International Journal of Advanced Multidisciplinary Research and Educational Development
Volume 2, Issue 3 | May – June 2026 | www.ijamred.com
ISSN: 3107-6513

An Approach to Evaluate and Improve Cloud Application Portability for Multi-Cloud Migration

Komal Ikke
Dept. of Information Technology
University Of Mumbai
Mumbai, India
komalikke20@gmail.com

Nikita Suryavanshi
Dept. of Information Technology
University of Mumbai
Mumbai, India
nikitasuryavanshi2001@gmail.com

Ms. Pooja Tupe
Dept. of Information Technology
University of Mumbai
Mumbai, India
ptupe.105@gmail.com

***Abstract*—Despite the acceptance of cloud computing by many firms, vendor lock-in still constitutes a key challenge to the adoption of cloud computing. Today, there is no quantitative model for assessing the application portability across various cloud service providers during the process of cloud migration. This study proposes a unique Cloud Portability Index (CPI) model that quantitatively assesses the cloud application portability by analyzing five critical dimensions including code dependency, data portability, architectural dependency, operational tooling, and external dependencies. By building upon ISO/IEC 19941:2017 standard on cloud interoperability and portability, this study offers a weighted multi-factor assessment model using mathematical formulas to compute the portability scores. Quantitative measures will be used to assess application portability in all IaaS, PaaS, and SaaS deployment models. The study uses Migration Complexity Index (MCI) to measure the cost of cloud migration based on the application size and level of integration.**

***Keywords*—Cloud Computing, Application Portability, Vendor Lock-in, Multi-Cloud Migration, Cloud Portability Index, Interoperability**

I. Introduction

Cloud computing has significantly reshaped enterprise IT infrastructure by allowing organizations to access on-demand computing resources, automated scalability, and globally distributed infrastructure without requiring major upfront capital investment. Recent industry surveys indicate that 89% of organizations now rely on multi-cloud solutions, while 94% of large enterprises have adopted multi-cloud strategies [1]. Despite these advantages, the rapid growth of cloud adoption has introduced a major challenge that threatens the flexibility cloud computing is intended to provide: vendor lock-in.

Vendor lock-in arises when cloud applications become heavily dependent on proprietary services, APIs, and platform-specific capabilities provided by a single cloud vendor. Opara-Martins et al. [2] conducted an extensive survey involving 114 cloud computing stakeholders and reported that as organizations move computing resources from on-premise environments to the cloud, concerns surrounding vendor lock-in become more severe. Their findings identified integration difficulties and limited data portability as the most significant obstacles. As a result, organizations face considerable barriers when attempting to migrate workloads, including increased technical complexity, costly re-engineering efforts, data migration issues, and rising operational expenses.

The absence of standardization among cloud providers further intensifies these challenges. Major cloud vendors design and maintain proprietary APIs, data structures, and service ecosystems tailored to their own infrastructure and business objectives. Although this enables providers to differentiate their services and improve platform performance, it also creates substantial obstacles for organizations attempting to transition between providers or implement effective multi-cloud architectures. Toosi et al. [4] highlight in their comprehensive survey that portability and interoperability remain critical concerns within interconnected cloud computing environments.

II. Problem Statement

Cloud applications frequently become tightly coupled to a specific cloud provider, making migration to alternative platforms both complex and expensive. Despite the growing importance of cloud portability, there is currently no widely accepted standardized or quantitative method for evaluating how portable an application is across different cloud environments. The lack of standardized portability metrics forces organizations to rely on incomplete data and subjective evaluations when making cloud migration decisions. Consequently, many enterprises encounter unforeseen expenses that may exceed original estimates by 200–300%, prolonged migration timelines that disrupt business objectives, and technical complications that only become apparent after substantial investments have already been made [5]. Petcu and Vasilakos [6] emphasize that the absence of quantitative portability measurement remains a major gap in both cloud computing research and industry practice.

