Industrial Engineering Journal

ISSN: 0970-2555

Volume : 53, Issue 2, February : 2024

MULTI-NETWORK TRUST-AWARE CLOUD SERVICE SELECTION MECHANISM IN PEER-ASSISTED ENVIRONMENTS

K.Krishna1 , Bodduna Srinivas2 , P.Sandeep3, Shivaraoyannam4,V.Anil Kumar5
1,2,3,4,5Assistant Professor, Department of Computer Science and Engineering, MallaReddy College of Engineering, Maisammaguda, Secunderabad-500 100,India.
Email: 1krishnak2k13@gmail.com

ABSTRACT:
Within the vast landscape of cloud service providers, each offering a multitude of virtual machines (VMs) with diverse configurations, selecting the appropriate VMs becomes a crucial concern for companies. A well-informed service selection not only enhances productivity and efficiency but also reduces costs. However, due to the modular nature of requests, conflicts between requirements, and the impact of network parameters, a systematic approach becomes imperative for effective service selection. To address this challenge, we introduce a groundbreaking framework, named "PolyCloud Assisted (PCA)", designed specifically for the hybrid environment encompassing peer-assisted, public, and private clouds. PCA tackles the service selection problem by identifying conflicts between requests and enterprise policies, identifying suitable services based on requirements, and minimizing VM rental and end-to-end network expenses. By leveraging resources from multiple clouds and optimizing overall costs, PCA employs a combination of set theory, B+ tree, and greedy algorithms to achieve its objectives. Our simulation results exhibit that PCA can reduce cloud-related costs by up to 30 percent and deliver responses at least seven times faster compared to recent studies.

Keywords: Virtual machines, Play cloud assistant, greedy algorithms.

INTRODUCTION

Enterprises leverage computing resources, storage space, and a wide array of applications to accomplish their business objectives. However, concerns surrounding the time and cost associated with upgrading, updating, and maintaining these resources persist. The advent of cloud computing has successfully alleviated many of these challenges. According to the National Institute of Standards and Technology (NIST), cloud computing is defined as a paradigm that enables access to shared resource pools [1]. Cloud providers offer a diverse range of resources in the form of services, categorized into three main levels: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). By renting these services, enterprises can significantly reduce their costs related to infrastructure maintenance and upgrades, allowing them to concentrate on their core objectives. Typically, service payments are usage-based, although some providers offer discounted or reserved services with monthly or yearly costs.

When an enterprise requires resources, it formulates requests consisting of multiple modules. Each module represents a specific service with particular specifications. Consequently, the challenge lies in finding the appropriate service for each requested module, bearing in mind that these modules may also need to communicate with one another. This introduces network traffic costs that further impact cloud-related expenses. Consequently, identifying suitable services for each module is a nontrivial task. Surveys on cloud computing emphasize the importance of service selection and the definition of qualified providers' services from an enterprise's perspective. In a Right Scale survey conducted in 2018 with 997 IT professionals, it was found that 96% of the surveyed companies were utilizing cloud services [2]. This highlights the popularity and significance of cloud computing. The survey also reveals that optimizing cloud-related costs is the top challenge faced by enterprises. Additionally, a Microsoft survey conducted in 2015 with 1979 IT specialists identified the most crucial factors driving enterprises towards cloud computing, including work efficiency, operational cost reduction, and

hardware maintenance [3]. Based on these surveys, the service selection process aims to achieve two important goals:
- Meeting the requirements of requested modules to enhance efficiency.
- Reducing expenditures related to service rental and network communications.

The appropriate service for each requested module should align with these goals. The Right Scale survey [2] indicates that 81% of polled enterprises adopt a multi-cloud strategy with an average of five cloud providers. This suggests that providers offer a variety of services, as a single provider may not be able to satisfy all user requests. Moreover, 63% of enterprises have requirements involving more than 1000 modules. In cases where module specifications vary, different providers may host the modules, necessitating inter-module communication. Such situations are likely to increase communication costs, resulting in the total cost comprising both rental and communication expenses. The survey also reveals that around 35% of enterprises have wasted cloud spending, with only a few of them implementing automated policies to optimize cloud costs. Consequently, there is a need for a system that assists enterprises in service selection.

