific	View SKU groups tied to contracts
billingAccounts.skuGroups.skus	Account-specific	Explore SKUs in a contract-defined group

6.3.3 Cloud Billing Account and Catalog APIs

The Google Cloud Billing Account and Catalog APIs provide programmatic access to cost management, budget control, and real-time pricing data. These APIs are essential for developers, administrators who want to automate billing tasks, retrieve pricing information, and build cost control applications [57]. The Cloud Billing APIs can be accessed via REST or RPC protocols. The core components are

A. Cloud Billing Account API

- Retrieve a list of all billing accounts.
- Get details of a specific billing account.
- List Google Cloud projects linked to a billing account.
- Enable or disable billing on a Google Cloud project.
- Change the billing account linked to a project.

B. Cloud Billing Catalog API

- Programmatic access to public Google Cloud pricing data.
- Retrieve billable SKUs, pricing, and metadata.
- Access without needing IAM permissions (public data).

Access Control Using IAM

Google Cloud uses Identity and Access Management (IAM) to manage access to the Cloud Billing APIs. For the Billing Catalog API, no IAM permissions are required, as it provides public access to SKU and pricing metadata. In contrast, the Billing Account API requires appropriate IAM roles to perform operations such as managing billing accounts, linking projects, and modifying billing settings.

Roles and Permissions Summary

Table 6.17: GCP Common IAM Roles for Cloud Billing API Access

Role	Description
Billing Account Administrator	Full control over billing accounts
Billing Account Viewer	View billing info and recommendations
Billing Account Costs Manager	Manage budgets and view/export cost data
Project Billing Manager	Assign or disable billing for a project
Billing Account User	Associate billing accounts with projects

Creating a Billing Application

To implement a billing-aware application using the Cloud Billing Account API, developers typically set up a Google Cloud environment and establish secure API access. The standard workflow includes the following steps:

1. Set up a Google Cloud project and enable billing.
2. Download the appropriate client library.
3. Create and authorize a service account.
4. Send REST or RPC requests to interact with billing data.

Example API Usage (Cloud Billing Account API):

a) To list all billing accounts accessible to the authenticated user:

```
GET https://cloudbilling.googleapis.com/v1/billingAccounts
```

b) To retrieve details of a specific billing account:

```
GET https://cloudbilling.googleapis.com/v1/billingAccounts/{ACCOUNT_ID}
```

c) To enable billing for a specific Google Cloud project, a PUT request is sent to the billing info endpoint:

```
PUT https://cloudbilling.googleapis.com/v1/projects/{PROJECT_ID}/billingInfo
```

d) Request payload example:

```
{
"billingAccountName": "billingAccounts/{ACCOUNT_ID}"
}
```

Using the Catalog API for Price Retrieval

The Cloud Billing Catalog API provides public access to pricing and SKU information without requiring authentication.

a) List Public Services:

```
GET https://cloudbilling.googleapis.com/v1/services?key=API_KEY
```

Returns display names, IDs, and metadata for all public services.

b) List SKUs for a Service:

```
GET https://cloudbilling.googleapis.com/v1/services/{SERVICE_ID}/skus?key=API_KEY
```

Returns detailed SKU data, including:

- SKU ID, description, service category
- Usage type (e.g., OnDemand, Preemptible)
- Pricing tiers, units, currency, and aggregation level
- Available service regions

GCP Client Libraries and API Interfaces for Cost Management

Google Cloud offers multiple interfaces for cost control and billing automation, including client libraries, RESTful APIs, and RPC-based services. These tools allow developers to programmatically manage billing accounts, retrieve SKUs and pricing data, estimate costs, and enforce access controls. This section summarizes available methods across different access layers.

Python SDK Installation

```
pip install --upgrade google-cloud-billing
```

REST API Resource Methods

Table 6.18: GCP REST Resource Methods for Cloud Billing and Catalog APIs

REST Resource	Methods	Description
v1.billingAccounts	create, get, list, move, patch, getIamPolicy, setIamPolicy, testIamPermissions	Manage billing accounts and IAM policies
v1.billingAccounts.projects	list	Lists projects associated with a billing account
v1.billingAccounts.subAccounts	create, list	Create and list billing subaccounts
v1.organizations.billingAccounts	create, list, move	Manage billing accounts under organizations
v1.projects	getBillingInfo, updateBillingInfo	Get/set billing information for a project
v1.services	list	List all public cloud services
v1.services.skus	list	List all SKUs for a given service
v1beta.billingAccounts.skus	get, list	Access SKUs visible to a billing account
v1beta.billingAccounts.skus.price	get	Get latest price for a specific SKU
v1beta.billingAccounts.skus.prices	list	List latest prices for SKUs available to your billing account
v2beta.services	get, list	Get/list public cloud services
v2beta.skus	get, list	Get/list public SKUs
v1beta.skuGroups, skuGroups.skus	get, list	Retrieve SKU groups and SKUs within groups
v1beta.billingAccounts.services	get, list	Get/list services visible to a billing account

v1beta.billingAccounts.skuGroups	get, list	Get/list SKU groups under a billing account
v1beta.billingAccounts.skuGroups.skus	get, list	Get/list SKUs in a billing account's SKU group
v1beta.skus.price, skus.prices	get, list	Get/list public SKU prices
v1beta:estimateCostScenario	POST	Estimate list prices without a billing account
v1beta.billingAccounts:estimateCostScenario	POST	Estimate cost using defined billing account

RPC API Methods

Table 6.19: GCP RPC API Methods for Billing Account and Catalog Management

RPC Service Name	Method	Description
google.cloud.billing.v1.CloudBilling	CreateBillingAccount	Creates billing subaccounts
	GetBillingAccount	Retrieves information about a billing account
	GetIamPolicy	Retrieves the IAM policy for a billing account
	GetProjectBillingInfo	Retrieves billing info for a specific project
	ListBillingAccounts	Lists all billing accounts accessible to the user
	ListProjectBillingInfo	Lists all projects associated with a billing account
	MoveBillingAccount	Changes the organization a billing account belongs to
	SetIamPolicy	Sets IAM policy for a billing account
	TestIamPermissions	Tests permissions on a billing account
	UpdateBillingAccount	Updates billing account metadata
	UpdateProjectBillingInfo	Updates billing info for a project
google.cloud.billing.v1.CloudCatalog	ListServices	Lists all public cloud services
	ListSkus	Lists SKUs available for a given cloud service

7 Sky Computing: A Unified Cloud Paradigm

Sky Computing emerges as a transformative evolution of traditional cloud computing, addressing long-standing challenges around interoperability, vendor lock-in, and workload portability. The concept of Sky Computing builds upon the foundational vision proposed by John McCarthy in 1961, which imagined computation as a public utility, similar to telephone services. While modern cloud computing has achieved technical parallels to this vision, such as elastic resource provisioning and shared infrastructure, it falls short in economic and structural alignment. Today's cloud ecosystem remains isolated, dominated by proprietary APIs and services that inhibit seamless interoperability and portability across providers. This divergence from utility computing motivates the need for a new paradigm: Sky Computing. As introduced by Stoica and Shenker, Sky Computing envisions a unified, utility-like infrastructure composed of multiple heterogeneous cloud providers. The architectural framework for Sky Computing draws a parallel with the layered architecture of the Internet [58].

7.1 Architectural Overview

The architecture of Sky Computing draws inspiration from the layered model of the Internet and is composed of three foundational layers: the Compatibility Layer, the Intercloud Layer, and the Peering Layer [58].

- **Compatibility Layer**: This layer abstracts away vendor-specific implementations using open-source tools such as Kubernetes, Docker, and TensorFlow. Applications written to target these abstractions can be deployed across multiple cloud providers without modification, facilitating portability and reducing coupling to any single platform.
- **Intercloud Layer**: Like BGP in networking, the Intercloud Layer helps find resources and schedule jobs across different clouds. It follows rules set by the user—like reducing cost or latency—and decides where to run or move jobs. This smart layer helps optimize performance and cost across all cloud providers.
- **Peering Layer**: To support efficient data movement across cloud boundaries, this layer proposes reciprocal data sharing agreements among cloud providers. These peering arrangements aim to eliminate costly egress fees and improve the performance of distributed applications by facilitating low-cost, high-bandwidth transfers between clouds.

Together, these layers define the technical foundation of Sky Computing. However, realizing this vision requires more than architectural design; it depends critically on an economic and operational enabler: the Intercloud Broker.

The Sky Broker acts as an intermediary that manages billing, service selection, and job migration across cloud boundaries. While the technical challenges to implementing Sky Computing are relatively minimal, the authors emphasize that the primary obstacles lie in establishing the necessary economic incentives and collaboration among cloud providers. Nonetheless, the potential benefits are significant: seamless cross-cloud interoperability, reduced costs, improved resilience, and the emergence of specialized cloud services that drive market innovation.

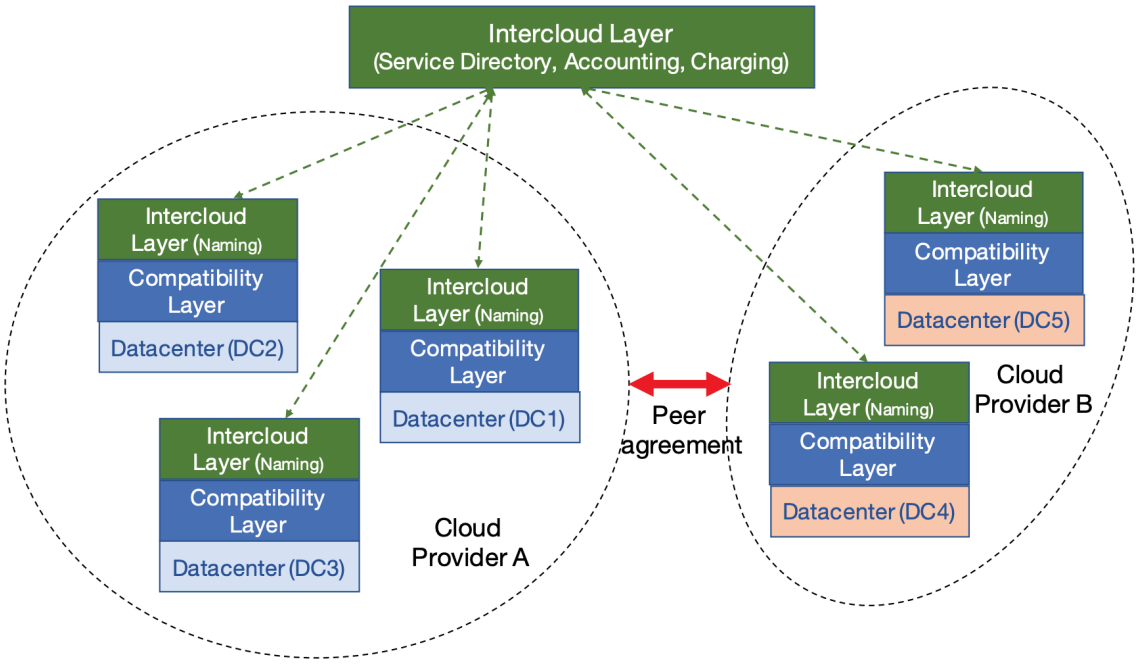

Figure 7.1: Sky computing architecture

Source: [58]

7.2 The Intercloud Broker

At the core of the Sky Computing model lies the Intercloud Broker. This important component connects user jobs to services across different cloud providers. Instead of making users choose and set up services manually, the broker automatically finds the right services, schedules workloads, and helps reduce costs

As illustrated in Figure 7.2, the intercloud broker is composed of several core components [20].

- **Identity and Access Management (IAM)**: Ensures secure authentication and authorization for cross-cloud job execution.
- **Job API**: Accepts user-submitted jobs often expressed as Directed Acyclic Graphs (DAGs) along with constraints such as cost, performance, or regional preferences.
- **Service Catalog**: Maintains metadata on service availability, compatibility, pricing, and performance metrics across multiple clouds.
- **Optimizer**: Translates job-level constraints into execution plans, using optimization techniques such as Integer Linear Programming (ILP).
- **Provisioner and Executor**: Allocates cloud resources and orchestrates execution workflows, including error recovery and monitoring.
- **Data Orchestration**: Manages the replication and movement of datasets between clouds to ensure availability where needed.
- **Billing Module (optional)**: Aggregates usage metrics and manages multi-cloud billing, either internally or through delegated systems.