III. Literature Review

A. ISO/IEC 19941:2017 Standard for Cloud Portability

The ISO/IEC 19941:2017 standard [7] provides foundational terminology and conceptual guidance for interoperability and portability in cloud computing environments. The standard differentiates interoperability, defined as the ability of systems to exchange and utilize information, from portability, which refers to the ability to transfer applications or data between cloud services. A key aspect of the framework is

ISSN: 3107-6513

the recognition that portability effectiveness depends on the extent to which migration can be achieved with minimal manual modification. Although the standard establishes important conceptual principles, it does not define specific quantitative methods for evaluating portability, leaving a gap between theoretical understanding and practical measurement.

B. Vendor Lock-in: Technical and Business Perspectives

Opara-Martins et al. [2] present an extensive analysis of vendor lock-in based on a survey of 114 cloud stakeholders. Their study identifies several major technical contributors to lock-in, including proprietary APIs, provider-specific services, custom data formats, and cloud-specific identity management systems. The research also highlights significant business consequences such as increased operational costs, reduced bargaining power, limited innovation opportunities, and decreased strategic flexibility. Silva et al. [3] conducted a systematic review that categorized lock-in mitigation strategies into three groups: technology-oriented approaches, such as abstraction layers and standardized APIs; organizational approaches, including contractual safeguards and exit strategies; and hybrid approaches that combine technical and organizational measures. Similarly, Parameswaran and Chaddha [8] identify interoperability and standardization as fundamental challenges that require coordinated efforts across the cloud computing industry.

C. Standards-Based Portability Approaches

Several standardization initiatives have been developed to address cloud portability challenges. The Topology and Orchestration Specification for Cloud Applications (TOSCA) [9] enables cloud applications to be modeled as portable deployment and management plans. Binz et al. [10] demonstrate that TOSCA-based solutions can significantly reduce deployment complexity and migration effort when effectively implemented. OpenStack [11] offers an open-source Infrastructure-as-a-Service (IaaS) platform that provides consistent APIs across deployments. However, studies show that many OpenStack implementations introduce proprietary extensions that may recreate vendor lock-in concerns. Additional standards include the Cloud Data Management Interface (CDMI) [12], which standardizes cloud data management operations, and the Open Cloud Computing Interface (OCCI) [13], which defines REST-based protocols for infrastructure management.

D. Portability Enhancement Technologies

Containerization technologies, particularly Docker [14], package applications and their dependencies into portable execution environments. Empirical research indicates that containerized applications can reduce migration effort by approximately 40-60% compared to conventional deployment approaches [15]. Kubernetes orchestration platforms [16] further improve portability by providing standardized mechanisms for deploying and managing applications across heterogeneous infrastructure environments. Pahl et al. [17] describe container technologies as one of the most advanced and effective approaches for enhancing cloud portability. Infrastructure as Code (IaC) tools, including Terraform and Pulumi [18], allow infrastructure configurations to be described declaratively and deployed across multiple cloud providers. These approaches improve infrastructure portability and reduce dependency on provider-specific deployment mechanisms.

E. Research Gaps and Motivation

Although substantial research has been conducted on cloud portability and interoperability, several critical gaps remain that limit practical implementation and broader adoption.

Gap 1 – Absence of a Unified Quantitative Framework: There is currently no comprehensive quantitative framework capable of measuring cloud application portability across multiple dimensions within a single normalized metric. Existing methods largely depend on qualitative analysis or isolated single-factor evaluations, despite the inherently multi-dimensional nature of portability.

Gap 2 – Insufficient Empirical Validation: Many proposed portability solutions remain primarily theoretical and lack extensive validation through real-world migration scenarios. This limits confidence in their practical applicability within production-scale cloud environments.

Gap 3 – Limited Automation in Assessment: Current portability evaluation methods often require extensive manual analysis and expert interpretation. This increases assessment complexity, introduces subjective bias, and raises implementation costs.

Gap 4 – Lack of Business-Oriented Evaluation: Existing studies rarely examine the relationship between portability investments and measurable business outcomes. Consequently, organizations lack clear guidance for prioritizing portability improvements or justifying investments in portability-enhancing technologies.

