Achieving Application Portability In Cloud Computing Environment

Brijesh Pandey, S. S. Soam, D. K. Yadav

Abstract: Application portability in a cloud computing condition is characterized to be the exchange of utilization or its segments starting with one cloud administration then onto the next cloud administration of its specialized comparability. The application should be such transferred that it does not make any significant changes in the application code although recompilation or relinking may be allowed. Application portability can be performed by the customer only in case of IaaS and PaaS services because for SaaS the application code relies on the cloud service provider itself. The application artifacts and their dependencies play a major role in the target environment. Containers and their related infrastructure hold valuable importance for application portability. In this paper, we will elaborate on all the issues, challenges, scenarios, and approaches for application portability that rose to date. We will study all the facets of data and application portability such as its instruction, syntax, metadata, and policies. The evolution of automation tools and technologies has made the portability of applications easier as the manual processes were inefficient and error-prone.

Keywords: Virtualization, Containers, Orchestration, OCCI.

I. INTRODUCTION

Cloud computation is a comparatively fresh model, in which computation is suggested as an efficacy, along with having the prospective for transforming a huge portion of the Information Technology business [1]. As stated by the United States NIST (National Institute of Standards and Technology), cloud computation is a paradigm for allowing omnipresent, expedient, on-request access of network to a common consortium of configurable computation reserves (for example, applications, networks, servers, services, and storage) which may be quickly provisioned as well as published with nominal attempt by the administration or facility supplier communication [2]. Because of the reason that cloud computation denotes diverse reserves provision, varying from hardware (computation power, storage) to software (applications, platforms for software development), additional decomposition of this has been done to facilitate numerous prototypes. As stated by NIST 3 models of service exist [3] which has been presented in Figure1. The lowest prototype of service is the IaaS (Infrastructure as a Service). In Infrastructure as a Service, simple computation resources such as storing, processing and networking have been supported.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author

Dr. Sudhir Singh Soam*, Ph.D, Dr. APJ AKTU Lucknow, Uttar Pradesh, India.
Dr. D. K. Yadav, PhD, IIT Bombay, Maharashtra, India.

The central prototype of service is the PaaS (Platform as a Service).

In Platform as a Service, the customers are allowed admission to implements of APIs, programming languages and software for developing as well as deploying their programs. The highest-level model of service is the SaaS (Software as a Service). In this archetype of service, ready to be utilized and comprehensive software programs are provisioned.

Customers of Cloud, either acting on PaaS, IaaS, or SaaS prototype may gain a lot of advantage from this model of computation. With the use of cloud computation, customers gain universal admission to their information as well as programs from around the globe. The sole prerequisite is internet access. Customers do not need to concern themselves regarding connecting, maintenance and updating the hardware and/or software used by them. This problem has presently been transferred completely to the suppliers of the cloud. The additional advantage is gained from the features of cloud computation for quickly provisioning as well as releasing resources. Consumers may have the reserves required by them, accessible at every specified pay and time simply for that which is used by them. Thus, it is not required for them to purchase open hardware or software which will never be used by them. Though cloud computation is developing fast along with offering various advantages for customers, quite a few issues still exist that need to be taken care of before it turns out to be extensively espoused by administrations and businesses. Inter alia, significant matters are interoperability and portability through platforms of the cloud. As Figure 2 depicts, the primary denotes the facility of transporting a facility from certain suppliers of the cloud to a different one, whereas the next one denotes the facility of 2 services from dissimilar clouds for exchanging data.

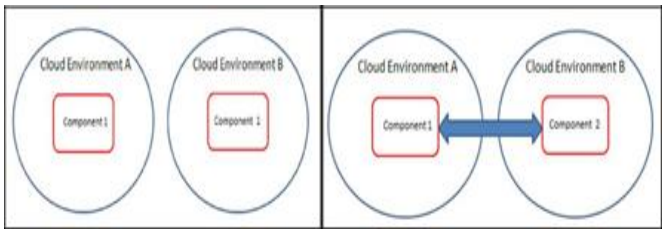


Figure 1: Left: a constituent is transferred from a certain environment to a different one (portability). Right: two constituents of dissimilar environments of the cloud are cooperating (inter-operability).