This broker-centric architecture transforms Sky Computing into a two-sided marketplace, matching user needs with cloud resources using detailed rules. It hides the complexity of using multiple clouds and makes it easier to manage and scale workloads based on user-defined policies.

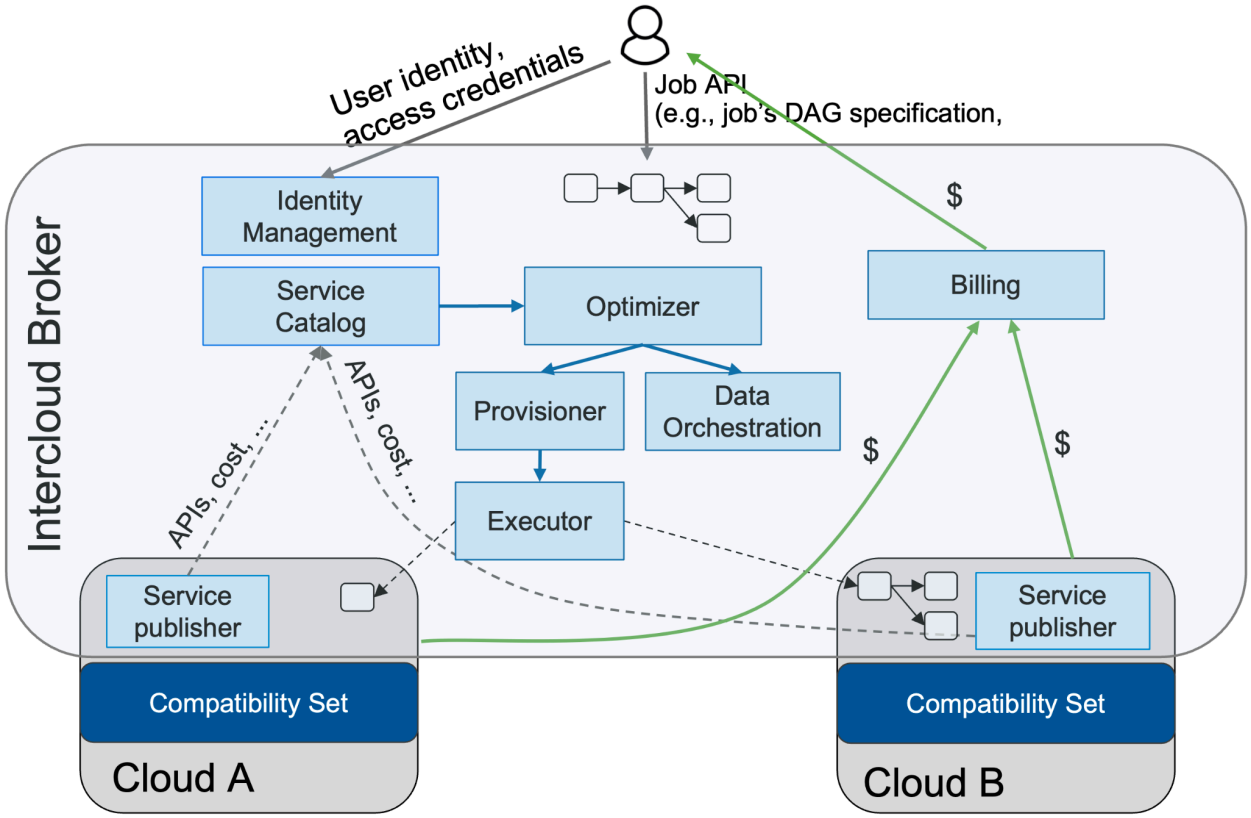

Figure 7.2: sky computing intercloud broker architecture

Source: [20]

7.3 Sky Pilot: A Practical Realization

The conceptual framework of the Intercloud Broker has been practically realized in Sky Pilot, a research system developed at UC Berkeley to demonstrate the viability of Sky Computing in real-world scenarios [59]. Sky Pilot serves as a concrete realization of the Intercloud Broker abstraction proposed within the Sky Computing paradigm. It is specifically designed to automate the placement and execution of computational batch jobs across multiple cloud providers, such as AWS, GCP, and Azure, with the goal of optimizing for user-defined objectives like cost, performance, and availability.

Sky Pilot enables users to submit computational jobs specified as directed acyclic graphs (DAGs) alongside high-level preferences (e.g., “minimize cost” or “minimize time”). Internally, the system abstracts away the heterogeneity of cloud APIs and resources, dynamically determining the most efficient and cost-effective deployment plan. This is accomplished through a modular architecture that includes the following components:

- **Service Catalog**: Maintains up-to-date information on available compute/storage services, hardware configurations, and long-term pricing across supported cloud providers.
- **Optimizer**: Utilizes an integer linear programming (ILP) formulation to determine an optimal execution plan based on user objectives and the availability of compatible resources.
- **Provisioner and Executor**: Handles the orchestration of resource allocation, job execution, and task lifecycle management, including automatic failover and retries.
- **Tracker**: Monitors live cloud resource availability, instance preemptions, and real-time pricing to enable re-optimization during job execution.

7.3.1 Experimental Validation

To validate its design, the authors conducted a real-world experiments on heterogeneous workloads spanning machine learning. These experiments showcase how SkyPilot selects optimal execution environments, handles runtime uncertainties, and exploits cross-cloud trade-offs.

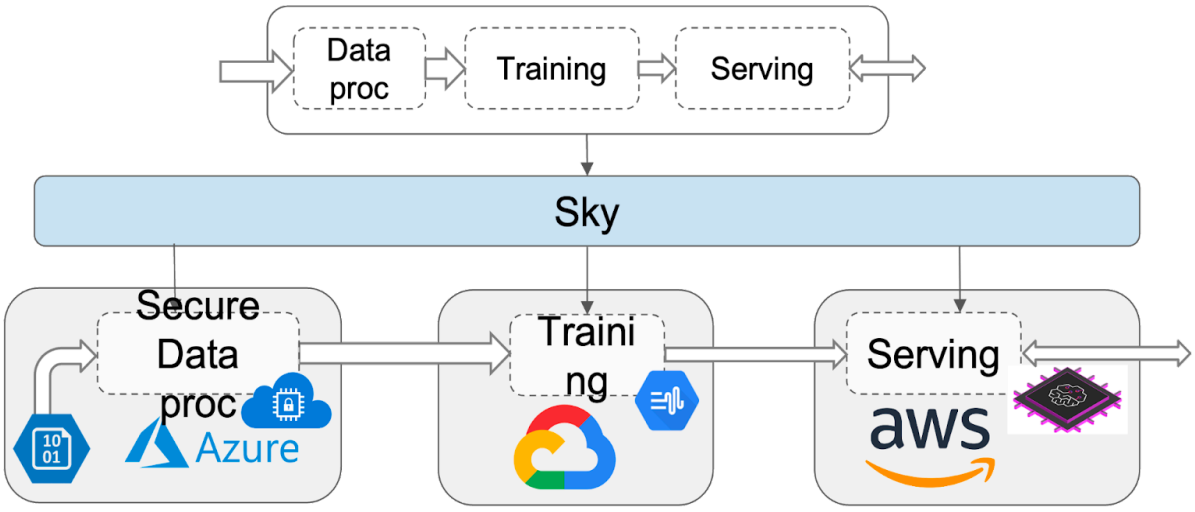

Figure 7.3: Machine learning pipeline running on top of Sky

Source: [59]

NLP Pipeline (BERT Fine-Tuning with Confidential Data) comprised three tasks: secure data preprocessing using Intel SGX (confined to Azure), model training (on either GCP TPU or Azure V100), and inference (on AWS Inferentia or T4). SkyPilot's optimizer placed each task on the optimal cloud provider: secure preprocessing on Azure, training on GCP (TPU), and inference on AWS. Compared to running all tasks on Azure (the only possible single-cloud option), this brokered plan achieved an 80% cost reduction and 62% faster execution. This experiment demonstrated SkyPilot's ability to compose workflows across providers while respecting hard constraints like hardware availability.

Sky Pilot shows how an intercloud broker can effectively virtualize a fragmented multi-cloud ecosystem into a cohesive, utility-like computation platform. By dynamically navigating the heterogeneity in hardware, services, and pricing across cloud providers, Sky Pilot offers automated, policy-driven job placement that aligns with user preferences. It reduces operational overhead by abstracting away cloud-specific complexities, enabling users to focus on high-level computational intent rather than low-level infrastructure management.

Through empirical validation, Sky Pilot has proven to be a robust and flexible platform for realizing the vision of Sky Computing. Its integration of cost-aware scheduling, multi-cloud support, and real-time resource reallocation positions it as a foundational step toward achieving true utility-grade computing infrastructure.

7.4 Serverless Sky Computing

Cordingly and Lloyd et al. [60] extends the Sky Computing paradigm to serverless Function-as-a-Service (FaaS) platforms. While serverless services like AWS Lambda and Google Cloud Functions offer elasticity and cost benefits, they suffer from vendor lock-in, limited portability, and region-specific constraints.

To address these issues, the authors propose a Sky layer a platform-neutral abstraction that enables the deployment, routing, and monitoring of serverless functions across multiple clouds and regions. This architecture supports application aggregation based on service-level objectives such as low latency, reduced cost, and low carbon footprint.

Their research is structured into three thrusts:

- **Thrust 1:** Evaluates serverless aggregation across 19 AWS regions, achieving a 65% reduction in latency and up to 99.8% reduction in carbon intensity.
- **Thrust 2:** Designs the Sky-layer and analyses trade-offs in multi-region and multi-cloud deployments.
- **Thrust 3:** Enables autonomous resource aggregation using ML models like CPU-TAMS, achieving 58% cost savings through optimized memory configurations.

They also developed tools such as the Function-as-a-Service Experiment Toolkit (FaaSET) and the Serverless Application Analytics Framework (SAAF) to support platform-neutral testing and monitoring. These results demonstrate that serverless Sky Computing can unify multi-cloud FaaS deployments, offering portability, performance improvements, cost-efficiency, and sustainability gains.

Complementing this direction, the BAASLESS platform by Larcher et al. [61] advances the Sky Computing vision by enabling federated serverless workflows that integrate Backend-as-a-Service (BaaS) components. BAASLESS introduces a globally distributed platform that federates FaaS, BaaS, and storage across AWS and GCP, using a unified SDK that abstracts vendor-specific interfaces. By dynamically scheduling each function, BaaS call, and storage operation based on real-time performance benchmarks, BAASLESS eliminates vendor lock-in and optimizes workflow make span. Its scheduler operates similarly to an Intercloud Broker, selecting

deployment plans that achieve up to 2.95× performance improvements over state-of-the-art serverless schedulers while maintaining or reducing cost. BAASLESS supports complex serverless applications such as AI pipelines and document processing, making it a concrete realization of serverless Sky Computing for BaaS-enabled workloads.

Together, these developments reflect a growing ecosystem of tools and systems that bring the Sky Computing paradigm to serverless environments, unlocking the potential for cloud-native applications to span providers with policy-driven intelligence and minimal operational complexity.

7.5 Tools, Technologies, and Limitations in Sky Computing

Recent literature emphasizes that Sky Computing, envisioned as a globally federated cloud ecosystem, relies on a combination of open-source software, compatibility layers, intercloud brokers, to enable seamless multi-cloud interoperability.