This study addresses these research gaps by proposing a comprehensive quantitative framework supported by mathematical formulations, systematic assessment methodologies, and alignment with industry standards. The framework is intended to support objective, data-driven cloud migration and portability decision-making.

IV. Proposed Methodology

Figure 1 illustrates the overall architecture and workflow of the proposed Cloud Portability Assessment System, highlighting the interaction between analysis modules, assessment engine, and reporting components.

A. Cloud Portability Index Framework

The Cloud Portability Index provides a normalized score between 0 and 1 quantifying overall application portability. The overall CPI is calculated as:

$$\mathrm{CPI} = \sum_{i=1} w_i \cdot P_i \qquad (1)$$

where $P_1 = P_{\text{code}}$, $P_2 = P_{\text{data}}$, $P_3 = P_{\text{arch}}$, $P_4 = P_{\text{ops}}$, $P_5 = P_{\text{deps}}$ represent the five portability dimensions, and weights w_i

ISSN: 3107-6513

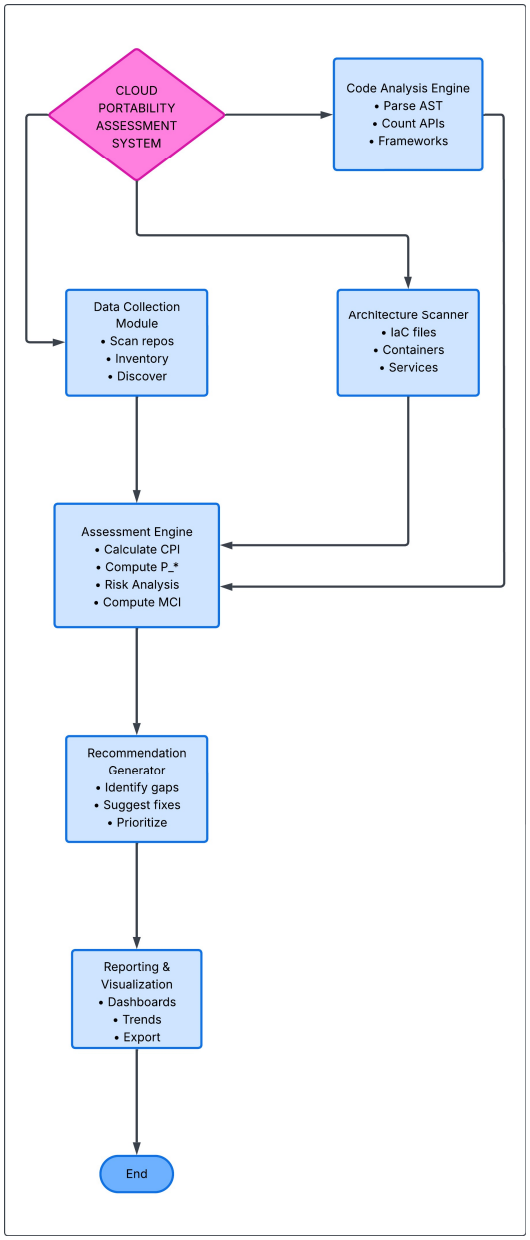

Fig. 1. Architecture and workflow of the Cloud Portability Assessment System

satisfy $\sum_{i=1}^{5} w_i = 1.0$. Default weights based on migration effort analysis are: $w_1 = 0.25$, $w_2 = 0.20$, $w_3 = 0.25$, $w_4 = 0.15$, $w_5 = 0.15$.

he Cloud Portability Index is computed as a weighted combination of five portability dimensions. This method enables the framework to balance different concerns involved in portability assessment. Each dimension P_i is assigned a weight w_i representing its relative importance and expected migration effort. Code (0.25) and Architecture (0.25) receive the highest weights because they usually require the greatest refactoring effort during migration. Data (0.20) is given moderate weight since data migration, although complex, is generally more manageable than code modification. Operations (0.15) and Dependencies (0.15) receive lower weights because they can often be handled through tooling and configuration instead of major application changes. The constraint $\sum w_i = 1.0$ ensures the final CPI remains normalized between 0 and 1.