As will be offered in the following section, interoperability and portability influence all 3 prototypes of service (IaaS/PaaS/SaaS). In this text, our core concern is towards the feature of portability inside the PaaS setting, viz. the viability of a cloud program to be moved through diverse podiums. Chief suppliers of a cloud platform like Microsoft Azure, Google App Engine, etc are presently making use of exclusive technologies and systems that avert the portability of cloud programs. This matter has a significant part to play in the broader implementation of cloud computing as it can deter customers from operating services of the cloud platform.
The objective of this study is introducing interoperability as well as portability notions, discussing their connection, along with presenting little superior kind of methods to address the concern of portability in PaaS context. The offered methods fluctuate from describing common criterions, generating APIs to theoretical/wrap exclusive ones, towards creating platforms that are open source. The aim of our study is the exploration of the methods for improving the portability of the programs of the cloud, along with the primary purpose of our aim is understanding how platform-freedom may be improved by utilizing Ontologies) and MDE (Model Driven Engineering).
The remnant of the paper is systematized as under. In the following section, the phrases interoperability and portability are presented along with the explanation of why these are vital matters for the broader approval of cloud computation. Subsequently, we concentrate on the matter of portability in the milieu of the PaaS model. Superior level methods for allowing moveable applications of cloud, trailed by a summary of present discoveries in this subject were discussed by us. Lastly, in section 4 few upcoming study instructions have been presented by us.
Cloud portability is the capability of moving programs as well as data beginning with a particular cloud computation setting following that on top of the following, with unrelated troubling effect. Cloud portability allows the relocation of cloud facilities from a particular cloud supplier to a different one or among a reserved cloud and an open cloud. Customers or administrations would choose to be not limited into a self-contained dealer choice and could require to ease risk and increase their compliance via the ability of moving their programs as well as information among cloud facility suppliers since the computation capacities of theirs change and the commercial requirements alter since the requirement arise they could ask for the ease of moving between the diverse cloud positioning prototypes by feasibly shifting to open clouds along with among open, sequestered and amalgam clouds.
The key motives for cloud portability are as follows:

- Financial Causes: The consumers of the cloud would gain a guarantee regarding their benefits in their programming improvements.
- Technical Causes: Through a lifespan of a program, a time can arise when external resources are needed from an open cloud; therefore it needs to be redistributed from a reserved cloud.

A. Definition of Portability

Versatility is the capacity to run segments or frameworks independently in any condition where conditions in distributed computing elude to both programming and equipment situations (both physical and virtual). Movability and interoperability identify with the capacity to fabricate frameworks from re-usable segments that will cooperate “out of the box” [43]. A specific worry for distributed computing is cloud on-boarding – the organization or relocation of frameworks to a cloud administration or set of cloud administrations. The distributed computing compactness classes are:
Application Portability: Application convenience empowers the re-utilization of use segments crosswise over cloud PaaS administrations and customary processing stages. Application transportability demands a standard interface uncovered by the supporting stage. A specific application movability issue that emerges with distributed computing is convenience among improvement and operational conditions.

- Data Portability: Information transportability empowers the re-utilization of information parts crosswise over various applications. The composition of the information is frequently intended to fit into a specific type of use preparing, and an important change is required to make data that can be managed by a substitute thing.

- Platform Portability: There are two varieties of platform portability:

    - Re-utilization of stage segments crosswise over cloud IaaS administrations and non-cloud framework.

    - Machine picture transportability: Re-utilization of packs containing applications and information with their supporting stages.

The customary way to deal with stage versatility empowers applications movability since applications that utilization the standard working framework interface can comparably be recompiled and kept running on frameworks that have diverse equipment. It requires a standard program portrayal that can be conveyed in various IaaS use situations.

II. CLOUD PORTABILITY IN THE ENTERPRISE

Idiosyncratic attempts have been undertaken along with methods of dealing with supervision handling this trial on dissimilar status of the next Figure 1.

A. Nested Virtualization

The key elected position of this method is that it may be associated with every program presently which is at this point combined within a Virtual Machine and upcoming figure.

Pros:

Authorizes reliable relocation of present programs among clouds

Cons:

This prototype logically requires self-healing, auto-scaling, and contemporary observing proficiencies

There is debasement as well as implementation overhead concerns.

There is an increased dependency on external hypervisor dealers as the rudimentary hypervisor among clouds.