7.5.1 Tools and Technologies

Table 7.1: Sky computing Tools/Technologies

Category	Tools / Frameworks
Open-Source Frameworks	Kubernetes, Docker, Apache Spark, Ray, TensorFlow, Kafka
Intercloud Brokers & APIs	SkyPilot, BAASLESS
Storage Compatibility	S3FS, GCS FUSE, Delta Sharing
Serverless Enablement	FaaSET, SAAF, BAASLESS SDK

7.5.2 Limitations and Challenges

Sky Computing still faces several open challenges:

- **Economic and Strategic Barriers:** Proprietary APIs, vendor lock-in, egress pricing, and lack of incentives among dominant cloud providers hinder standardization and cross-cloud adoption
- **Technical Gaps:** No universal intercloud routing, naming, or billing systems exist. Debugging, data consistency, and optimization across heterogeneous platforms remain unsolved
- **Performance and Portability:** Cross-provider deployment introduces performance unpredictability, increased development overhead, and lack of support for real-time workloads

7.5.3 Contribution of this Thesis to Sky computing

Sky Computing is a new approach that aims to make it easier for organizations to use multiple cloud providers such as AWS, Azure, and GCP in a seamless and flexible way. It allows applications to move freely across cloud platforms based on factors like cost, performance, and availability. For this to work, users must be able to compare real-time data across different cloud

platforms, including pricing, performance, and hidden costs. This thesis contributes directly to that need.

Earlier chapters of this work cover several important areas that support the goals of Sky Computing. Chapter 4 analyzed the different pricing models used by AWS, Azure, and GCP. Chapter 5 compared prices across regions and evaluated additional hidden costs like data ingress and egress charges, which can significantly affect the total cost of using cloud services. Chapter 6 looked into cost control APIs offered by each provider, which allow users to monitor, track, and manage costs programmatically.

The analysis of data transfer (ingress and egress) costs is especially relevant for Sky Computing. In a multi-cloud setup, data often moves between cloud providers or between cloud and on-premise systems. This movement can lead to high egress charges, which vary by provider and region. For example, sending data out of a cloud region to the internet or another cloud can cost significantly more than expected. Sky Computing systems must take these costs into account when deciding where to place workloads. The findings in this thesis help highlight these cost differences and show why they are important for effective cross-cloud workload placement.

In Chapter 8, this thesis presents a working prototype a web-based dashboard that compares the cost of virtual machines instances across AWS, Azure, and GCP. The dashboard uses real-time API data to help users choose the most cost-effective option for their compute needs. Although this tool is not a complete Sky Computing platform, it works like a basic version of the cost-aware decision-making component found in Sky Computing architectures. It helps users avoid vendor lock-in and make smarter, more flexible choices based on cost and region.

In summary, this thesis supports Sky Computing by providing a deeper understanding of cloud pricing, exposing hidden costs like data transfer fees, and building a working tool for real-time cost comparison. These contributions help pave the way toward the Sky Computing vision where cloud resources are treated like a utility, and users can move workloads across clouds easily and efficiently.

8 Implementation

To enhance the usability and accessibility of the cost analysis prototype, a web-based graphical user interface (GUI) is developed. Instead of navigating through individual pricing calculators or APIs, users can access equivalent virtual machine (VM) configurations and pricing models within a single unified interface. This interface serves as the central control point for users to interact with cloud pricing data across AWS, Azure, and GCP.

The prototype's frontend is developed using the open-source Panel library, part of the HoloViz ecosystem in Python. Panel is a useful open-source Python library that makes it easier to build data-driven tools, dashboards, and complex web applications within the Python environment [62].

8.1 GUI Functionalities

The GUI enables users to:

- Select cloud providers and regions interactively.
- Define specific resource configurations such as vCPU and RAM.
- Automatically retrieve matching instance types from the selected configuration.
- Choose pricing models, such as On-Demand, Reserved (RI/CUD), or Spot/Preemptible.
- Generate dynamic visual comparisons of monthly cost estimates.
- Visualize cost estimates using Bokeh-based bar charts, with monthly breakdowns.
- Review cost summaries, including hourly, monthly, and effective pricing.

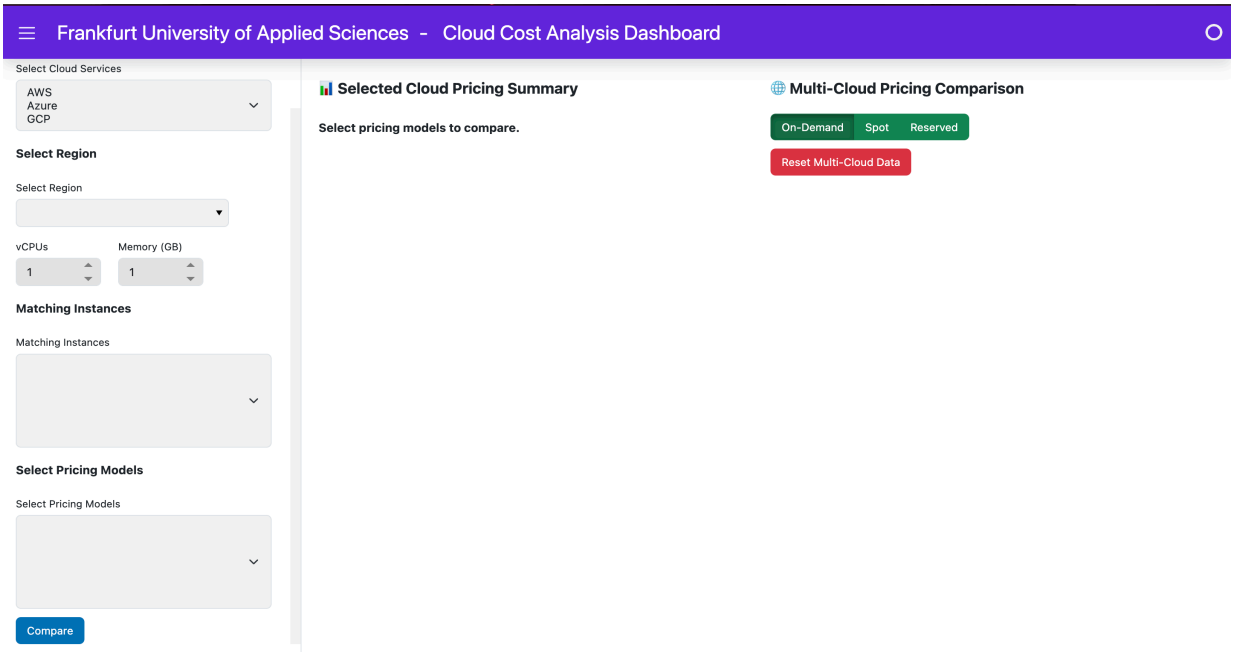

Figure 8.1: Initial interface of the Cloud Cost Analysis Dashboard

8.1.1 Cloud Provider and Region Selection

To keep pricing analysis accurate and up to date, the prototype includes region selection for AWS, Azure, and GCP. When a user selects a cloud provider in the interface, the system uses that provider's native SDKs and APIs to fetch the list of available compute regions. This process runs in the background, so users always see the latest region options without needing to update anything manually.

A. **Region Extraction**

Upon selecting a cloud provider, the application invokes the following API logic for region discovery:

- **AWS**
  - API Used**:** boto3.client("ec2").describe_regions(AllRegions=True)
  - Return Value: A sorted list of region codes (e.g., us-east-1, eu-west-1).
- **Azure**
  - API Used: ComputeManagementClient.resource_skus.list() from azure.mgmt.compute
  - Logic: Filters SKU objects with resource type virtualMachines, aggregates distinct locations, normalizes to lowercase.
  - Return Value: A dictionary mapping API identifiers (e.g., centralus) to human-readable names (e.g., Central US).
- **GCP**
  - API Used: compute.regions().list() from the googleapiclient.discovery module
  - Logic: Authenticates using a service account and retrieves a list of region names from the project-level metadata.
  - Return Value: A list of region names such as us-west1, asia-east2.

Table 8.1: Cloud Provider Region API Comparison

Cloud Provider	API Used	Data Size	Time Taken
AWS	boto3.ec2.describe_regions()	3.62 KB	0.96 sec
Azure	ComputeManagementClient.resource_skus.list()	84.33 MB	36.47 sec
GCP	compute.regions().list()	316.05 KB	0.89 sec

The execution time for retrieving region list data varies based on both the data size and the network bandwidth available during the request. In this test , the system used a fast connection with an average download speed of 268.42 Mbps and an upload speed of 159.70 Mbps**.** Despite the high-speed internet, the Azure API took significantly longer (36.47 seconds) due to the large payload size of approximately 84.33 MB, in contrast to the much smaller responses from AWS (3.62 KB) and GCP (316.05 KB), which completed in under a second. This highlights how both payload size and API design can influence perceived performance, even on fast networks.

One important implementation detail specific to AWS is the mismatch between how regions are named in the API and how they appear in the AWS Pricing Calculator or pricing functions. When regions are retrieved using describe_regions(), they are returned in code format e.g., "us-east-1".

However, in the AWS Pricing Calculator, the same region is displayed using a human-readable name such as "US East (N. Virginia)", and this is also the format expected by pricing. To bridge this gap, a static mapping is maintained between AWS region codes and their corresponding display names

B. GUI Integration and User Flow

When a cloud provider is selected on the GUI, the corresponding region dropdown is dynamically populated with available regions. The figure 8.2 illustrates this functionality for AWS, Azure, and GCP. Region data is retrieved using native APIs provided by each cloud platform to ensure current and accurate listings. These dynamic dropdowns ensure that users interact only with relevant and available regions.

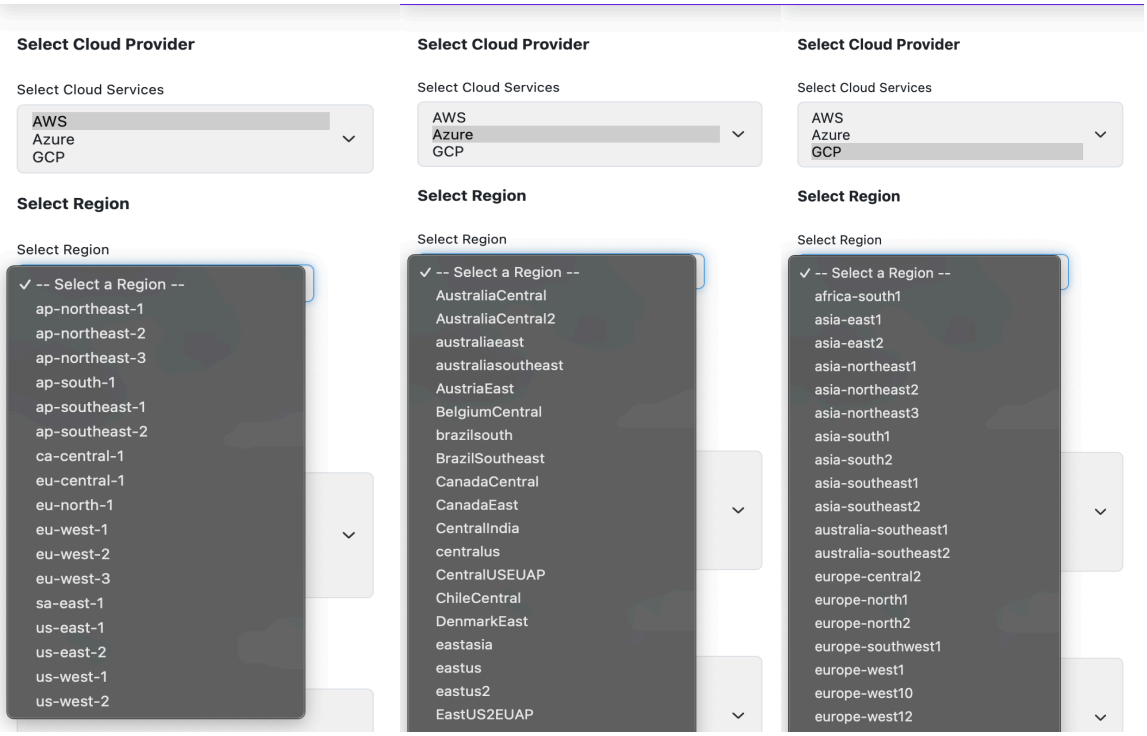

Figure 8.2: Region Selection Across AWS, Azure, and GCP

8.1.2 Resource Configuration Input

The prototype includes dynamic instance type filtering based on user-defined CPU and memory and region. After a region is selected, users specify the desired configuration using input fields for vCPUs and Memory (GB)**.** The system then queries the cloud provider-specific instance catalogs to retrieve matching compute instance types.

