Jugo: A Generic Architecture for Composite Cloud as a Service

Mahmud Hossain, Rasib Khan, Shahid Al Noor, and Ragib Hasan
SECRETLab, Department of Computer and Information Sciences
University of Alabama at Birmingham, AL, USA
{mahmud, rasib, shaahid, ragib}@cis.uab.edu

***Abstract*—Cloud computing has become the industry standard for rapid application deployment, scalable server support, mobile and distributed services, and it provides access to (theoretically) infinite resources. Unfortunately, researchers are still trying to converge towards cross-provider cloud computing frameworks to enable compatibility and seamless resource transition between cloud providers. Moreover, users are restricted to using the provider-specific pre-configured options of resources and services, irrespective of their current needs. At the same time, cloud services are provided as a direct service from the providers to the clients. This creates a segregated cloud market clientele, and non-negotiable pricing strategies for the cloud services. In this paper, we propose Jugo, a generic architecture for cloud composition and negotiated service delivery for cloud users. Jugo acts as a match-maker for service specifications from the users with the currently available assets from the cloud providers. The engagement of a middle-man as an opaque cloud service provider will create a better opportunity for cloud users to find cheaper deals, price-matching, and flexible resource specifications, with increased revenue and higher resource utilization for the cloud service providers.**

***Keywords*-Cloud as a Service, Composite Cloud, Multi-Provider Cloud, Opaque Cloud Service**

I. Introduction

The increasing popularity of cloud computing is introducing more players into the cloud market. Cloud service providers deliver services to their clients directly, with the pre-configured set of service descriptions, irrespective of the clients' requirements for much granular specification flexibility. The segregated market for cloud users creates a restricted business model for the users, as well as for new cloud providers intending to enter the market [1]. Hence, the cloud users are coerced towards full-priced services without the effect of demand-supply equilibrium in the cloud market [1, 2]. Furthermore, resource fragmentation is a major concern for cloud providers to improve the overall resource utilization [3–7]. Resource migration and efficient resource allocation strategies might be able to reduce the fragmented resources [8, 9]. However, resource fragmentation problem becomes persistent because of implementing complex, non-scalable, and unproven solutions for fragmentation removal. Additionally, novel cloud computing frameworks based on mobile, ad-hoc, and localized devices [10–12] are proposed as disassociated platforms, without service composition approaches to merge the service availability via a generic service gateway.

There are various niche consumer markets operating with multiple providers [13–15]. Instead of only a direct service delivery, such services have developed an opaque service model (e.g., Priceline[1], Airbnb[2], Uber[3]). An opaque service provider acts as a negotiator between the end user and the service provider in terms of prices and services [16, 17]. Opaque service models involve an enhanced marketing strategy to acquire more consumers [18]. Such form of services have proven effective in various market niches, enhancing upto 33% of the core providers' revenue [19–21]. Our complimentary work, the Litigo model, has already illustrated the benefits of an opaque player in the cloud market [19].

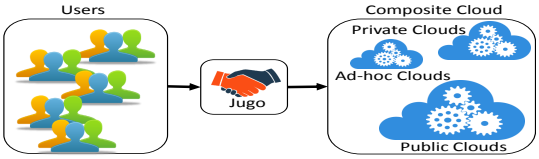

Fig. 1: Jugo Architecture Overview

In this paper, we present Jugo[4], a generic architecture for delivering composite cloud services. Jugo provides a service negotiation approach for cloud users based on their individual resource requirements. Jugo aggregates the asset availability from multiple cloud service providers to allocate the requested resources for the users. We posit that the Jugo opaque service model creates room for novel cloud services, such as resource aggregation from multiple providers, price-matching, flash deals, fine-grained service negotiations, and opportunities for fragmented resource utilization for cloud providers.

The rest of the paper is organized as follows. The architecture and operational model for Jugo are presented in Section II and Section III respectively. Section IV presents the related work. Finally, we provide the conclusion in Section V.

II. The Jugo Architecture

In this section, we present the detailed architecture of Jugo and the corresponding components, as illustrated in Figure 2.