This paper presents a framework designed to identify appropriate services for an enterprise's requested modules. The framework aims to satisfy both aforementioned goals, namely, fulfilling requirements and reducing long-term cloud-related expenses. It takes into account the modularity of requests, rental and communication costs, VM specifications, QoS parameters (such as response time and availability), and the abundance of providers. The vast number of providers offering services with diverse specifications contributes to the complexity of service selection. A study by T. H. Noor et al. in 2013 identified approximately 6686 distinct cloud services varying in terms of cost, location, and VM specifications [4].

The proposed framework addresses the limitations of previous studies concerning scalability in handling a large number of providers and service selection response time. Furthermore, most existing studies fail to consider the modularity of requests and the comprehensive inter-service

RELATED WORK

The previous studies can be classified by different factors, including their methods and algorithms, considered parameters, assumptions, and objective function. The employed algorithms and methods have a significant effect on the execution time of the selection process. Thus, we have classified the previous studies based on their methods. The four considered groups are multiple-criteria decision making (MCDM), greedy and dynamic algorithms, logical descriptions, and other algorithms. PCA framework uses both the logical description and greedy algorithms. The following paragraph describes each group in detail. MCDM: It is proper for solving decision problems with a small number of parameters that conflict with each other. For example, cost parameter always affects the availability and computational power of services. It is less probable to rent a powerful computational service with a small charge. MCDM assigns a weight to each parameter according to its priority, and there are more than 30 methods for it. S. K. Garg et al. [5] used analytic hierarchy process (AHP) in a framework, which ranks cloud services. Even though this research considers many QoS parameters, the framework neglects network cost, communications between services, and modularity of requests. A. Taha et al. [6] used AHP and considered modularity, but did not consider communications parameters. Reference [7] uses TOPSIS. Its method avoids the rank reversal problem of AHP and does not consider network cost and modularity of requests. R. R. Kumar et al. [8] used both of the AHP and TOPSIS methods for criteria weights and service ranking, respectively. Even though their solution considers fuzzy environments, it lacks the service traffic parameter. S. Silas et al. [9] introduced a system based on elimination and choice expressing reality (ELECTRE). Even though they considered network delay, they did not contemplate the cost and modularity of re-quests. U. S¸ener et al. [10] proposed a decision support system for service selection that uses multiple MCDM methods. L. Zhao et al. [11] described a service selector system for service-oriented environments. Modules connections have not been mentioned in [10] and [11]. W. Fan et al. [12] proposed a trust model to find more trusted services when parameter values have uncertainty. Drawbacks of [9] and [11] exist in [12]. Reference [13] uses MCDM and fuzzy logic for considering qualitative parameters. Its scalability is not good enough for the large number of services. Reference [14] uses an approach similar to [13], but it is for cloud database selection and does not consider the cost.

Linear programming, Greedy, and Dynamic algorithms: Knapsack algorithm and K-means clustering are two of the used methods in this category. Also, some studies use heuristic algorithms. K.-C. Huang et al. [15] tried to minimize the cost-of-service composition in dynamic cloud environment with respect to constraints of service-level agreement (SLA). They used nonlinear programming, did not consider the network costs, and concentrated only on service response time satisfaction. H. Qian et al. [16] proposed a greedy algorithm, which considers VM rent, distribution of providers, and connections between services. Customers are software providers, but values of parameters are not end-to-end. Also, they did not mention the method of service finding and effect of network cost on the selection. A. C. Zhou et al. [17] used A∗ method and spot instances to reduce the monetary costs of workflows in dynamic cloud environment. Reference [18] describes a dynamic system, which updates the selection by the change of VM rent or user requirements. References [17] and [18] do not consider the cost of data transfer. J. Yang et al. [19] used Markov Decision Process (MDP) and a greedy algorithm to intro-duce a way of service combination in the cloud. L. Sun [20] predicted future QoS of the cloud services by employing MDP. L. Sun showed that the selected services satisfy users requirements and economic constraints. References [19] and [20] lack network cost parameter. S. K. Garg et al. [21] proposed a cloud service selector system for SaaS providers. Their system uses a heuristic algorithm that its goals are the reduction of software failures and rent of VMs. Even though their system considers the cost of bandwidth usage, it assumes that every module must be mapped onto a separated cloud. This assumption is not possible for large modular requests or situations that security and reduction of cost make the modules to be mapped onto one cloud. G. F. Anastasi et al. [22] proposed a cloud broker, which finds in frastructure services by employing genetic algorithm. Even though their proposed broker considers the network costs and QoS parameters, the applications are workflows. Logical descriptions and artificial intelligence: Dominant methods of this category are first-order logic and machine learning algorithms. Reference [23] is one of the studies that proposes a framework for enterprises. It describes the requests by first-order logic and finds the conflict between them by formal verification and CO. The framework uses heuristic algorithms to find the proper services. This frame-work does not contemplate network parameters. Y. Liu et al. [24] used rough set theory to find the proper service and did not consider modularity of requests. N. Somu et al. [25] used rough set-based hypergraph technique for the selection of trustworthy services. Drawbacks of [6] exist in [25]. C. Esposito et al. [26] selected cloud-based databases with the help of game theory and theory of evidence. This research lacks network cost parameter and requests are not modular.