Once the user specifies the desired region and configuration (e.g., 8 vCPUs and 32 GB RAM), the system invokes provider-specific API functions to retrieve matching instance types:

- **AWS:** Filters instance types returned by *describe_instance_types* for a given region.
- **Azure:** Matches SKUs in *resource_skus.list()* by parsing memory and core attributes.
- **GCP:** Lists available machine types in a zone using *machineTypes().list(),* matching based on exact core and RAM.

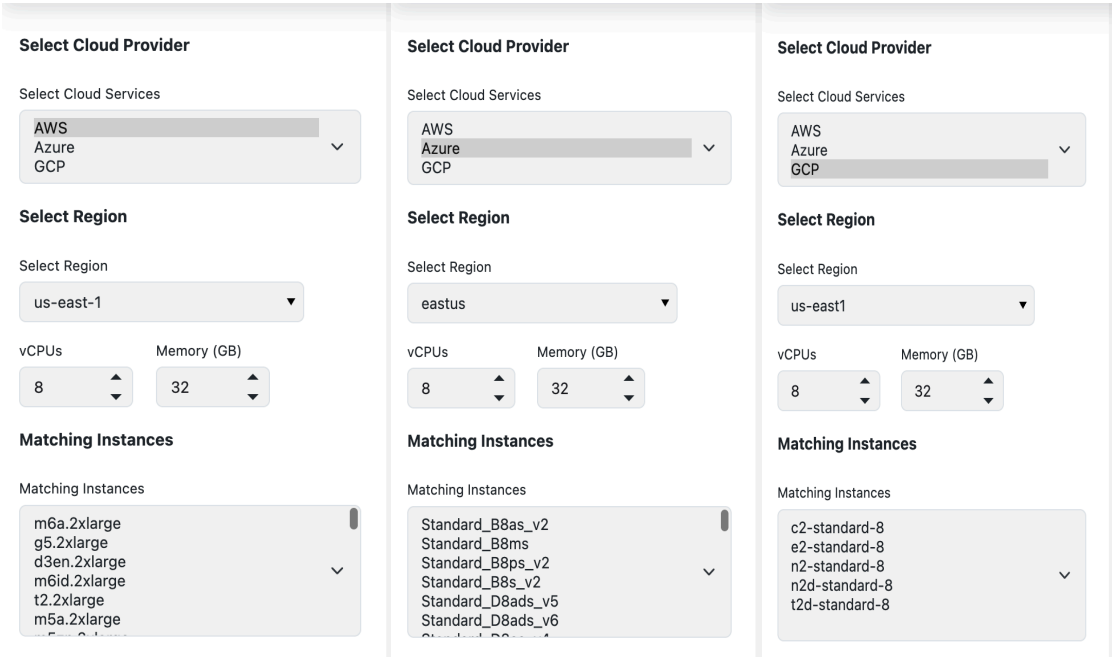

Figure 8.3: Instance Type Matching Based on User-Defined Resource Configuration

These dropdowns are populated in real-time and reflect only those instance types that meet the exact vCPU, memory specification and region, improving the precision of downstream cost analysis. The table below summarizes the performance of instance matching queries for each cloud provider:

Table 8.2: Instance Type Query Time and Data Size by Provider

Cloud Provider	Matching Source	Data Size Parsed	Time Taken
AWS	describe_instance_types	1.71 MB	~9.97 sec
Azure	resource_skus.list() (pre-cached)	84.33 MB (full SKUs)	~0.18 sec
GCP	machineTypes().list() (per zone)	61.89 KB	~0.80 sec

Azure's matching is extremely fast due to SKU pre-caching and optimized filtering logic. AWS requires complete pagination through instance metadata. GCP's performance is dependent on zone selection and usually returns results within a second.

8.1.3 Pricing Model Selection

Once a user selects a desired instance types from AWS, Azure, and GCP, the system fetches the pricing models and their prices associated with the instance and the region

A. AWS Pricing Retrieval

For AWS, the pricing data is retrieved using the Boto3 SDK and AWS Pricing API. On-Demand pricing is queried directly using the *get_products()* method for the specified instance type, region, and operating system (Linux). Reserved Instances (RI), both Standard and Convertible, are retrieved with 1-year and 3-year commitments, and cover all payment options including All Upfront, Partial Upfront, and No Upfront. Spot pricing is obtained from the EC2 *describe_spot_price_history()* API, with an average calculated over the previous seven days to ensure pricing stability. The final output includes hourly and monthly cost estimates (normalized to 730 hours per month), effective hourly rates for prepaid RIs, and a detailed breakdown for each pricing model and payment method.

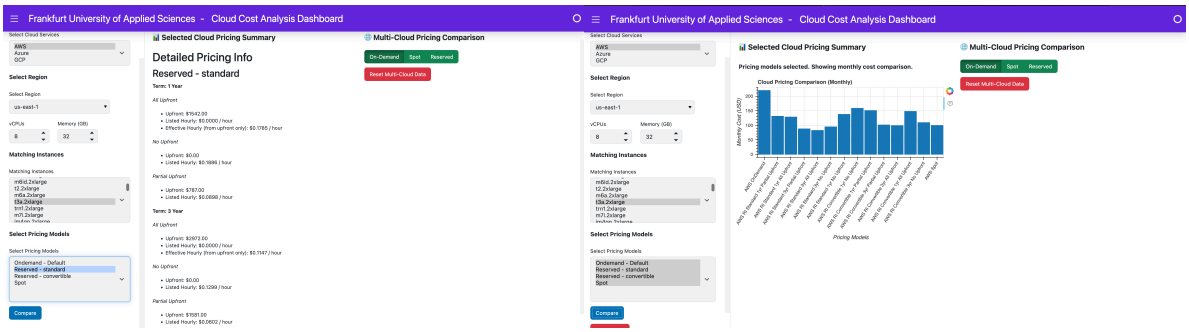

Figure 8.4: AWS Pricing Model Selection and Cost Breakdown

To validate the accuracy of API-based pricing retrieval, the same EC2 instance configurations are also priced using the official AWS Pricing Calculator GUI. For example, the 1-year Standard Reserved Instance with all upfront payment was priced at $1,542.00, and the 3-year term at $2,972.00 both via the pricing calculator and through the pricing dashboard developed using API data. This cross-verification, shown in the images, confirms the consistency and accuracy of the pricing data retrieved programmatically compared to AWS's own pricing calculator .

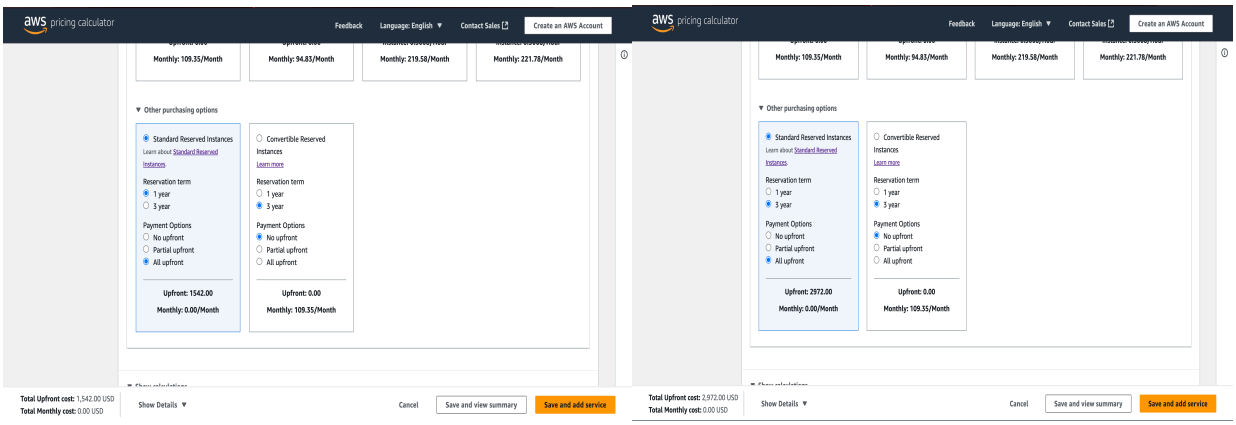

Figure 8.5: Pricing from the AWS pricing calculator

B. Azure Pricing Retrieval

Azure pricing is retrieved through the Azure Retail Prices API using REST-based queries with filters for the region (armRegionName) and SKU name (armSkuName). The implementation handles paginated responses to ensure complete SKU coverage for each VM family. Only Linux-compatible instances are considered, and pricing models supported include Pay-As-You-Go (On-Demand) and Reserved Instances for 1-year and 3-year terms, again covering All Upfront, Partial Upfront, and No Upfront options. Retrieved data is normalized into hourly and monthly rates, and each entry is annotated with pricing model, payment type, reservation term, and SKU metadata.

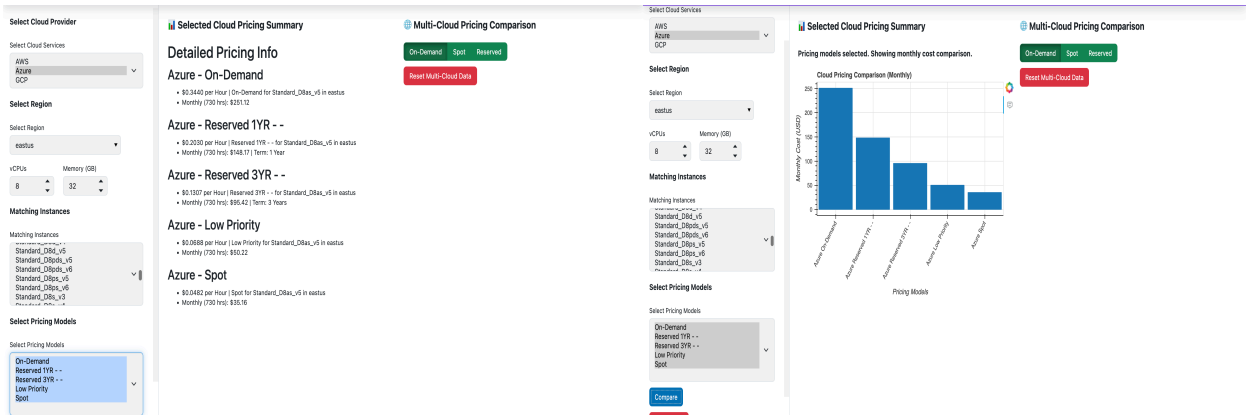

Figure 8.6: Azure Pricing Model Selection and Cost Comparison

To ensure the correctness of pricing data fetched via the API, the results are cross-validated using the Azure Pricing Calculator GUI. For instance, the 1-year Reserved Instance with no upfront payment for the D8as v5 VM shows a monthly cost of $148.17, while the 3-year Reserved Instance under the same conditions costs $95.42 per month. These values are consistent with the results retrieved via the Azure Retail Prices API and displayed in our cost dashboard. This alignment between GUI-based and API-based outputs confirms the reliability of our automated pricing retrieval process.