Table I presents CPI score interpretation guidelines.

TABLE I
CPI SCORE INTERPRETATION GUIDELINES

CPI Range	Classification	Interpretation
CPI ≥ 0.7	High Portability	Low migration risk; suitable for multi-cloud
0.4 ≤ CPI < 0.7	Medium Portability	Moderate effort; improvement recommended
CPI < 0.4	Low Portability	Critical lock-in; significant refactoring required

B. Dimensional Portability Metrics

1) Code Portability Score: Code portability evaluates application source code dependencies on vendor-specific implementations:

$$P_{\text{code}} = (1 - R_{\text{vendor}}) \times P_{\text{framework}} \times P_{\text{language}} \quad (2)$$

Code portability is calculated as a multiplicative combination of three factors. This multiplicative approach ensures that weak performance in any one factor significantly lowers the overall code portability score, reflecting the practical reality that successful migration depends on addressing all three aspects.

Vendor API Ratio :

$$R_{\text{vendor}} = \frac{N_{\text{vendor}}}{N_{\text{total}}} \quad (3)$$

This ratio measures the proportion of vendor-specific API calls within the codebase. N_{vendor} denotes the number of API calls made to proprietary cloud provider services (e.g., AWS Lambda-specific APIs, Google Cloud Functions, Azure-specific SDKs), while N_{total} represents the total number of API calls in the application. The expression $(1 - R_{\text{vendor}})$ converts this into a portability metric where 0% vendor-specific calls results in 1.0 (perfect portability), 30results in 0.7 (moderate portability), and 100% vendor-specific calls results in 0 (no portability).

Framework Portability :

$$P_{\text{framework}} = \frac{N_{\text{portable}}}{N_{\text{frameworks}}} \quad (4)$$

This measures the proportion of frameworks that are portable across different cloud providers. Portable frameworks include standard technologies such as Spring Boot, Express.js, and Django, while cloud-specific frameworks include AWS SAM and Google Cloud Functions Framework.

ISSN: 3107-6513

TABLE II
LANGUAGE PORTABILITY

$P_{language}$	Case
1.0	for standard languages (Java, Python, Go)
0.7	for vendor-extended languages
0.3	for proprietary languages

Language portability is determined based on language characteristics: 1.0 for standard languages that execute consistently across platforms, 0.7 for vendor-extended languages, and 0.3 for proprietary languages limited to a specific provider.

Example Calculation: An application with 150 vendor-specific API calls out of 500 total ($R_{vendor} = 0.30$), 3 portable frameworks out of 4 total ($P_{framework} = 0.75$), and standard Java language ($P_{language} = 1.0$) results in:

$$P_{code} = (1 - 0.30) \times 0.75 \times 1.0 = 0.525 \quad (5)$$

Code portability applies a multiplicative combination of vendor API ratio, framework portability, and language portability to evaluate dependence on vendor-specific implementations.

2) Data Portability Score: Data portability evaluates the ease of extracting, transferring, and importing data across cloud platforms:

$$P_{data} = \alpha \cdot S_{format} + \beta \cdot S_{volume} + \gamma \cdot S_{export} + \delta \cdot S_{schema} \quad (6)$$

with default weights $\alpha = 0.3$, $\beta = 0.2$, $\gamma = 0.3$, $\delta = 0.2$ ($\Sigma = 1$), where:

Data portability is calculated as a weighted sum of four sub-scores. This additive approach allows partial credit for good performance in some dimensions even if others are problematic.

Format Score (weight 0.3):

$$S_{format} = \frac{N_{standard\_formats}}{N_{total\_formats}} \quad (7)$$

Format Score measures data format standardization as the ratio of standard formats (CSV, JSON, XML, Parquet, Avro) to total formats. The higher weight (0.3) reflects that format compatibility is often the main concern in data portability. Proprietary formats refer to vendor-specific binary formats that require specialized tools for access.