A. Overview

A CSP provides Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The Jugo composite cloud is formed by combining all the cloud services for various cloud types, such as public cloud providers, private clouds willing to rent out unused resources, and other cloud frameworks, as shown in Figure 1. Researchers have proposed ad-hoc clouds considering the mobile and stationary computing devices in the nearby physical environment, and may also be included within the Jugo architecture [10–12].

As shown in the overview (Fig 1), users utilize Jugo to request for cloud resources. The users provide the specifications and the price for the cloud resources. Next, Jugo negotiates the service requirements with the pool of CSPs to find the best price and resource matching service contract. Finally, the user is delivered the service and redirected to the corresponding CSP. The CSP verifies the redirected request from Jugo and grants access for the requested resources to the user.

[1] www.priceline.com [2] www.airbnb.com [3] www.uber.com

[4] **Jugo** is a Latin synonym for '*to aggregate*'.

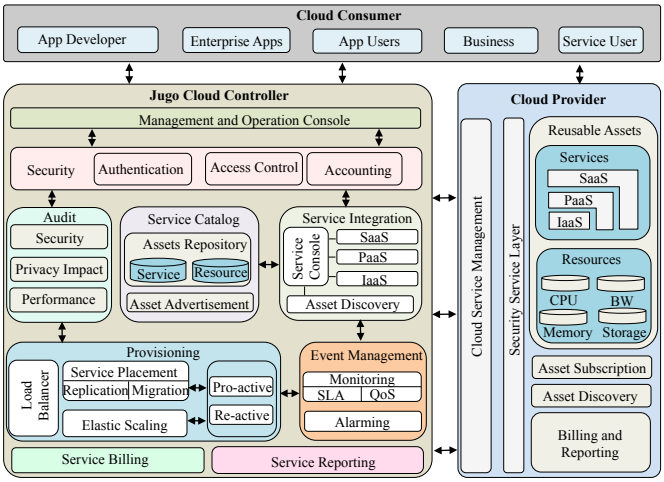

Fig. 2: Detailed Architecture of Jugo

B. Jugo Cloud Service Providers

1) Reusable Assets: Reusable assets are the small fragmented idle resources of a CSP and difficult to allocate to the users [9] because of satisfying their service level agreements (SLA) requirements. A CSP keeps track of such reusable assets in an asset database (see Algorithm 1) where Jugo can inquire and receive a price lower than the regular price. Consequently, the CSP increases the resource utilization and makes additional profit. CSPs can also sell resources in bulk to Jugo at a discounted price in order to increase their resource utilization.

2) Asset Subscription Service (ASB): CSPs advertise their reusable assets through ASBs. Jugo needs to subscribe to the ASB of a CSP to receive the advertisements. Jugo receives the periodic advertisements and registers the available assets from various CSPs (see Algorithm 2).

Algorithm 1: Retrieve Reusable Assets

```
for datacenter in DataCenterList do
    for service in dc.getServiceList() do
        if service == IaaS then
            var resList = service.getFragmentedResocues()
            insert restList into DB.IaaS.ResourceTable
        else if service == PaaS then
            var svcList = service.getIdlePlatforms()
            insert svcList into DB.PaaS.ServiceTable
        else if service == SaaS then
            var svcList = service.getIdleServices()
            insert svcList into DB.SaaS.ServiceTable
        end if
    end for
end for
```

3) Asset Discovery Service (ADS): The Jugo provider may also contact a CSP to retrieve further description of the assets via the ADS interface. Unlike ASB, ADS does not require any subscription and allows Jugo to obtain on-demand asset information from the CSPs.

C. Jugo Cloud Controller

The Jugo Cloud Controller is composed of the following service components for cloud users and CSPs.

1) Service Catalog (SVC): The SVC requests the ADS to compile reusable assets from multiple CSPs. ADS retrieves assets' specifications from the CSPs. Eventually, the SVC delivers the aggregated asset information to the users through the Asset Advertisement service (see Algorithm 3).