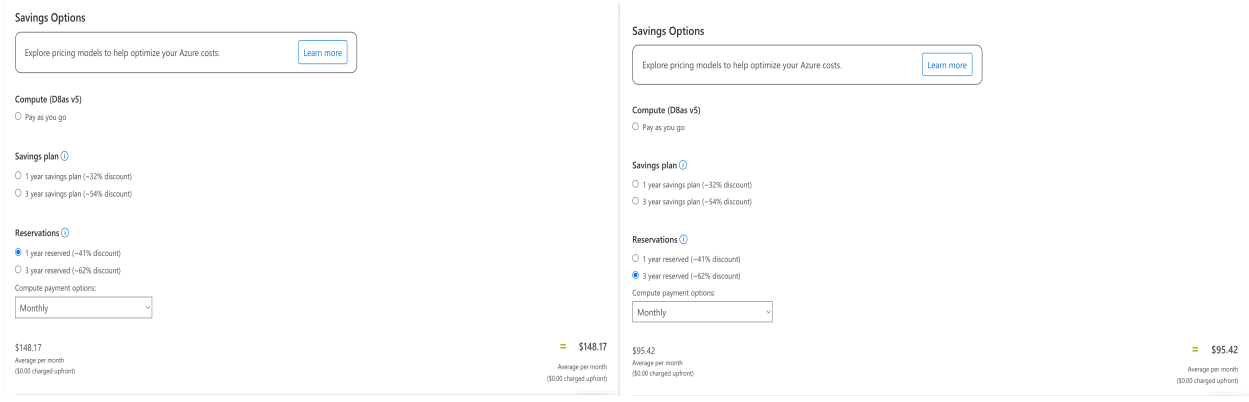

Figure 8.7: Pricing from the Azure pricing calculator

C. GCP Pricing Retrieval

GCP pricing is accessed via the Cloud Billing Catalog API, querying SKUs under the Compute Engine service (service ID: 6F81-5844-456A). The system filters out custom, sole tenancy, and non-matching instance families, focusing on instances based on user-selected vCPU and memory values. Supported models include On-Demand, Preemptible (Spot), and Committed Use Discounts (CUD) with 1-year and 3-year terms. The cost is calculated using a linear pricing formula:

$$Hourly\ Rate\ =\ (vCPU\ Count\ \times\ vCPU\ Rate)\ +\ (RAM\ (GB)\ \times\ RAM\ Rate)$$

$$Monthly\ Cost\ =\ Hourly\ Rate\ \times\ 730$$

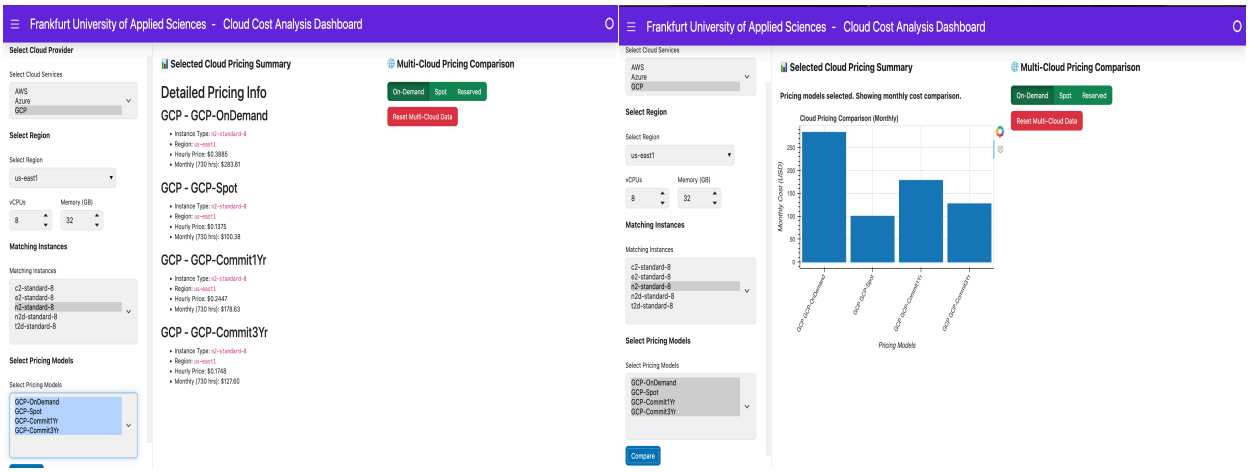

Figure 8.8: GCP Pricing Model Selection and Cost Comparison

To confirm the reliability of the pricing data retrieved from the API, equivalent configurations are compared using the official Google Cloud Pricing Calculator. For example, the 1-year committed use price is shown as $179.65/month in the GCP calculator, while our API-based dashboard reports a closely matching value of $180.71/month. Similarly, the 3-year committed use pricing is $128.62/month in the calculator and $129.71/month from the API. The difference of approximately $1 (less than 1%) is likely due to rounding differences or slight fluctuations in the base pricing components (vCPU/RAM granularity).

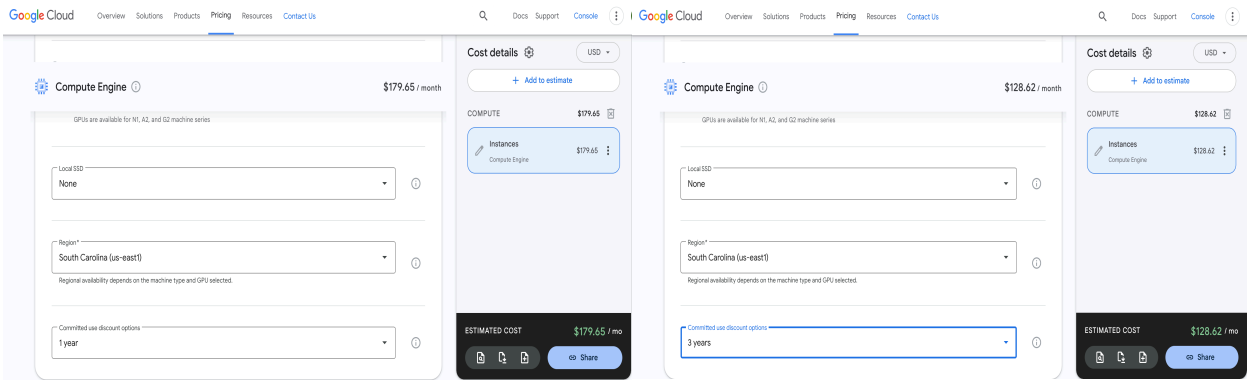

Figure 8.9: Pricing from the GCP pricing calculator

All retrieved pricing data is then visualized through an interactive bar chart using the Bokeh library, enabling users to compare monthly costs across selected pricing models. Each model is also accompanied by a detailed breakdown displaying hourly rates, upfront payments (if any), and configuration metadata such as region and instance type.

D. Output Presentation and Comparison

After pricing model selection, the dashboard:

- Plots a comparative bar chart of monthly costs using Bokeh.
- Displays detailed summaries for each selected VM:
  - Term type (On-Demand, Spot, Reserved)
  - Region, instance type
  - Hourly, Monthly, and Effective pricing
- Supports grouped multi-cloud comparison by pricing type (via RadioButtonGroup)
- Provides options to clear/reset selections for a fresh analysis.

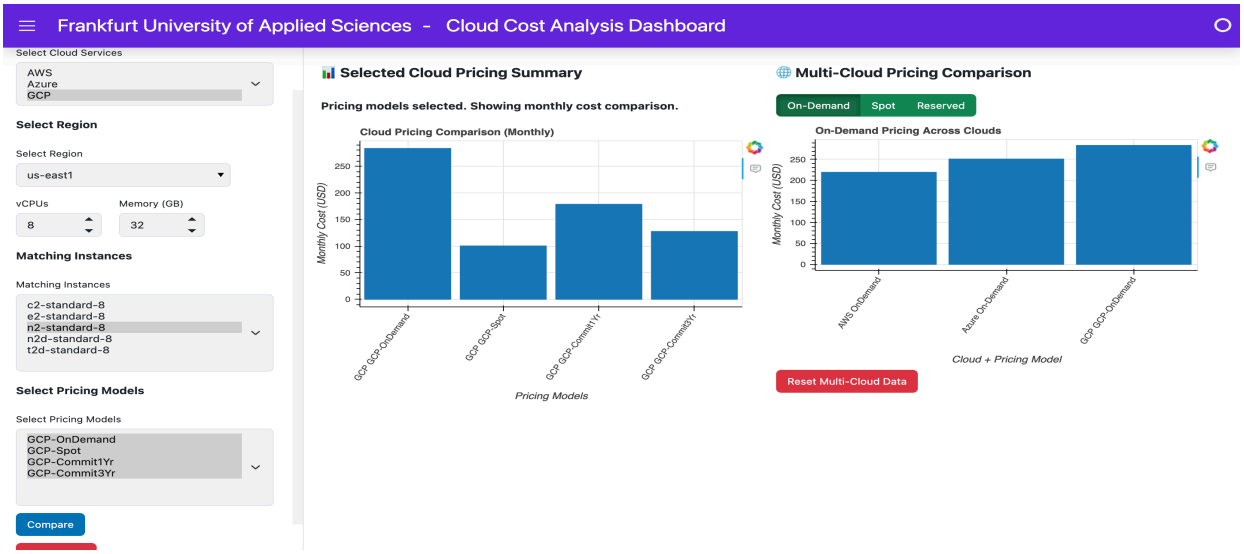

Figure 8.10: Multi-Cloud Cost Comparison

Users can toggle between On-Demand, Reserved, and Spot pricing categories using an integrated radio button group and may reset the dashboard to perform new comparisons. This dynamic and extensible architecture allows for seamless evaluation of cloud compute pricing.

8.1.4 Summary

To simplify and enhance multi-cloud pricing analysis, a dynamic and user-friendly dashboard is developed using the open-source Panel library in Python. This web-based GUI enables users to interactively select cloud providers (AWS, Azure, GCP), regions, and compute configurations (vCPUs and RAM), and retrieve accurate instance types and pricing models in real time. The interface centralizes cost comparison, reducing the need to navigate multiple pricing tools or APIs. It supports region discovery, resource filtering, and visual cost analysis across different pricing models, including On-Demand, Reserved, Spot.

The system uses official APIs from each provider to ensure reliable and up-to-date pricing data. It retrieves region lists, filters instances based on hardware specs, and calculates monthly costs using pricing metadata for each cloud. The dashboard visualizes pricing results using Bokeh bar charts and presents detailed cost summaries, enabling users to make informed, region aware decisions. This analysis is crucial, as cloud pricing varies significantly across providers, regions, and models making it essential to evaluate these factors thoroughly before selecting a cloud provider for deployment.

This prototype is particularly useful for organizations and developers planning cloud infrastructure, as it allows them to assess costs beforehand and choose the right compute service aligned with budgets. Knowing pricing in advance helps avoid unexpected expenses, supports better capacity planning, and ensures strategic decision-making when choosing between On-Demand, Reserved, or Spot instances.

Additionally, this cost dashboard aligns well with the vision of Sky Computing, which promotes seamless interoperability and resource orchestration across multiple cloud platforms. By exposing standardized and dynamic pricing insights via APIs, the prototype plays a key role in enabling cost-aware orchestration decisions. This is essential for automated agents or systems in a Sky Computing context, where dynamic cost and availability are central to workload migration, and cross-cloud deployment strategies.

The implementation code and installation instructions for the dashboard prototype are publicly available on GitHub

9 Conclusion

This thesis examined cloud pricing models and cost-optimizing strategies across AWS, Azure, and GCP. It also included a prototype for comparing costs across multiple clouds and explored how these efforts support the Sky Computing approach. As more organizations move to hybrid and multi-cloud setups, managing cloud costs has become increasingly important.

To analyze cloud pricing models, this study used twofold approach: a Systematic Literature Review (SLR) and practical analysis using real-time pricing data from major cloud providers AWS, Microsoft Azure, and Google Cloud Platform (GCP). The literature review identified key pricing models commonly used in the cloud industry, including pay-as-you-go (On-Demand), Reserved Instances, Spot Instances, Hybrid Pricing. While these models offer flexibility for different use cases, the review revealed notable research gaps, particularly the lack of regional pricing comparisons and limited validation using actual provider calculators.

To bridge these gaps, the study carried out a real-time, region-specific pricing comparison of Virtual Compute Instances (VCIs) across AWS, Azure, and GCP, focusing on general-purpose instance types in Frankfurt (Germany) and Virginia (USA). The findings showed that regional price variations are significant, with Azure Savings Plans in Frankfurt being up to 56% more expensive than in Virginia. Spot pricing across all providers was found to offer substantial savings but also showed high volatility depending on the region, making it less predictable. AWS provides a wide range of pricing models, offering flexibility but also introducing complexity in cost planning. Azure follows a structured, enterprise-aligned pricing approach, though it suffers from high regional pricing differences. In contrast, GCP provides more stable and transparent pricing, especially through its Committed use Discounts.