PCA FRAMEWORK

This section describes assumptions, input, and output of the cloud service selection problem. Next, our proposed frame-work (PCA) and its components are explained in detail.

3.1 Assumptions

Users of an enterprise have requests, which each one con-sists of a set of connected modules. Each module has specific requirements, which are determined by a user. For

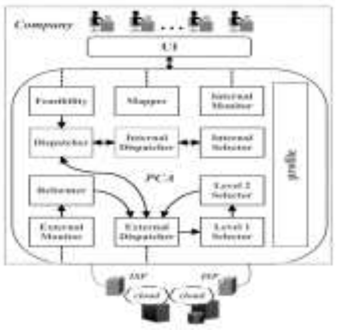

Fig. 1. PCA framework and its components in enterprise.

each module, the enterprise needs a proper service in a cloud. PCA framework helps in this selection process. Fig. 1 shows components of the PCA framework. It receives users' requests as input and finds a suitable service for each requested module as output. Each request is processed in-dividually. These suitable services can be in public, private, and peer-assisted clouds. When an enterprise uses external clouds, it usually does not use internal resources efficiently [34]. These centralized users' resources can add up and create a peer-assisted cloud, which can satisfy some requests and reduces the cloud-related monetary costs.
We assume that enterprises may use multihoming. This means that the enterprise may rent the Internet service from two or more ISPs, which brings more availability. Because if one link goes down, the enterprise still can keep up its connectivity through another ISP. Furthermore, each link may have a different delay and bandwidth cost, and PCA tries to select the proper link for each request. Each peer is available with a probability less than one. Like [33], we assume that after VM allocation in internal resources, if a peer is available and receives a request, this peer answers the request. This means that in situations like data loading period the peer cannot turn off the system.

3.2 PCA Architecture

For achieving a better selection, PCA considers VMs specifications, network delay, SLA of providers, VM location, rent of VM, network cost of ISPs, and inter-data center traffic cost. This framework deals with network parameters and request modularity as important factors in contrast to works like [5], [9], [16]. In studies like [13] and [16], the increase in the number of cloud service providers can make time of the service selection process to take minutes. Thus, PCA makes the scalability more reasonable by using B+ tree and indexing.
This is a component-based framework, which each com-ponent makes a decision with due regard to a portion of the requested specifications. This partitioning of responsibilities has two benefits. First, if the processing load of the PCA is high in an organization, components can be implemented distributively. Second, it makes the framework usable for future works. Researchers can focus on specific parts of the service selection algorithms without any concern about other responsibilities.
The framework needs two prerequisites to be deployed in an enterprise. First, peers of an enterprise must install a daemon software to enable peer-assisted cloud. Second, the enterprise must use software-defined network (SDN). SDN extracts control plane from the switches [35]. Thus, the admin or system can configure switches remotely. This control plane can be placed in PCA and brings transparency to service selection. Also, it needs a system with at least 8 GB free RAM.
The framework works as follows: First, a user defines requirements of requested modules via a user interface (UI). These requirements are converted to JavaScript Object Notation (JSON) format and are sent to PCA as a request. Feasibility component receives the request. As the component name implies, it checks the request feasibility. If it finds any conflict, it informs the user to reform specifications of the request. This interactive communication keeps up until the component accepts the request, or user