Volume Score (weight 0.2):

$$S_{volume} = 1 - \frac{Data\_Size\_GB}{Threshold\_GB} \quad (8)$$

Volume Score evaluates the practicality of data transfer based on dataset size, where Threshold_GB represents the practical migration limit (e.g., 1000 GB for a typical enterprise migration). Scores are constrained between 0 and 1: small datasets (< 100 GB) score close to 1.0 (high portability), medium datasets (500 GB) score 0.5 (moderate migration effort), and large datasets (> 1000 GB) approach 0 (major challenge).

TABLE III
EXPORT SCORE

S_{export}	Case
1.0	if automated export APIs are available
0.6	if manual export process required
0.3	if export severely restricted or expensive
0.0	if export not permitted

Volume Score (weight 0.2):

$$S_{schema} = \frac{N_{portable\_schemas}}{N_{total\_schemas}} \quad (9)$$

Schema Score measures schema portability as the ratio of portable schemas (SQL-standard, JSON Schema, Avro Schema) to total schemas. Proprietary schemas refer to vendor-specific database schemas that include non-standard extensions.

3) Architecture Portability Score: Architecture portability evaluates structural design independence from cloud-specific architectural patterns:

$$P_{arch} = \frac{S_{abstraction} + S_{container} + S_{IaC}}{3} \quad (10)$$

Architecture portability is the arithmetic mean of three architectural characteristics. The equal weighting (1/3 each) reflects that all three factors are important for achieving architectural portability.

Service Abstraction Score:

$$S_{abstraction} = 1 - \frac{N_{proprietary\_services}}{N_{total\_services}} \quad (11)$$

Service Abstraction Score measures independence from cloud-specific services. Proprietary services include AWS Lambda and Google Cloud Functions (platform-specific), while portable services include Kubernetes deployments and standard databases. Higher scores indicate better service abstraction.

Container Score:

TABLE IV
CONTAINER SCORE

$S_{container}$	Case
1.0	if fully containerized with standard runtimes (Docker/OCI)
0.7	if partially containerized
0.4	if using cloud-specific compute services
0.0	if tightly coupled to VM images

ISSN: 3107-6513

Container Score evaluates containerization level. Containerization is critical for portability as it encapsulates application dependencies.

IaC Score:

TABLE V
INFRASTRUCTURE AS CODE (IAC) SCORE

S_{IaC}	Case
1.0	if using cloud-agnostic IaC tools (Terraform, Pulumi)
0.6	if using cloud-specific IaC with abstraction layers
0.3	if using cloud-specific tools (CloudFormation, ARM templates)
0.0	if manual infrastructure management

IaC Score assesses infrastructure definition portability. IaC portability enables infrastructure replication across clouds.

4) Operational Portability Score: Operational portability measures the ease of replicating operational procedures, monitoring, and management across platforms:

$$P_{ops} = \frac{S_{monitor} + S_{log} + S_{auto} + S_{security}}{4} \quad (12)$$

Operational portability is the arithmetic mean of four operational dimensions. Equal weighting reflects that all operational aspects must be portable for successful migration.

TABLE VI
MONITORING SCORE

$S_{monitor}$	Case
1.0	if using standard protocols (Prometheus, OpenTelemetry)
0.6	if using vendor-agnostic tools
0.3	if dependent on vendor-specific monitoring
0.0	if no monitoring portability

TABLE VII
LOGGING SCORE

S_{log}	Case
1.0	if exporting to standard formats (syslog, JSON)
0.6	if centralized but vendor-specific
0.3	if locked to vendor logging services
0.0	if no log export capability

Automation Score:

$$S_{auto} = \frac{N_{portable\_automations}}{N_{total\_automations}} \quad (13)$$

Automation Score measures the ratio of portable automations to total automations. Portable automation includes standard CI/CD tools (GitOps, Ansible), while proprietary includes vendor-specific orchestration.

Security Score:

$$S_{security} = \frac{S_{auth} + S_{encryption}}{2} \quad (14)$$

where $S_{auth} = 1.0$ if OAuth/SAML, 0.5 if vendor-specific; and $S_{encryption} = 1.0$ if standard algorithms, 0.5 if vendor KMS only.