These findings highlight the importance of understanding that cloud service costs can vary significantly depending on the region and provider. Each cloud provider uses its own pricing structures, models, and billing mechanisms, which can directly impact the total cost of ownership. Therefore, before selecting any cloud service, it is essential to evaluate what each provider offers in terms of pricing options, regional cost differences, and available discounts. Doing so helps in making informed, cost-effective decisions tailored to workload needs and geographic deployment strategies.

In addition to compute pricing, the study also included a detailed analysis of data ingress and egress costs, which are often overlooked in pricing evaluations but are highly relevant in hybrid or multi-cloud environments. Understanding data ingress and egress costs is crucial because they can significantly impact overall cloud expenses especially in hybrid or multi-cloud setups where large volumes of data move across regions or platforms. Ignoring these charges may lead to unexpected cost overruns and poor infrastructure planning. While inbound data (ingress) is generally free across all providers, outbound data (egress) pricing varies considerably by region and destination. For example, AWS charges up to $92 per TB for internet egress from Frankfurt. Azure offers 100 GB of free internet egress per region each month, but beyond that, the cost can rise to $78.30 per TB. GCP, using a tiered network model, charges $112.64 per TB on the Premium Tier, while the Standard Tier provides a more cost-effective option at $66.56 per TB.

To further support cost comparison, a Total Cost of Ownership (TCO) compared between cloud-based compute services and a similarly configured on-premises server. The results highlighted that maintaining an on-prem setup involves high upfront capital investment and ongoing operational expenses for power, cooling, and space. Over a three-year period, AWS emerged as the most cost-effective cloud provider, followed by Azure and GCP. All three cloud options proved to be more economical than operating an equivalent on-premises infrastructure, while also offering better scalability, reduced maintenance, and operational flexibility.

These insights emphasize the importance of geographic pricing and the pricing models when it comes to making cloud adoption and cost optimization decisions. Organizations looking to maximize value from cloud services should carefully consider regional pricing and the pricing models as part of their overall strategy.

Following these insights, the study implemented a two-phase instance selection analysis to validate one of the best-practice strategies identified for cost optimization: intelligent instance selection based on workload predictability. Phase 1 used only On-Demand instances, while Phase 2 adopted a mixed pricing strategy combining Reserved, On-Demand, and Spot Instances. The results revealed 30–45% cost savings across all three providers, confirming that mixed pricing strategies yield significantly better cost efficiency than relying on a single model. Aligning instance types with workload behaviour enables flexible, performance-optimized resource usage.

To support automation and cost control in cloud environments**,** the research also reviewed the cloud native, cost control APIs available from each provider. While the features and focus areas of these APIs may vary across providers, they all serve the common purpose of enabling organizations to monitor usage, enforce budgets, and optimize cloud spending. The information presented is based on official white papers and documentation provided by the respective cloud vendors. Overall, these APIs are essential tools for implementing effective and automated cost management strategies in cloud environments.

To make cost comparison more accessible, a web-based dashboard prototype was developed using Python's open-source Panel library. The dashboard connects to native APIs and dynamically fetches real-time data on virtual machine pricing across AWS, Azure, and GCP. It allows users to compare equivalent instance types and pricing models, including On-Demand, Reserved, and Spot options. Overall, the prototype successfully bridges technical cloud pricing data with accessible, actionable insights, making it a valuable tool for academic analysis, enterprise cost planning, and FinOps strategies in hybrid and multi-cloud environments.

This thesis supports the idea of Sky Computing by showing how cloud prices vary, explaining why data transfer costs matter, and creating a tool to help make smarter cost-based decisions. Together, these efforts take a practical step toward building a unified system where workloads can easily move between cloud providers based on cost and performance.

10 Future work

This study lays the groundwork for several meaningful extensions. Future research could broaden the analysis by incorporating a wider range of Virtual Compute Instances (VCIs), including memory-optimized, compute-optimized, and GPU-based instances. It could also look at other cloud services such as storage, databases, and networking. Adding Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) options would give a more complete view of how cloud pricing works across different service types. Looking at more global regions could provide a better understanding of how pricing and cost efficiency vary across locations.

For the on-premises vs. cloud comparison, this study focused on a single server setup. A more complete analysis could include other key components like storage, networking, security systems, software licenses, and staffing needs (e.g., IT administrators and support teams). Comparing full project-level infrastructure including setup, scaling, and maintenance would give a clearer and more realistic view of the long-term cost trade-offs between cloud and on-premises environments.

While this research illustrates cost-saving potential through simulated pricing scenarios using the pricing calculators, a natural progression would involve implementing the proposed strategies in real-world cloud environments. Deploying the machine learning web application architecture across AWS, Azure, and GCP. Real deployments would utilize services such as Auto Scaling Groups, Load Balancers, and Spot Instances, in conjunction with native cost tracking tools like AWS Cost Explorer, Azure Cost Management, and GCP Billing Reports. Collecting real-time data on performance, availability, and cost would enable a comprehensive assessment of trade-offs between cost-efficiency and system reliability.

The current prototype is primarily focused on compute services, specifically comparing virtual machine instances across AWS, Azure, and GCP. While this provides valuable insights, it is a limitation—since other critical cloud services like storage, databases, and networking are not yet included. These services also contribute significantly to overall cloud costs, and their inclusion would make the tool more complete and practical for real-world decision-making.

To improve the prototype, the interface can be designed to accept specific user inputs such as number of CPUs, memory, preferred region, and storage needs. Based on this input, the system should automatically identify and compare the most relevant instance types across all three cloud providers. It should then suggest the most cost-effective option and display a side-by-side comparison of the same configuration on the other providers. Expanding this feature to include services like object storage or managed databases would allow the tool to recommend not just the best compute solution, but the overall lowest-cost cloud setup based on user-defined workloads. This would turn the prototype into an intelligent advisor for cloud cost planning and bring it closer to the vision of Sky Computing where workloads move across providers in real time based on cost, performance, and user preferences.

11 References

[1] G. Rastogi and R. Sushil, “Cloud computing implementation: Key issues and solutions,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2015, pp. 320–324. Accessed: June 28, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7100266/

[2] V. K. Saxena and S. Pushkar, “Cloud computing challenges and implementations,” in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Mar. 2016, pp. 2583–2588. doi: 10.1109/ICEEOT.2016.7755159.

[3] M. Armbrust *et al.*, “Above the Clouds: A Berkeley View of Cloud Computing”.

[4] “IaaS vs PaaS vs SaaS: A Complete Overview | Virtasant.” Accessed: July 16, 2025. [Online]. Available: https://www.virtasant.com/blog/iaas-vs-paas-vs-saas-a-complete-overview

[5] P. J. Nesse *et al.*, “Exploiting cloud computing — A proposed methodology for generating new business,” in *2011 15th International Conference on Intelligence in Next Generation Networks*, Oct. 2011, pp. 241–246. doi: 10.1109/ICIN.2011.6081083.

[6] G. Callou *et al.*, “Estimating sustainability impact, total cost of ownership and dependability metrics on data center infrastructures,” in *Proceedings of the 2011 IEEE International Symposium on Sustainable Systems and Technology*, May 2011, pp. 1–6. doi: 10.1109/ISSST.2011.5936859.

[7] “Choosing the Right Infrastructure: Cloud, Hybrid, or On-Premise? - Softjourn,” Softjourn Inc. Accessed: July 11, 2025. [Online]. Available: https://softjourn.com/insights/cloud-vs-on-premise

[8] K. K. Azumah, L. T. Sørensen, and R. Tadayoni, “Hybrid Cloud Service Selection Strategies: A Qualitative Meta-Analysis,” in *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, Aug. 2018, pp. 1–8. doi: 10.1109/ICASTECH.2018.8506887.

[9] V. A. K. Gorantla, V. Gude, S. K. Sriramulugari, N. Yuvaraj, and P. Yadav, “Utilizing Hybrid Cloud Strategies to Enhance Data Storage and Security in E-Commerce Applications,” in *2024 2nd International Conference on Disruptive Technologies (ICDT)*, Mar. 2024, pp. 494–499. doi: 10.1109/ICDT61202.2024.10489749.

[10] B. E. Babu, G. S. V. Prasad, P. S. Sarma, and S. Neelima, “Hybrid Cloud Strategies: Balancing the Benefits of Public and Private Clouds for Optimal Performance and Security,” in *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, Dec. 2024, pp. 1–6. doi: 10.1109/ICMNWC63764.2024.10872160.

[11] F. Huang, H. Li, Z. Yuan, and X. Li, “An Application Deployment Approach Based on Hybrid Cloud,” in *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*, May 2017, pp. 74–79. doi: 10.1109/BigDataSecurity.2017.54.

[12] B. Zhou, F. Zhang, J. Wu, and Z. Liu, “Cost Reduction in Hybrid Clouds for Enterprise Computing,” in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2017, pp. 270–274. doi: 10.1109/ICDCSW.2017.13.

[13] "What is Hybrid Cloud Infrastructure and How Does It Work?," StarWind Blog. Accessed: July 05, 2025. [Online]. Available: https://www.starwindsoftware.com/blog/what-is-hybrid-cloud-infrastructure/
[14] R. Pellegrini, P. Rottmann, and G. Strieder, "Preventing vendor lock-ins via an interoperable multi-cloud deployment approach," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2017, pp. 382–387. doi: 10.23919/ICITST.2017.8356428.
[15] M. M. Alshammari, A. A. Alwan, A. Nordin, and I. F. Al-Shaikhli, "Disaster recovery in single-cloud and multi-cloud environments: Issues and challenges," in *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Nov. 2017, pp. 1–7. doi: 10.1109/ICETAS.2017.8277868.
[16] U. Bellur, A. Malani, and N. C. Narendra, "Cost Optimization in Multi-site Multi-cloud Environments with Multiple Pricing Schemes," in *2014 IEEE 7th International Conference on Cloud Computing*, June 2014, pp. 689–696. doi: 10.1109/CLOUD.2014.97.
[17] "Future of Cloud Computing - 8 Trends & Predictions." Accessed: July 11, 2025. [Online]. Available: https://www.cloudpanel.io/blog/future-of-cloud-computing/
[18] D. Seth, H. Nerella, M. Najana, and A. Tabbassum, "Navigating the Multi-Cloud Maze: Benefits, Challenges, and Future Trends," *Int. J. Glob. Innov. Solut. IJGIS*, June 2024, doi: 10.21428/e90189c8.8c704fe4.
[19] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, Ann Arbor Michigan: ACM, June 2021, pp. 26–32. doi: 10.1145/3458336.3465301.
[20] S. Chasins *et al.*, "The Sky Above The Clouds," May 14, 2022, *arXiv*: arXiv:2205.07147. doi: 10.48550/arXiv.2205.07147.
[21] A. Carrera-Rivera, W. Ochoa, F. Larrinaga, and G. Lasa, "How-to conduct a systematic literature review: A quick guide for computer science research," *MethodsX*, vol. 9, p. 101895, Nov. 2022, doi: 10.1016/j.mex.2022.101895.
[22] Y. Zhou, H. Zhang, X. Huang, S. Yang, M. A. Babar, and H. Tang, "Quality assessment of systematic reviews in software engineering: a tertiary study," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, Nanjing China: ACM, Apr. 2015, pp. 1–14. doi: 10.1145/2745802.2745815.
[23] S. Kansal, G. Singh, H. Kumar, and S. Kaushal, "Pricing Models in Cloud Computing," in *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*, in ICTCS '14. New York, NY, USA: Association for Computing Machinery, Oct. 2014, pp. 1–5. doi: 10.1145/2677855.2677888.
[24] M. K. M. Murthy, H. A. Sanjay, and J. P. Ashwini, "Pricing models and pricing schemes of IaaS providers: a comparison study," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, in ICACCI '12. New York, NY, USA: Association for Computing Machinery, Aug. 2012, pp. 143–147. doi: 10.1145/2345396.2345421.
[25] S. Arshad, S. Ullah, S. A. Khan, M. D. Awan, and M. S. H. Khayal, "A survey of Cloud computing variable pricing models," in *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, Apr. 2015, pp. 27–32. Accessed: Mar. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7320330
[26] I. Lee, "Developing pricing strategies for cloud service providers in a competitive market," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing*