B. Containers

Different from VM which requires the knowledge of a hypervisor stratum, containers function as facilities inside the OS (operating system), thus consequently, this provides an essentially easier and extra suitable combined prototype and Figure.

Pros:

Have established as being uncomplicated for implementation along with rapidly attaining approval and impetus.

Cons:

This prototype furthermore distinctively requires self-healing, auto-scaling, and contemporary observing competences.

Altering present programs into containers is frequently not an inconsequential responsibility.

C. Compatibility of API

Application interface correspondingly and essentially displays the capability of giving an archetypal API which will function across every cloud. Essentially 2 categories of cloud APIs compatibility exist. The primary of those are structures of API compatibility like LibCloud or JCLouds and Figure.

Pros:

Supporting every cloud network structures of the designated cloud.

Cons:

An inhibited resemblance to the degree of the number of clouds maintained in this selection exists along with the ones which are already quite scarce

Mapping semantics of API may frequently be puzzling and intricate.

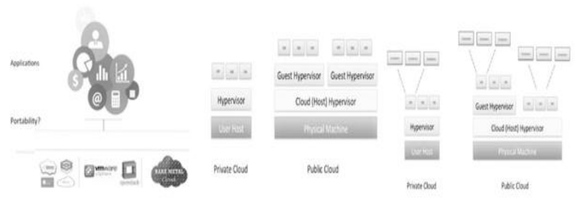


Figure2: cloud portability, Nested Virtualization, and Containers

D. PAAS

PaaS provides a stratum of replication which authorizes customers to conduct programs particularly to the cloud rather than being accessible to the central cloud substructure and Figure.

Pros:

Forthright

Promises a foreseeable life span, safety, scalability, great accessibility, as well as positioning experience through clouds

Cons:

PaaS needs are linked to a limited collection of stacks and programs.

The replication is limited in a way where it does not expose impelled cloud rudiments, for instance, novel networking competences.

E. Automation and Orchestration

Orchestration Arrangement is fundamentally a method of showing the physical process for monitoring as well as directing programs and mechanizes those. Automation may be linked to a general diversity of usages in its unaltered form and Figure.

Chief categories of orchestration platforms exist:

Pure-play orchestration

Container orchestration

Pros:

May be employed to a broad variety of programs comprising big data, legacy, networking, as well as further intricate programming stacks.

Comprises portability of failover, auto-scaling, as well as unceasing positioning procedures.

Exposed the complete capabilities as well as the power of all of the APIs and rudimentary cloud facilities.

Cons:

Needs modification for ensuring complete portability. Structuring may be an extensive plus sometimes intricate course.

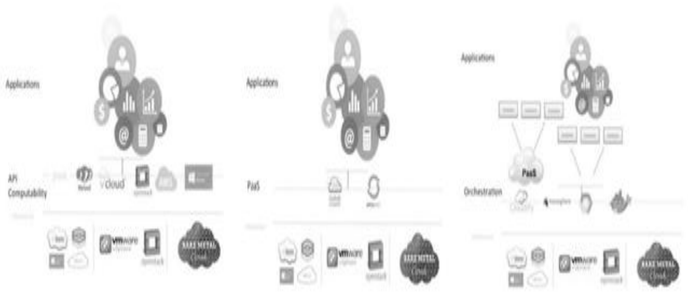


Figure 3: API Compatibility, PaaS and Automation, and Orchestration

III. PORTABILITY / INTEROPERABILITY CONCERNS ABOUT CLOUD COMPUTING

Before putting the focus on the feature of portability contextual to PaaS, it is essential to initiate the idea of portability contextual to cloud computation along with explaining its differences compared to interoperability.