discards the request. Feasibility checking happens in two phases: first, by the enterprise's policies, and then by previously answered requests. Feasibility algorithm is described in subsection 5.1.
By acceptance of the request, Feasibility component passes it to Dispatcher, which is a mediator between Internal and External Dispatchers and saves the selection result in the repository. Because of the mediating role of the Dis-patcher, organizations can configure it to block the Internal Dispatcher, which means they do not want to use the peer-assisted cloud. Dispatcher component sends the request to External Dispatcher, which has the ability to communicate with public and private cloud providers. Also, different implementations of External Dispatcher can get cloud providers' SLAs and their VMs specifications by web crawling or communicating with some cloud brokers.
External Dispatcher sends the request to Level 1 Selector for further processing. For each module in the request, Level 1 Selector finds all the VMs that can satisfy the specifications. Level 1 Selector does not consider the network parameters and cost. Also, the selected set of VMs for each module must not violate policies of the enterprise. Its algorithm is explained in subsection 5.2. Level 1 Selector sends the request and its resultant sets to Level 2 Selector. Level 2 Selector ranks VMs based on network delay, bandwidth, performance, and their total cost. After that, it chooses the proper VM for each module and informs the selection to External Dispatcher. Level 2 Selector algorithm is described in subsection 5.3. If no VM can satisfy the module, an error is passed to the user. External Dispatcher informs Dispatcher about the selection and total cost.
This time, Dispatcher sends the request to Internal Dis-patcher, which tries to reduce the total cost. Before any processing, it sends the request to Internal Selector, which determines which modules can be satisfied by internal re-sources through some service replications. The replication happens in multiple resources for reaching to a specific availability. The current implementation chooses the internal replications randomly and checks their ability to satisfy requirements of a requested module. All the specified mod-ules in this step have the chance of mapping onto internal resources, but they may increase the total cost. Thus, Internal Dispatcher determines which ones are better to be mapped onto internal resources to culminate in the reduction of total cost. Its algorithm is detailed in subsection 5.4. After finding the proper VMs and their locations, dispatchers begin to send allocation requests to clouds.

PROPOSED ALGORITHM

PCA FRAMEWORK ALGORITHM:

Our proposed framework, PCA, can process modular requests and considers rent and communication costs of clouds. It addresses scalability and response time of the selection algorithm. Furthermore, it uses free resources of the enterprise's peers, which make a peer-assisted cloud. Use peer-assisted cloud without considering public and private clouds. In contrast to these previous studies, our proposed framework unites different types of clouds. The next section describes the PCA framework

Next, our proposed frame-work (PCA) and its components are explained in detail. Users of an enterprise have requests, which each one con-sists of a set of connected modules. Each module has specific requirements, which are determined by a user. For each module, the enterprise needs a proper service in a cloud. PCA framework helps in this selection process. Fig. 1 shows components of the PCA framework. It receives users' requests as input and finds a suitable service for each requested module as output. Each request is processed in-dividually. These suitable services can be in public, private, and peer-assisted clouds. When an enterprise uses external clouds, it usually does not use internal resources efficiently [34]. These centralized users' resources can add up and create a peer-assisted cloud, which can satisfy some requests and reduces the cloud-related monetary costs.