Companion (UCC Companion), Dec. 2018, pp. 387–392. doi: 10.1109/UCC-Companion.2018.8653578.
[27] A. Choudhary, P. K. Verma, and P. Rai, "Comparative Study of Various Cloud Service Providers: A Review," in *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, Dec. 2022, pp. 1–8. doi: 10.1109/ICPECTS56089.2022.10047594.
[28] G. R. -, "Microsoft Azure vs. Google Cloud Platform: An In-Depth Review of Services, Pricing, and Performance," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 13, no. 1, p. 232021, Jan. 2025, doi: 10.37082/IJIRMPS.v13.i1.232021.
[29] K. K. Azumah, L. T. Sørensen, and R. Tadayoni, "Hybrid Cloud Service Selection Strategies: A Qualitative Meta-Analysis," in *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, Aug. 2018, pp. 1–8. doi: 10.1109/ICASTECH.2018.8506887.
[30] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Multiple Workflows Scheduling in Multi-tenant Distributed Systems: A Taxonomy and Future Directions," *ACM Comput Surv*, vol. 53, no. 1, p. 10:1-10:39, Feb. 2020, doi: 10.1145/3368036.
[31] A. Pavlenko *et al.*, "Vertically Autoscaling Monolithic Applications with CaaSPER: Scalable Container-as-a-Service Performance Enhanced Resizing Algorithm for the Cloud," in *Companion of the 2024 International Conference on Management of Data*, in SIGMOD '24. New York, NY, USA: Association for Computing Machinery, June 2024, pp. 241–254. doi: 10.1145/3626246.3653378.
[32] A. Erben, R. Mayer, and H.-A. Jacobsen, "How Can We Train Deep Learning Models Across Clouds and Continents? An Experimental Study," *Proc VLDB Endow*, vol. 17, no. 6, pp. 1214–1226, Feb. 2024, doi: 10.14778/3648160.3648165.
[33] X. Liu and R. Buyya, "Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions," *ACM Comput Surv*, vol. 53, no. 3, p. 50:1-50:41, May 2020, doi: 10.1145/3355399.
[34] A. Mampage, S. Karunasekera, and R. Buyya, "A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions," *ACM Comput Surv*, vol. 54, no. 11s, p. 222:1-222:36, Sept. 2022, doi: 10.1145/3510412.
[35] M. Liu, L. Pan, and S. Liu, "Cost Optimization for Cloud Storage from User Perspectives: Recent Advances, Taxonomy, and Survey," *ACM Comput Surv*, vol. 55, no. 13s, p. 266:1-266:37, July 2023, doi: 10.1145/3582883.
[36] Z. Wang and Z. Shao, "TimeUnion: An Efficient Architecture with Unified Data Model for Timeseries Management Systems on Hybrid Cloud Storage," in *Proceedings of the 2022 International Conference on Management of Data*, in SIGMOD '22. New York, NY, USA: Association for Computing Machinery, June 2022, pp. 1418–1432. doi: 10.1145/3514221.3526175.
[37] Master Of Science In Information Technology , Washington University Of Science And Technology, Alexandria, Virginia, USA and A. Islam, "HYBRID CLOUD DATABASES FOR BIG DATA ANALYTICS: A REVIEW OF ARCHITECTURE, PERFORMANCE, AND COST EFFICIENCY," *Int. J. Manag. Inf. Syst. Data Sci.*, vol. 1, no. 4, pp. 96–114, Sept. 2024, doi: 10.62304/ijmisds.v1i04.208.
[38] J. L. Díaz, J. García, J. Entrialgo, M. García, and D. F. García, "Joint optimization of the cost of computation and virtual machine image storage in cloud infrastructure," in

Proceedings of the 2020 Summer Simulation Conference, in SummerSim ’20. San Diego, CA, USA: Society for Computer Simulation International, Nov. 2020, pp. 1–8.
[39] Kalyan Chakravarthy Thatikonda, “Comprehensive Cloud Migration Strategies: Optimizing for Reliability, Security, and Performance,” *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 11, no. 2, pp. 72–82, Mar. 2025, doi: 10.32628/CSEIT251112388.
[40] Arpita Hajra and Lagan Goel, “Cloud Migration Strategies for Legacy ERP Systemsn Challenges & Solution,” *Int. J. Res. Publ. Semin.*, vol. 16, no. 2, pp. 169–175, Apr. 2025, doi: 10.36676/jrps.v16.i2.61.
[41] B. E. Babu, G. S. V. Prasad, P. S. Sarma, and S. Neelima, “Hybrid Cloud Strategies: Balancing the Benefits of Public and Private Clouds for Optimal Performance and Security,” in *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, Dec. 2024, pp. 1–6. doi: 10.1109/ICMNWC63764.2024.10872160.
[42] “EC2 On-Demand Instance Pricing – Amazon Web Services,” Amazon Web Services, Inc. Accessed: July 06, 2025. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/
[43] “Azure VM Pricing: What You Don’t Know (not yet).” Accessed: July 06, 2025. [Online]. Available: https://intercept.cloud/en-gb/blogs/azure-vm-pricing
[44] “VM instance pricing,” Google Cloud. Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/compute/vm-instance-pricing
[45] “AWS Pricing Calculator.” Accessed: July 06, 2025. [Online]. Available: https://calculator.aws/#/
[46] “Pricing Calculator | Microsoft Azure.” Accessed: July 06, 2025. [Online]. Available: https://azure.microsoft.com/en-us/pricing/calculator
[47] “Google Cloud Pricing Calculator.” Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/products/calculator?hl=en
[48] “Network pricing,” Google Cloud. Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/vpc/network-pricing
[49] R. L. Sawyer, “Calculating Total Power Requirements for Data Center”.
[50] K. Konstantinos, P. Mitropoulou, E. Filiopoulou, C. Michalakelis, and M. Nikolaidou, *Cloud computing and economic growth*. 2015. doi: 10.1145/2801948.2802000.
[51] “Welcome - AWS Billing and Cost Management.” Accessed: July 06, 2025. [Online]. Available: https://docs.aws.amazon.com/aws-cost-management/latest/APIReference/Welcome.html
[52] jojohpm, “Cost Management automation overview - Microsoft Cost Management.” Accessed: July 06, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/cost-management-billing/automate/automation-overview
[53] bandersmsft, “Azure Consumption REST APIs.” Accessed: July 06, 2025. [Online]. Available: https://learn.microsoft.com/en-us/rest/api/consumption/
[54] lmazuel, “azure.mgmt.consumption package.” Accessed: July 06, 2025. [Online]. Available: https://learn.microsoft.com/en-us/python/api/azure-mgmt-consumption/azure.mgmt.consumption?view=azure-python
[55] “Cloud Billing Budget API,” Google Cloud. Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/billing/docs/reference/budget/rest
[56] “Get Google Cloud pricing information | Cloud Billing.” Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/billing/docs/how-to/get-pricing-information-api

[57] "Get Started with the Cloud Billing API | Google Cloud." Accessed: July 06, 2025. [Online]. Available: https://cloud.google.com/billing/v1/getting-started
[58] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, in HotOS '21. New York, NY, USA: Association for Computing Machinery, June 2021, pp. 26–32. doi: 10.1145/3458336.3465301.
[59] Z. Yang *et al.*, "SkyPilot: An Intercloud Broker for Sky Computing".
[60] R. Cordingly and W. Lloyd, "Enabling Serverless Sky Computing," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*, Sept. 2023, pp. 232–235. doi: 10.1109/IC2E59103.2023.00038.
[61] T. Larcher, P. Gritsch, S. Nastic, and S. Ristov, "BaaSLess: Backend-as-a-Service (BaaS)-Enabled Workflows in Federated Serverless Infrastructures," *IEEE Trans. Cloud Comput.*, vol. 12, no. 4, pp. 1088–1102, Oct. 2024, doi: 10.1109/TCC.2024.3439268.
[62] "Overview — Panel v1.7.2." Accessed: July 06, 2025. [Online]. Available: https://panel.holoviz.org/

12 Appendix

Table 12.1: Quality Assessment Scores and Inclusion Results

Paper ID	Paper Title	Q 1	Q 2	Q 3	Q 4	Q 5	Q 6	Q7	Q8	Total Score	Verdict
P1	A Survey of Cloud Computing Variable Pricing Models	1	1	1	1	1	1	0.5	0.5	8	included
P2	Comparative Study of Various Cloud Service Providers: A Review	1	1	1	1	0.5	1	0.5	1	6.5	included
P3	Cloud Pricing Models: A Survey and Position Paper	1	1	1	0.5	1	1	0.5	0.5	6.5	excluded
P4	A Study of Hierarchical Cloud Resource Pricing	0.5	0.5	0.5	1	0.5	0.5	1	0.5	5	excluded
P5	Developing Pricing Strategies for Cloud Service Providers in a Competitive Market	1	1	1	1	1	1	1	1	8	included
P6	Research on Pricing Model of Cloud Storage	1	1	1	0	1	1	0.5	0.5	6	excluded
P7	A Comparative Analysis of Pricing Models for Enterprise Cloud Platforms	1	1	1	1	0.5	1	0.5	0.5	6.5	excluded
P8	Pricing Models in Cloud Computing	1	1	1	1	1	1	0.5	0.5	7	included
P9	Pricing Models and Pricing Schemes of IaaS Providers: A Comparison Study	1	1	1	1	1	1	1	1	8	included
P10	Cloud Pricing Models: Taxonomy, Survey, and	0.5	1	1	0.5	0.5	1	1	1	6.5	excluded

	Interdisciplinary Challenges										
P11	Microsoft Azure vs. Google Cloud Platform: An In-Depth Review of Services, Pricing, and Performance	1	1	1	1	1	1	1	1	8	included
P12	Comparative Analysis of Cloud Platform: Amazon Web Service, Microsoft Azure, And Google Cloud Provider	1	0.5	0.5	0.5	0.5	0.5	0	1	4.5	excluded
P13	Decoding the Cloud Giants: A Comparison of AWS, Azure and GCP	1	1	1	0	1	1	0.5	1	6.5	excluded
P14	Serverless Computing: The Future of Scalability and Efficiency with AWS, Azure, and GCP	1	1	1	0	1	1	0.5	1	6.5	excluded
P15	Navigating the World of Cloud Storage: AWS, Azure, and More	0,5	0.5	0.5	0.5	0.5	1	1	1	5.5	excluded
P17	Cost-Performance Evaluation of General Compute Instances: AWS, Azure, GCP, and OCI	1	1	1	0	1	1	0.5	1	7.5	excluded
P18	"Hybrid Cloud Service Selection Strategies: A Qualitative Meta-Analysis"	1	1	1	1	1	1	1	1	8	included
P19	Hybrid Cloud Strategies: Balancing the	1	1	0.5	1	1	1	0.5	1	7.0	included

	Benefits of Public and Private Clouds for Optimal Performance and Security										
P20	Cost Optimization for Cloud Storage from User Perspectives: Recent Advances, Taxonomy, and Survey	1	1	1	0.5	0.5	1	0.5	1	6.5	included
P21	Edge Federation: Towards an Integrated Service Provisioning Model	1	1	1	0.5	0.5	1	0.5	1	6.5	excluded
P22	Joint optimization of the cost of computation and virtual machine image storage in cloud infrastructure	1	1	1	1	1	1	0.5	1	7.5	included
P23	How Can We Train Deep Learning Models Across Clouds and Continents? An Experimental Study	1	1	1	1	1	1	0.5	1	7.5	included
P24	Multiple Workflows Scheduling in Multi-tenant Distributed Systems: A Taxonomy and Future Directions	1	1	1	1	0.5	1	0.5	0.5	6.5	included
P25	Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and	1	1	1	1	0.5	1	0.5	1	7	included