Algorithm 2: Advertise Reusable Assets

```
var prevAssetList = NULL
while true do
    var currentAssetList= retrieveReusableAsset()
    if prevAssetList ≠ currentAssetList then
        newAssetList = getUpdateAssets(
        prevAssetList, currentAssetList)
        for cc in getCloudControllerList() do
            sendUpdate(cc.IP, cc.Port, newAssetList)
        end for
    end if
    prevAssetList = currentAssetList
    Thread.Sleep(DELAY_PERIOD)
end while
```

Algorithm 3: Asset Compilation

```
for csp in getCloudProviderList() do
    for sv in csp.getserviceList() do
        insert sv.Specification into DB.Service.Table
    end for
    for res in csp.getResouceList() do
        insert res.Specification into DB.Res.Table
    end for
end for
```

2) Service Integration Engine (SIE): SIE integrates cloud services from multiples CSPs, and provides a unified application interface to the users. SIE also provides mechanisms to add and configure services from heterogeneous clouds (see Algorithm 4). The SIE exposes web and application programming interfaces (APIs) via the Management and Operation Console (MOC). Jugo provides unified APIs to handle heterogeneous service APIs of CSPs that are used by cloud users to build and deploy applications in any cloud platform. The protocol translation and management are done by the IaaS/PaaS/SaaS manager of the Service Console module.

3) Provisioning Service (PS): Resources can be pro-actively or re-actively balanced by replicating and migrating the services to the closest or larger CSPs via the Load Balancer. PS can also elastically allocate more resources for the users from multiple CSPs depending on user-defined provisioning specifications.

4) Event Management Service (EMS): EMS consists of two modules: Monitoring and Alarming. The Monitoring module periodically checks and verifies the SLA and Quality of Service (QoS), and generates log provenance for the given CSP. The SLA logs generated and preserved by the Monitoring module are used during the performance audit phase. The Alarming module allows the users to manage the usage of resources. A user can specify usage threshold parameters for the allocated services and to define the required actions that will be performed when a metric exceeds the threshold using the Alarming module.

5) Auditing Service: This service is composed of three components: security audit module, privacy impact audit module, and performance audit module. Jugo acts as a gateway between the cloud consumers and the CSPs. The Auditing module periodically logs the activities among the users, CSPs, and the Jugo Cloud Controller, and maintains the secure provenance of the logs for post-verifiable auditing [22, 23]. The Auditing service also provides an interface for auditors to retrieve and analyze the stored logs for forensic investigation.

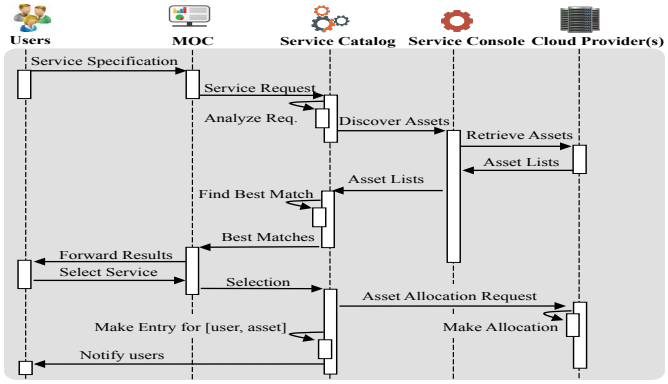

Fig. 3: Service selection process.

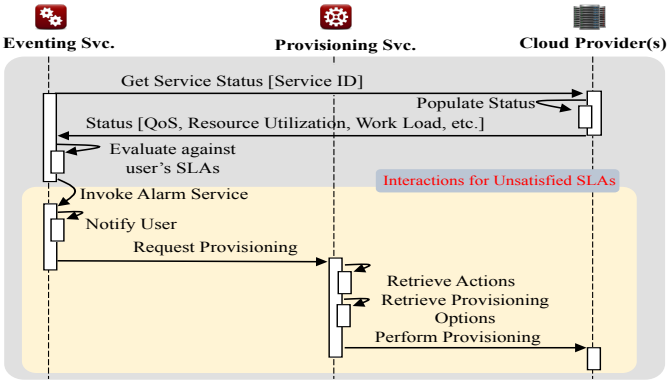

Fig. 4: Service provisioning process.