5) Dependency Portability Score: Dependency portability evaluates external service dependencies and their portability characteristics:

$$P_{deps} = 1 - \frac{\sum_{i=1}^{N} (C_i \times L_i)}{N} \quad (15)$$

where N is total dependencies, C_i measures criticality, and L_i quantifies vendor coupling.

Dependency portability evaluates external service dependencies. The formula computes the average lock-in risk across all dependencies and subtracts it from 1 to convert the value into a portability score.

TABLE VIII
CRITICALITY FACTOR

C_i	Case
1.0	Critical to application function (payment, authentication)
0.6	Important but alternatives exist (caching, queuing)
0.3	Optional / enhancement features (analytics)

TABLE IX
LOCK-IN FACTOR

L_i	Case
1.0	Proprietary with no alternatives (AWS-specific)
0.6	Limited alternatives exist (2–3 provider options)
0.3	Multiple providers available (standard APIs)
0.0	Open-source or standard protocol

The product $(C_i \times L_i)$ represents the lock-in risk associated with dependency i. Averaging across all N dependencies provides the overall dependency risk, which is then subtracted from 1 to produce a portability score. High portability (0.8–1.0) indicates mostly open-source dependencies, medium portability (0.4–0.7) reflects a mix of proprietary and portable dependencies, and low portability (0–0.4) indicates strong reliance on proprietary services.

Example Calculation: For 3 dependencies: (1) Authentication (Critical, Proprietary): $C = 1.0$, $L = 1.0 \rightarrow 1.0$;

ISSN: 3107-6513

(2) Caching (Important, Multiple providers): $C = 0.6$, $L = 0.3 \rightarrow 0.18$; (3) Analytics (Optional, Open-source): $C = 0.3$, $L = 0.0 \rightarrow 0.0$. Thus, $P_{deps} = 1 - [(1.0 + 0.18 + 0.0)/3] = 1 - 0.39 = 0.61$.

C. Migration Complexity Index

The Migration Complexity Index estimates the relative effort required for cloud migration:

$$MCI = (1 - CPI) \times F_{size} \times F_{integration} \quad (16)$$

The Migration Complexity Index estimates migration effort by combining portability assessment with application characteristics. The multiplicative relationship means that low portability, larger application size, and high integration complexity together increase migration effort.

Portability Factor:

The term $(1 - CPI)$ converts the CPI into a complexity measure: high portability (CPI = 0.8) produces a complexity factor of 0.2, medium portability (CPI = 0.5) produces 0.5, and low portability (CPI = 0.2) produces 0.8.

Size Factor:

$$F_{size} = \frac{\log_{10}(KLOC + 100)}{3} \quad (17)$$

The Size Factor uses logarithmic scaling to reflect that migration effort does not increase linearly with code size. The constant 100 prevents extreme values for very small applications. For example: 10 KLOC gives $F_{size} = \log_{10}(110)/3 = 0.68$; 100 KLOC gives 0.77; and 1000 KLOC gives 1.01. Division by 3 normalizes the factor within practical ranges.

Component 3 – Integration Factor:

$$F_{integration} = 1 + (N_{integrations} \times 0.2) \quad (18)$$

The Integration Factor applies linear scaling where each external system integration adds 20

Example Calculation: An application with CPI = 0.6, 200 KLOC, and 8 external integrations: $F_{size} = \log_{10}(300)/3 = 0.83$; $F_{integration} = 1+(8\times0.2) = 2.6$; MCI = $(1-0.6)\times0.83\times 2.6 = 0.86$. The result, MCI = 0.86, indicates low migration effort (< 1.0), with an estimated migration duration of 1–2 months.

Interpretation Thresholds: MCI < 1.0 indicates low effort (1–2 months) suitable for straightforward migration; $1.0 \leq$ MCI < 3.0 indicates moderate effort (3–6 months) for standard migration projects; and MCI ≥ 3.0 indicates high effort (6+ months) requiring substantial resources.