Feasibility component has two responsibilities of receiving the new policies and processing the requests to find the conflicts between specifications and policies. Policy receiver part is a grammar parser. The format of input policies in the current version of the PCA is as follows:

(Group group ID/ Service service Name/ All Groups/ All Services)(permit/ deny)[bandwidth value][locations values...][services values... location values...][providers values... location values...][availability value]
Feasibility component can receive and parse policies for

```
Input:  A request and prohibitive and accepted policies
Output:  List of violated requrements and their domain
procedure VALIDATE($Req, PPolicy, APolicy$)
    $error \leftarrow \{\}$
    $bw \leftarrow APolicy[bandwidth, Req.group]$
    if $bw \neq 0$ and $Req.bandwith > bw$ then
        $error \leftarrow error \cup (Req.bandwith - bw)$
    end if
    /*Do the same for availability*/
    for all Module m in Req do
        for all Parameter p in m do
            $\Lambda \leftarrow APolicy[Req.group, p] \cup APolicy[m, p]$
            $\Gamma \leftarrow PPolicy[Req.group, p] \cup PPolicy[m, p]$
            if $p \not\subseteq \Lambda$ then
                $error \leftarrow error \cup (p - \Lambda)$
            end if
            if $p \cap \Gamma \neq \emptyset$ then
                $error \leftarrow error \cup (p \cap \Gamma)$
            end if
        end for
    end for
    /*Do the same for services and their communica-
tional requirements*/
    return $error$
end procedure
```

CONCLUSION AND FUTURE ENHANCEMENT

In summary, we have proposed PCA framework, which consists of five main components of Feasibility, Level 1 and 2 Selectors, Internal Dispatcher, and Internal Selector. As simulation has shown, PCA is reasonably scalable and responds in less than a few seconds. It selects the proper providers at least seven times faster than the recent studies. Also, by the help of Internal Selector and Dispatcher, which consider traffic costs and unused internal resources, PCA has succeeded in reducing the total cost significantly. By the favor of Feasibility, PCA is suitable for enterprises that want to exert some policies on their users' requests. It can process up to 1480 different connections in less than one second.

REFERENCES

[1] P. Mell and T. Grance. (2011, Sep.) The NIST definition of cloud computing. NIST. [Online]. Available: http://nvlpubs.nist.gov/ nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
[2] K. Weins. (2018, Feb.) Cloud computing trends: 2018 state of the cloud survey. Right Scale. [Online]. Available: https://www.rightscale.com/blog/cloud-industry-insights/ cloud-computing-trends-2018-state-cloud-survey
[3] (2015) Cloud computing survey the results. Microsoft. [Online]. Available: http://download.microsoft.com/documents/uk/technet/cloud-power/ITPro survey v5.pdf
[4] T. H. Noor, Q. Z. Sheng, A. H. Ngu, A. Alfazi, and J. Law, "Cloud armor: a platform for credibility-based trust management of cloud services," in Proc. 22nd ACM Int. Conf. Inform. & Knowledge Manage. (CIKM). San Francisco, California, USA: ACM, 2013, pp. 2509– 2512.
[5] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," Future Generation Compute. Syst., vol. 29, no. 4, pp. 1012–1023, Jun. 2013.
[6] A. Taha, S. Manzoor, and N. Suri, "SLA-based service selection for multi-cloud environments," in Proc. IEEE Int. Conf. Edge Computing (EDGE). Honolulu, Hawaii, USA: IEEE, 2017, pp. 65–72.