Algorithm 4: Cross-Protocol Service Gateway

Input: inRequest
1: var $svcManager$ = NULL
2: **switch** ($inRequest.TYPE$)
3: **case** $IaaS$:
4: $svcManager$ = IaaS.Manager
5: **case** $PaaS$:
6: $svcManager$ = PaaS.Manager
7: **case** $SaaS$:
8: $svcManager$ = SaaS.Manager
9: **end switch**
10: var msg = svcManager.TranslateMsg($inRequest.Msg$, $inRequest.CloudProvider$)
11: svcManager.InvokeAPI(msg, $inRequest.MethodName$, $inRequest.CloudProvider$)

6) Security Service: The Security Service provides authentication, access control, and accounting for cloud users and CSPs. Authentication service (AuthS) manages users' credentials and manages Service Access Tokens (SAT). AuthS can issue SATs to verified users for accessing inbound (e.g., provisioning and auditing services) and outbound (e.g., IaaS/PaaS/SaaS service) cloud services based on the defined privileges for services and resources. The Jugo Cloud Controller provides support for open-source and cross-platform authentication protocols [24]. Hence, verified and provisioned Jugo users can bypass the Jugo Cloud Controller and can directly access the cloud resources by presenting the SATs. Access Control Service (ACS) provides policies to define service and resource privileges for each user.

7) Service Reporting and Billing: The reporting module provides reports on service usage and SLAs. The reporting module generates aggregated usage reports for individual CSPs. Similarly, the billing includes the charges for Jugo as well as for the allocated resources from the CSPs.

8) Management and Operation Console (MOC): The MOC is the contact point for the Jugo Cloud Controller. The MOC interface enables users to interact with Jugo's services as well as with the CSPs. The interface allows Jugo users to explore available resources, submit requests, subscribe updates from CSPs, manage and configure Jugo services, monitor usage statistics, and retrieve reporting and billing information.

III. Operational Model

Next, we present three use cases and the corresponding operational models for Jugo.

A. Service Selection

Service selection describes the interactions to purchase services through Jugo (Figure 3). A user submits service specifications (see Table I) to the SVC through the MOC. The SVC analyzes the user's requirements, followed by discovery and compilation of available assets from multiple CSPs. The SVC performs the specification request vs. resource availability negotiation and determines the best deals for the client at the given time. The client selects and notifies the SVC with the desired service(s) from the available offers. Finally, the SVC requests the corresponding CSP(s) to allocate the selected resources for the given service(s).

TABLE I: Jugo Message Formats

(a) Service Specification

Request [List[Services]]
Services [List[Service]]
Service [Type, List[Resources], List[SLA], PriceRange, ServiceLocation, NumberOfUsers, ServicePeriod]
Resources [Resource$_x$, $x \in R$]
SLA [QoS, Throughput, PowerBackup, Security, List[Provisioning], ServiceCommunity]
Provisioning [Action, Options]

(b) Security Access Token (SAT)

SAT [UserId, IssueInstance, Issuer, Subject, PublicKey, Signature, SignHash, List[Services]]
Services [List[Service]]
Service [Address, List[Capabilities]]
Capabilities [Actions, Obligation, List[AccessLocations]]
Obligation [NotBefore, NotAfter]
AccessLocations [Locations$_x$, $x \in R$]

B. Service and Resource Provisioning

The Eventing Service periodically monitors the performance and status of the allocated CSP services via Jugo (see Figure 4). Eventing Service evaluates the service status against users' SLAs and raises a notification alarm in case of any variation of the QoS parameters. The Provisioning Service is requested to take necessary actions (e.g., resource scale up, migration, and replication) to improve service performance and to maintain the SLAs and QoS parameters. The Provisioning Service retrieves provision actions and options (e.g., on-demand measure and proactive measure), and re-acts accordingly.