	Future Directions										
P26	A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions	1	1	1	0.5	0.5	1	1	0.5	6.5	included
P27	Vertically Autoscaling Monolithic Applications with CaaSPER:	1	1	1	1	1	1	0.5	1	7.5	included
P28	TimeUnion: An Efficient Architecture with Unified Data Model for Timeseries Management Systems on Hybrid Cloud Storage	1	1	1	1	1	1	0.5	0.5	7	included
P29	Cloud Migration Strategies for Legacy ERP Systems	1	1	1	1	1	1	0.5	0.5	7	included
P30	Comprehensive Cloud Migration Strategies: Optimizing for Reliability, Security, and Performance	1	1	1	0.5	0.5	1	0	1	7	included
P31	Hybrid Cloud Databases for Big Data Analytics: A Review of Architecture, Performance, and Cost Efficiency"	1	1	1	1	0.5	1	0	1	7.5	included
P32	Mastering cloud platform engineering with key modern concepts	1	1	1	0.5	0	0.5	0	1	5	excluded

P33	Multi-Cloud and Hybrid Infrastructure: Addressing Consistency Challenges Across Cloud Providers	1	1	1	0.5	0	0.5	0	1	5	excluded
P34	Cost optimization strategies for micro services in AWS: Managing resource consumption and scaling efficiently	1	1	1	0.5	0	0.5	0	0.5	4.5	excluded
P35	Hybrid Storage Integration: Bridging On-Premises and Cloud through APIs"	1	1	1	0.5	0	0.5	0	1	5	excluded

Table 12.2 Data Extraction form

Paper ID	Title	Authors	Year	Key Findings	Gaps Found
P1	A Survey of Cloud Computing Variable Pricing Models	Sahar Arshad, Saeed Ullah, Shoab Ahmed Khan, M. Daud Awan, M. Sikandar Hayat Khayal	2014	1. Dynamic pricing models like Spot instances help reduce costs for low-priority jobs. 2. Mixed pricing strategies (spot + on-demand) are recommended for balancing cost and reliability.	Real time pricing strategies are not deeply explored. Regional price variations were not discussed.
P2	Developing Pricing Strategies for Cloud Service Providers in a Competitive Market	In Lee	2018	1. Hybrid pricing (subscription + pay-as-you-go) maximizes profits and keeps customer costs fair. 2. Subscription discounts are key for influencing customer purchasing decisions.	1.Customer value perception beyond cost not deeply considered. 2. Regional price variations were not discussed.
P3	Pricing Models in Cloud Computing	Sahil Kansal	2014	1. Cloud providers use pay-per-use, subscription, and hybrid pricing models to meet different needs. 2. Pricing is dynamic and driven by demand, competition, and provider strategies.	1.Current models work well in stable markets but not in unpredictable or highly dynamic environments. 2.Regional price variations were not discussed.
P4	Pricing Models and Pricing Schemes of IaaS Providers: A Comparison Study	Mohan Murthy	2012	1. Providers use various pricing models (linear, step, discounts), which users can leverage to optimize costs. 2. Long-term users can save with reserved instances, while short-term users benefit from on-demand pricing.	1.No standardized pricing models make provider comparison difficult for users. 2. Regional price variations were not discussed.
P5	Microsoft Azure vs. Google Cloud Platform: An In-Depth Review of Services, Pricing, and Performance	Gaurav Rohatgi	2025	1. Azure integrates better with enterprise tools; GCP is stronger in analytics and offers automatic discounts. 2. Both use flexible pay-as-you-go and discounted pricing for predictable workloads.	1. AWS pricing is not compared, so market overview is incomplete. 2. Regional price variations were not discussed.
P6	Comparative Study of Various Cloud Service Providers: A Review	Anurag Choudhary, Pradeep Kumar Verma, Piyush Rai	2022	It provides a comparative analysis of major cloud service providers) in terms of pricing models, compute services, storage, database,	1. The paper is descriptive and real-world calculations are not present

				networking, security, and tools	2. Regional price variations were not discussed.
P7	Hybrid Cloud Service Selection Strategies: A Qualitative Meta-Analysis	Kenneth K. Azumah, Lene T. Sørensen, Reza Tadayoni	2022	Current hybrid cloud cost methods mainly focus on scheduling and auto-scaling. Many miss hidden costs like data transfers, management, and security. Handling real-time, changing workloads efficiently is still a challenge and needs more research.	1. No real-world validation of the proposed strategies. Lack of quantitative-comparison between practices.
P8	Hybrid Cloud Strategies: Balancing the Benefits of Public and Private Clouds for Optimal Performance and Security	B Eswar Babu, Ganesula Satya Vara Prasad, PKV Subbaraya Sarma, Sanugula Neelima	2024	1. Public cloud usage for non-critical tasks helps reduce costs, while private cloud ensures security for sensitive operations balancing them is essential. 2.Machine learning algorithms (Random Forest, Gradient Boosting) can effectively predict and optimize cost-performance trade-offs in hybrid cloud setups.	1.Lack of a unified model that simultaneously considers cost, security, and performance when making hybrid cloud deployment decisions. 2.Limited research on dynamic and real-time cost optimization methods for handling unpredictable and fluctuating workloads.
P9	Cost Optimization for Cloud Storage from User Perspectives: Recent Advances, Taxonomy, and Survey	Mingyu Liu, Li Pan, Shijun Liu	2023	Users can reduce storage costs by smartly using tiered storage (hot/cold/archive tiers) and choosing suitable billing methods, though balancing cost and performance remains tricky. Multi-cloud and edge storage offer savings through flexibility, though they require careful handling to avoid lock-in and slowdowns. Use of AI/ML is emerging to further automate and optimize cost decisions dynamically.	1. There is a need for more advanced algorithms and AI-based methods to automate decisions like tier selection, replica placement, and data migration to minimize costs without sacrificing performance. 2.Handling vendor lock-in and seamless data migration across clouds is still challenging, especially when balancing between redundancy, latency, and cost during multi-cloud storage operations.
P10	Joint Optimization of the Cost of Computation	José Luis Díaz	2020	1.Integrating Virtual Machine Image (VMI) storage costs into allocation strategies (via	Storage costs of VM root volumes and inter-VM data transfers are not yet

	and Virtual Machine Image Storage in Cloud Infrastructure			Malloovia+I) can achieve up to 20–30% cost savings, especially for workloads with low computational demand and large VMIs. 2.The number of VMIs deployed and the balance between computation and storage costs are critical factors for effective cloud cost optimization.	included in the optimization model.
P11	How Can We Train Deep Learning Models Across Clouds and Continents? An Experimental Study	Alexander Erben, Ruben Mayer, Hans-Arno Jacobsen	2024	1.Combining older, cheaper spot GPUs (like T4s) across clouds can be more cost-effective than using centralized expensive hardware, if model granularity allows scaling. 2.Egress and communication costs can dominate total costs in geo-distributed training, so staying within regions or selecting providers with low egress fees is critical for cost optimization. 3.Hybrid setups are only cost-efficient if the cloud is nearby.	No automated system yet chooses the cheapest cloud region while considering egress and model needs. Also, small models don't scale well across continents, more research is needed to improve that.
P12	Multiple Workflows Scheduling in Multi-tenant Distributed Systems: A Taxonomy and Future Directions	Muhammad H. Hilman, Maria A. Rodriguez, Rajkumar Buyya	2020	Combining reserved, on-demand, and spot cloud instances can significantly reduce operational costs, but requires intelligent workload-aware strategies to predict demand and optimize usage. Using dynamic and workload-aware resource provisioning (e.g., scaling resources up and down based on demand and QoS needs) is essential for minimizing costs in multi-tenant environments.	Predictive models for workload forecasting are still immature improving these can help select the right mix of reserved, spot, and on-demand resources for further cost reduction.
P13	Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy,	Xunyun Liu, Rajkumar Buyya	2020	Using smart scaling based on workload predictions helps reduce stream processing costs. Also, placing related tasks close together cuts data transfer costs in hybrid setups.	Most systems lack task-level monitoring, making cost prediction and resource planning harder. They also don't fully account for performance issues in hybrid setups, which affects smart scheduling.

	Review, and Future Directions				
P14	A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions	Anupama Mampage, Shanika Karunasekera, Rajkumar Buyya	2022	1.Hybrid serverless models that combine cloud and edge resources significantly reduce costs by optimizing placement of functions based on performance and resource availability. 2. Effective management of cold-start delays and workload profiling can substantially enhance cost efficiency in hybrid deployments.	1.There is limited research on dynamic pricing models and customized service levels in hybrid serverless environments, indicating a need for flexible cost strategies. 2 Current research doesn't fully address data locality and state management, which are key for cost-effective hybrid cloud setups.
P15	Vertically Autoscaling Monolithic Applications with CaaSPER: Scalable Container-as-a-Service Performance Enhanced Resizing Algorithm for the Cloud	Anna Pavlenko	2024	1.Reactive-Proactive Scaling Approach: Combining proactive forecasting with reactive adjustments significantly reduces cloud resource waste, optimizing costs without harming performance. 2.User-Driven Optimization: Allowing users to tailor autoscaling preferences based on price-performance trade-offs effectively balances cost savings with required workload efficiency.	The current setup focuses mostly on CPU, missing out on memory and storage savings. Also, it doesn't use app-specific data, which could lead to better optimization.
P16	TimeUnion: An Efficient Architecture with Unified Data Model for Timeseries Management Systems on Hybrid Cloud Storage	Zhiqi Wang, Zili Shao	2022	1.Efficient hybrid storage integration, using fast block storage for recent data and cheaper object storage for older data, significantly improves performance and reduces costs. 2.Dynamically adjusting data partition sizes based on real-time storage usage helps achieve optimal cost efficiency without compromising performance.	1.Lack of comprehensive strategies for automatic hot and cold data migration tailored specifically for hybrid cloud environments. 2.Insufficient exploration into optimizing query caching and indexing strategies across multi-tier cloud storage to further lower costs.
P17	Cloud Migration Strategies for	Arpita Hajra, Lagan Goel	2025	Hybrid and phased migration lowers costs and	There's no detailed cost model for hybrid setups

	Legacy ERP Systemsn Challenges & Solution			reduces risks by moving less critical tasks to the cloud. Though refactoring costs more at first, it leads to better performance and long-term savings.	after migration. Key factors like data transfer costs and cloud pricing differences are often missed. Most studies focus on migration, not ongoing cost optimization in hybrid or multi-cloud environments.
P18	Comprehensive Cloud Migration Strategies: Optimizing for Reliability, Security, and Performance	Kalyan Chakravarthy Thatikonda	2025	Hybrid and multi-cloud use cuts time-to-market by ~40% and boosts resource use by ~45%. Using Reserved and Spot Instances with auto-scaling can lower costs by 30–70%. Right-sizing and cleanup save up to 40% on compute and 25% on storage. Tagging improves cost tracking, while automation tools like IaC and CI/CD reduce overhead and improve efficiency.	The paper lacks provider-specific migration challenges, ingress/egress cost analysis, and real-world case studies. It also overlooks hybrid/multi-cloud complexities, compliance issues, and detailed automation/tool examples.
P19	Hybrid cloud databases for big data analytics: a review of architecture, performance, and cost efficiency	Ashraful Islam	2024	1.Hybrid cloud databases help cut costs by combining pay-as-you-go public cloud models with private clouds, reducing infrastructure expenses by up to 28% while still maintaining control and flexibility. 2.AI and ML integration in hybrid clouds improves workload distribution and automates resource management, leading to lower operational costs and better performance during peak demands.	1.Although hybrid systems cut costs, there's little focus on real-time cost monitoring and automated budget controls across public and private clouds. 2.Edge computing cost impacts unclear While edge improves performance, studies do not yet clearly show how it affects overall cost optimization in hybrid cloud setups. .