D. Assessment Methodology Workflow

The systematic assessment workflow consists of six phases:

Phase 1 - Discovery: Identify all components, map services and dependencies, document system architecture.

Phase 2 - Analysis: Conduct static code analysis (AST parsing), perform dependency mapping, review data schemas.

Phase 3 - Scoring: Calculate P_{code}, P_{data}, P_{arch}, P_{ops}, P_{deps}, and compute overall CPI.

Phase 4 - Risk Analysis: Classify risk level (High/Medium/Low), identify critical gaps, compute Migration Complexity Index.

Phase 5 - Recommendation: Generate improvement actions, prioritize by impact and effort, create improvement roadmap.

Phase 6 - Monitoring: Track changes over time, measure improvement progress, conduct periodic reassessment.

V. Discussion

A. Framework Advantages

The CPI framework addresses several limitations identified in existing literature by introducing objective numerical scoring mechanisms that support consistent and repeatable portability assessment, unlike predominantly qualitative approaches [2], [6]. Its multi-dimensional structure enables a comprehensive evaluation of portability rather than focusing on isolated technical factors [3]. In addition, the mathematical formulation of the framework supports automation, thereby reducing dependence on expert interpretation and minimizing subjective bias. Compared with traditional effort-based evaluation methods [6], the CPI framework enables prospective assessment, allowing organizations to estimate portability before migration activities begin rather than evaluating outcomes retrospectively. Furthermore, the framework extends the conceptual principles established by ISO/IEC 19941:2017 [7] by introducing practical and measurable quantitative methodologies.

B. Practical Implications

The CPI framework provides organizations with a structured basis for cloud provider selection, migration planning, and architectural decision-making. Applications achieving a CPI score of ≥ 0.7 can generally be considered suitable for multi-cloud deployment strategies [19], whereas applications with CPI scores below 0.4 require careful evaluation due to elevated vendor lock-in risks. The dimensional analysis offered by the framework delivers actionable insights by identifying weak portability areas, enabling organizations to prioritize modernization and remediation efforts according to the lowest-performing dimensions. Additionally, the framework supports continuous portability monitoring, allowing development teams to track changes over time and prevent portability degradation as systems evolve.

C. Integration with Industry Standards

The CPI framework is aligned with the principles of ISO/IEC 19941:2017 [7] while extending the standard through concrete quantitative assessment methods that operationalize its conceptual guidance. The framework is also compatible with established portability-enhancing technologies and standards, including TOSCA-based deployment orchestration [9], containerization technologies [17], and Infrastructure as Code (IaC) tools [18]. Organizations already adopting these technologies can integrate CPI-based assessments to validate portability objectives and systematically measure improvements over time.

D. Limitations and Considerations

Although the framework provides standardized evaluation mechanisms, the default weight distributions are designed to represent generalized priorities and may require adjustment to reflect organization-specific objectives and operational contexts. In addition, the framework assumes that increased portability is inherently beneficial; however, organizations must balance portability goals against other considerations such as performance optimization, development speed, operational complexity, and cost efficiency. While automated assessment tools can identify many portability concerns, certain complex dependencies involving business logic, organizational workflows, and legacy integrations still require expert human evaluation. Moreover, the framework evaluates portability at a specific point in time, making continuous assessment essential for preventing portability regression as cloud applications and infrastructure evolve.

VI. Conclusion and Future Scope

This research addresses the challenge of vendor lock-in by proposing a comprehensive Cloud Portability Index (CPI) framework that enables quantitative portability assessment across five key dimensions: code, data, architecture, operations, and dependencies. The primary contributions of this study include: (1) the development of a unified quantitative framework using normalized scores ranging from 0 to 1, (2) mathematical formulations for evaluating five distinct portability dimensions, (3) a systematic assessment methodology designed to reduce dependence on expert judgment, (4) the introduction of a Migration Complexity Index (MCI) for estimating migration effort, and (5) alignment with the ISO/IEC 19941:2017 standard while extending it through practical measurement and assessment mechanisms.