C. Authentication and Access Control

The Jugo security service issues SATs to the existing and authenticated Jugo users. The users can then use the SATs to access the allocated resources at the CSPs (see Figure 5). A user authenticates with the Jugo and requests a SAT for the purchased resources. The Authentication Server verifies the user and retrieves the privileges from the Authorization Server. Next, the Authentication Server generates a signed SAT with the user capabilities and issues the SAT to the user. Subsequently, the Jugo user sends a service access request with the SAT to a CSP. The CSP verifies the SAT and checks privileges for the

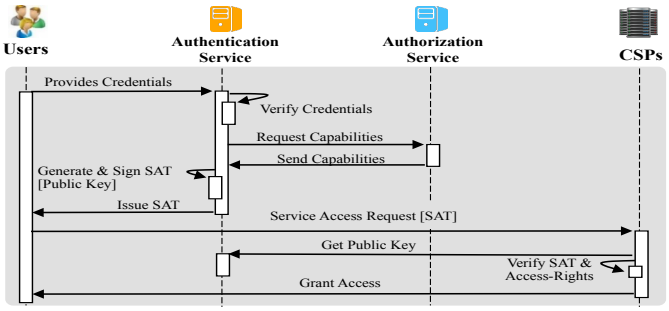

Fig. 5: Authentication and Access Token Generation

delegated resource access, and then grants/denies the request according to the verification result.

The message format for SATs is presented in Table I. A SAT describes a list of allowed services and capabilities for each of the services (e.g. GET, POST, etc.). A SAT also contains a list of verifiable obligation parameters for a request to be served (e.g. not before, not after). The SAT specifies the access locations for the service request, along with the user's identity, resource name, CSP's name, user's contact, Jugo's public key, SAT-signature and SAT-hash.

IV. Related Work

Several schemes have been proposed to unify multiple cloud services and their functionalities [25, 26]. To reduce vendor lock-in and to improve portability, Costa et al. [27] presented a general purpose high level Cloud API. A common management interface for the IaaS users has been proposed in [28]. Rochwerger et al. [29] proposed a service oriented framework to allow dynamic interoperability of cloud service providers. Maximilien et al. [30] presented a cloud-agnostic middleware broker for cloud users. An open-standard abstraction layer for cross-platform cloud services have been proposed in [31]. Additionally, researchers have proposed third-party broker based strategies to select cloud resources based on user requirements [32–35]. Unlike Jugo, these works did not consider composite platforms with multiple cloud service providers with different service capabilities. Jugo focuses on a unified platform for service negotiation, delivery, and management between different types of cloud resources and the users. Jugo can leverage the above designs and concepts to provide open-standards for such composite cloud architectures.

V. Conclusion

The proposed Jugo model introduces a novel architecture for composite service delivery for multiple cloud providers. The concept leverages the opaque marketing approach providing users with cloud resources via service negotiation and price-matching assurance. Jugo enables cloud service providers to rent out under utilized and fragmented resources, resulting in increased profits and resource utilization. Moreover, Jugo may also promote smaller cloud providers and private cloud resources willing to rent out idle resources and enter the cloud computing market niche. We posit that increasing the number of Jugo-contracted cloud providers will eventually benefit the cloud users by creating better service deals, the Jugo providers by receiving increased service revenue, as well as the cloud service providers by improving the quality of services.

VI. Acknowledgements

This research was supported by the National Science Foundation CAREER Award CNS-1351038.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *IEEE HPCC*, 2008.

[2] C. Péloquin, http://veilletourisme.ca/2005/02/08/les-modeles-%C2%ABopaques%C2%BB-revolutionneront-ils-la-distribution/, 2005.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] G. Foster, G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "The right tool for the job: Switching data centre management strategies at runtime," in *IFIP*. IEEE, 2013.

[5] L. Tomás, B. Caminero, and C. Carrion, "Improving grid resource usage: Metrics for measuring fragmentation," in *IEEE/ACM CCGRID*, 2012.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[7] D. Pandit, S. Chattopadhyay, M. Chattopadhyay, and N. Chaki, "Resource allocation in cloud using simulated annealing," in *IEEE AIMoC*, 2014.