[7] R. R. Kumar and C. Kumar, "A multicriteria decision-making method for cloud service selection and ranking," in Proc. Advances in Compute. and Computational Sci. Singapore, Republic of Singapore: Springer, 2018, pp. 139–147.
[8] R. R. Kumar, S. Mishra, and C. Kumar, "Prioritizing the solution of cloud service selection using integrated MCDM methods under fuzzy environment," The J. of Supercomputing, vol. 73, no. 11, pp. 4652–4682, Nov. 2017.
[9] S. Silas, E. B. Rajsingh, and K. Ezra, "Efficient service selection middleware using ELECTRE methodology for cloud environments," Inform. Technol. J., vol. 11, no. 7, pp. 868–875, Jul. 2012.
[10] U. S¸ener, E. G¨okalp, and P. E. Eren, "Clouds: A decision support system for cloud service selection," in Proc. Int. Conf. Econ. of Grids, Clouds, Syst., and Services. Biarritz, France: Springer, 2017, pp. 249–261.
[11] L. Zhao, Y. Ren, M. Li, and K. Sakurai, "Flexible service selection with user-specific QoS support in service-oriented architecture," J. of Network and Compute. Applicant., vol. 35, no. 3, pp. 962–973, May 2012.
[12] W.-J. Fan, S.-L. Yang, H. Perros, and J. Pei, "A multi-dimensional trust-aware cloud service selection mechanism based on evidential reasoning approach," Int. J. of Automation and Computing, vol. 12, no. 2, pp. 208–219, Apr. 2015.
[13] I. Patiniotakis, Y. Verginadis, and G. Mentzas, "PuLSaR: preference-based cloud service selection for cloud service brokers," J. of Internet Services and Applicant., vol. 6, no. 1, pp. 6–26, Aug. 2015.
[14] L. Sun, J. Ma, Y. Zhang, H. Dong, and F. Khadeer Hussain, "Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection," Future Generation Compute. Syst., vol. 57, no. 1, pp. 42– 55, Apr. 2016.
[15] K.-C. Huang, M.-J. Tsai, S.-J. Lu, and C.-H. Hung, "SLA constrained service selection for minimizing costs of providing composite cloud services under stochastic runtime performance," SpringerPlus, vol. 5, no. 1, pp. 294–312, Mar. 2016.
[16] H. Qian, H. Zu, C. Cao, and Q. Wang, "CSS: Facilitate the cloud service selection in IaaS platforms," in Proc. Int. Conf. Collaboration Technologies and Syst. (CTS). San Diego, California, USA: IEEE, 2013, pp. 347–354.
[17] A. C. Zhou, B. He, and C. Liu, "Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds," IEEE Trans. On Cloud Compute., vol. 4, no. 1, pp. 34–48, Jan. 2016.
[18] J. O. Gutierrez-Garcia and K. M. Sim, "Agent-based cloud service composition," Appl. intelligence, vol. 38, no. 3, pp. 436–464, Apr. 2013.
[19] J. Yang, W. Lin, and W. Dou, "An adaptive service selection method for cross-cloud service composition," Concurrency and Computation: Practice and Experience, vol. 25, no. 18, pp. 2435–2454, Jul. 2013.
[20] L. Sun, "An influence diagram based cloud service selection approach in dynamic cloud marketplaces," Cluster Computing, pp. 1–10, Dec. 2017.
[21] S. K. Garg, L. Gao, and J. Montgomery, "Clouds selection for network appliances based on trust credibility," in Proc. Int. Telecommun. Networks and Applicant. Conf. (ITNAC). Sydney, New South Wales, Australia: IEEE, 2015, pp. 302–307.
[22] G. F. Anastasi, E. Carline, M. Coppola, and P. Dazzi, "QoS-aware genetic cloud brokering," Future Generation Compute. Syst., vol. 75, no. 1, pp. 1–13, Oct. 2017.
[23] C. Chen, S. Yan, G. Zhao, B. S. Lee, and S. Singhal, "A systematic framework enabling automatic conflict detection and explanation in cloud service selection for enterprises," in Proc. IEEE 5th Int. Conf. Cloud Computing (CLOUD). Honolulu, Hawaii, USA: IEEE, 2012, pp. 883–890.
[24] Y. Liu, M. Esseghir, and L. M. Boulahia, "Cloud service selection based on rough set theory," in Proc. Int. Conf. and Workshop Network of the Future (NOF). Paris, France: IEEE, 2014, pp. 1–6.

[25] N. Somu, K. Kirthivasan, and V. S. Shankar Sriram, "A rough setbased hypergraph trust measure parameter selection technique for cloud service selection," The J. of Supercomputing, vol. 73, no. 10, pp. 4535–4559, Oct. 2017.
[26] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," IEEE Trans. Compute., vol. 65, no. 8, pp. 2348– 2362, Aug. 2016.
[27] S. Soltani, P. Martin, and K. Elgazzar, "QuARAM recommender: case-based reasoning for iaas service selection," in Proc. Int. Conf. Cloud and Autonomic Computing (ICCAC). London, United Kingdom: IEEE, 2014, pp. 220–226.
[28] C. Quinton, D. Romero, and L. Duchien, "Automated selection and configuration of cloud environments using software product lines principles," in Proc. IEEE 7th Int. Conf. Cloud Computing (CLOUD). Anchorage, Alaska, USA: IEEE, 2014, pp. 144–151.