The CPI framework supports evidence-based cloud adoption and migration decisions by transforming subjective architectural evaluations into measurable, data-driven analyses. Through standardized scoring and dimensional assessment, organizations can compare architectural alternatives, monitor modernization progress, identify portability weaknesses, and communicate vendor lock-in risks more effectively to both technical and business stakeholders.

Several opportunities exist for future research and further enhancement of the framework. These include large-scale empirical validation through real-world migration case studies across different industries and application types, development of automated assessment tools integrated with CI/CD pipelines, and the application of machine learning techniques to determine optimal dimension weight distributions. Additional research may also extend the framework to incorporate dimensions such as security compliance, regulatory portability, and cost portability. Furthermore, collaboration with ISO/IEC standardization initiatives could support broader industry adoption and refinement of quantitative portability assessment practices.

Vendor lock-in remains one of the most significant obstacles to fully realizing the flexibility and strategic benefits promised by cloud computing. By providing a structured and quantitative approach to portability assessment, the Cloud Portability Index framework offers organizations a practical foundation for evaluating, improving, and managing cloud portability in increasingly complex multi-cloud environments.

Acknowledgements

I am grateful to Prof. Shrivarmangai for his guidance about various topics, including Cloud Computing, throughout this project.

References

[1] Flexera, “2023 State of the Cloud Report,” Technical report, Flexera Software LLC, 2023.
[2] J. Opara-Martins, R. Sahadi, and F. Tian, “Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective,” *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 5, no. 1, pp. 1–18, 2016.
[3] G. C. Silva, L. M. Rose, and R. Calinescu, “A systematic review of cloud lock-in solutions,” in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, pp. 363–368, IEEE, 2013.
[4] A. N. Toosi, R. N. Calheiros, and R. Buyya, “Interconnected cloud computing environments: Challenges, taxonomy, and survey,” *ACM Computing Surveys*, vol. 47, no. 1, pp. 1–47, 2014.
[5] D. Petcu, B. D. Martino, S. Venticinque, M. Rak, T. Ma´hr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, S. S˘perka, and V. Stankovski, “Experiences in building a mOSAIC of clouds,” *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, pp. 1–22, 2013.
[6] D. Petcu and A. V. Vasilakos, “Portability in clouds: approaches and research opportunities,” *Scalable Computing: Practice and Experience*, vol. 15, no. 3, pp. 251–270, 2014.
[7] ISO/IEC 19941:2017, “Information technology — Cloud computing — Interoperability and portability,” International Organization for Standardization, Geneva, Switzerland, 2017.
[8] A. V. Parameswaran and A. Chaddha, “Cloud interoperability and standardization,” *SETLabs Briefings*, vol. 7, no. 7, pp. 19–26, 2009.
[9] OASIS, “Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0,” Organization for the Advancement of Structured Information Standards, 2013.
[10] T. Binz, U. Breitenbu¨cher, O. Kopp, and F. Leymann, “TOSCA: Portable automated deployment and management of cloud applications,” in *Advanced Web Services*, pp. 527–549, Springer, New York, 2014.
[11] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “OpenStack: Toward an open-source solution for cloud computing,” *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.
[12] SNIA, “Cloud Data Management Interface (CDMI) Version 1.1.1,” Storage Networking Industry Association, 2015.
[13] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson, “Toward an open cloud standard,” *IEEE Internet Computing*, vol. 16, no. 4, pp. 15–25, 2012.
[14] D. Merkel, “Docker: Lightweight Linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, Article 2, 2014.
[15] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and Linux containers,” in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 171–172, IEEE, 2015.
[16] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, Omega, and Kubernetes,” *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
[17] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, “Cloud container technologies: A state-of-the-art review,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 677–692, 2019.
[18] K. Morris, *Infrastructure as Code: Managing Servers in the Cloud*. O’Reilly Media, Inc., Sebastopol, CA, 2016.
[19] Cloud Standards Customer Council, “Interoperability and Portability for Cloud Computing: A Guide Version 2.0,” Technical report, 2017.