[8] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "A distributed approach to dynamic VM management," in *IEEE CNSM*, 2013.

[9] S. Chen, J. Wu, and Z. Lu, "A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness," in *IEEE CIT*, 2012.

[10] R. Hasan, M. M. Hossain, and R. Khan, "Aura: an IoT based cloud infrastructure for localized mobile computation outsourcing," in *IEEE MobileCloud*, 2015.

[11] S. Al Noor, R. Hasan, and M. M. Haque, "Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *IEEE Cloud*, 2014.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *ACM MCC*, 2012.

[13] A. Bellisario, "Why serve a niche market?" Online at http://www.entrepreneur.com/article/203958, 2009.

[14] K. Popp and R. Meyer, *Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry*. BoD–Books on Demand, 2010.

[15] T. Isckia, "Amazon's evolving ecosystem: A cyber-bookstore and application service provider," *Canadian Journal of Administrative Sciences / Revue Canadienne des Sciences de l'Administration*, vol. 26, no. 4, pp. 332–343, 2009.

[16] D. Shapiro and X. Shi, "Market Segmentation: The role of opaque travel agencies," *Journal of Economics & Management Strategy*, vol. 17, no. 4, pp. 803–837, 2008.

[17] L. Illge and L. Preuss, "Strategies for sustainable cotton: Comparing niche with mainstream markets," *Corporate Social Responsibility and Environmental Management*, vol. 19, no. 2, pp. 102–113, 2012.

[18] J. Wilner, "Exclusive interview: Priceline.com," http://www.ecommercetimes.com/story/2267.html, 2000.

[19] S. A. Noor, R. Khan, M. M. Hossain, and R. Hasan, "Litigo: A cost-driven model for opaque cloud services," in *IEEE CLOUD*, 2016.

[20] Y. Jiang, "Price discrimination with opaque products," *J Revenue Pricing Manag*, vol. 6, no. 2, pp. 118–134, 2007.

[21] M. McGrath, "Priceline beats on profit and revenue, boosted by 38.8% increase in bookings," Forbes.com, http://j.mp/271JfqC, 2014.

[22] R. Khan, S. Zawoad, M. Haque, and R. Hasan, "Otit: Towards secure provenance modeling for location proofs," in *ACM ASIACCS*, 2014.

[23] S. Zawoad, A. Dutta, and R. Hasan, "Towards building forensics enabled cloud through secure logging-as-a-service," *IEEE TDSC*, vol. 13, no. 2, 2015.

[24] B. Leiba, "Oauth web authorization protocol," *IEEE Internet Computing*, vol. 16, no. 1, p. 74, 2012.

[25] L. Osmani, S. Toor, M. Komu, M. Kortelainen, T. Linden, J. White, R. Khan, P. Eerola, and S. Tarkoma, "Secure cloud connectivity for scientific applications," *IEEE TSC*, 2015.

[26] M. Komu, M. Sethi, R. Mallavarapu, H. Oirola, R. Khan, and S. Tarkoma, "Secure networking for virtual machines in the cloud," in *IEEE CLUSTER WORKSHOPS*, 2012.

[27] B. Costa, M. Matos, and A. Sousa, "Capi: cloud computing api," *Inforum, Simpósio de Informática*, 2009.

[28] B. S. Lee, S. Yan, D. Ma, and G. Zhao, "Aggregating IAAS service," in *SRII*, 2011.

[29] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[30] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson, "Toward cloud-agnostic middlewares," in *ACM SIGPLAN OOPSLA*, 2009.

[31] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *IEEE ITNG*, 2010.

[32] I. Houidi, M. Mechtri, W. Louati, and D. Zeghlache, "Cloud service delivery across multiple cloud platforms," in *IEEE SCC*, 2011.

[33] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," in *IEEE CLOUD*, 2012.

[34] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358–367, 2012.

[35] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, 2012.