A. Portability in cloud computation

Advantages, Levels and Need of application portability: By the term Cloud portability, it refers to the fact that confirms that information, services or applications function as per a common programmable communication in various kinds of cloud services. One of the major causes behind this is migration or shift of all or some existing services in between the clouds. To optimize expenses, for reducing price, technical or SLA issues, it could be used. The tricky or problematic circumstance appears if a cloud application of a user lies in between two or more than two administrative areas or providers at the same time. There are three methods to classify portability, which are: functional, data and service enhancement portability. The very first category or functional portability is meant by asserting the operability of an application through a vendor-agnostic way. Device agnostic aspect is allowing hardware and software parts to be functional in a different system and that doesn't need any particular features or adaptations. So, maximum systems are compatible in this regard. In the second category, data portability refers to a situation when a consumer or user can fetch application information from a particular provider as well as transfer it to a recurring application organized by a different provider. The last category, i.e. service enhancement metadata can be attached utilizing annotation. To fetch information regarding other data, metadata is used. Besides, control APIs allows infrastructural addition as well as reconfiguration based on traffic or other aspects. The need to use portability at PaaS is to lessen the burden of rewriting, provided the application is ported. For a user, data portability at PaaS helps to focus on reducing the load to rewrite during portability. It permits users to shift their private data from one service to another by selecting an appropriate service as per their requirement. One more benefit to port is that it ensures a user not to be in lock-in stage to one provider. Data portability is an interesting idea from the CSP as well as when users can shift information without any bar, faith is boosted between CSPs as well as the users. Moreover, further data are usable for the CSPs to advertise.

Layers of the PaaS: As per the definition application portability is the programming capacity to be applied on several data processing systems without altering the program to a divergent language incorporating no or very few changes. Portability describes the capability of using the mechanism on many software as well as hardware climates. PaaS is the fundamental idea and it is classified into three different layers. These are infrastructure, management, and platform. As PaaS doesn't reveal all its physical features, users get only app cells, dynos, gears, workers' units like specific ideas that are usable for some particular cases in PaaS, when the Central Processing Unit, as well as disk usages, are not important.

For IaaS, the utilization of CPU is the primary aspect, although RAM size, as well for instance counts, is also vital for PaaS. Besides, a global position to deploy the application is a major aspect as well. Based on latency aspects, applications can be regulated regionally. The platform is constituted by the host environment alongside two stack parts, namely, the runtime and service stack. In the case of SaaS, the runtime stack is made up of the programming language utilized to develop the application, like Ruby on Rails (quite popular). More application dependence and specialization are related to higher stack; so it amplifies the lock-in issue. Furthermore, the service stack is classified into include-one and native services. Native services are conducted by PaaS vendors which cover latency as well as performance services like data storage. Third parties provide additional services that cover: analytics-based data stores, search engines, and messaging services. Besides, the extensible approach is another aspect that involves the building of packs to develop their packs of services. The administration layers permit controlling of the user application that includes pushing, beginning or pausing an application. It also covers indigenous and additional services. Most significantly, resource utilization, as well as supervising features, is vital for billing purposes alongside the scalable criteria. These are performed by the administration layer. Although, the referred operations are utilized cumulative by every PaaS mostly, instructions, as well as processes, aren't standardized. These can differ from providers to providers to a great extent.

Application portability challenging areas: When it comes to the application portability, four genres are mentioned, which are: programming language and format, platform-explicit services, data storage, and platform-explicit configuration files. The particular program language utilizes for building an application is a vital aspect of cross-platform utilization. Every cloud dimension has some specific languages as well as a structure that they support. For example, Java is supported by Google App Engine but for Java class libraries it's not supportive. Open Shift supports them. Particular APIs are used to give services by the Cloud. It's a high-level operation which is usable by the provider and it doesn't require the from-scratch use. Conventional software applications use the configuration files for instructing the climate regarding the way of executing the application. Besides, there are Platform-specific configuration files. Google App Engine utilizes the "app engine – web XML" file. Adjusting the configuration file to every targeted cloud dimension impacts the portability which can be sorted out by standardizing and intermediating. The former defines the common rules or benchmarks for PaaS provisions. The adjustment of these criteria by every CSP will ensure the designing, deployment as well as management of developers' applications regardless of a cloud platform. There are two kinds of stores here, namely: data-based store, assigned for strongly ordered information, and file store which is similar to a hard disk on the cloud platform. Analytical methods utilized herein data sets as well as APIs to manipulate pictures are instances of these. Lessening of application growth time is achievable by using these platforms.

The developers can be included in operations from platform services through binding to recurrent platform APIs and not do programming for these operations since the beginning.

NIST denotes portability as the capability ―of potential customers of cloud computation to transport their information or programs through manifold cloud settings at reduced prices as well as nominal interference‖ and in particular to portability of system as ―the capability of migration to a completely-stationary VM (Virtual Machine) case or an image of machinery from certain supplier to a different supplier [4]. Significant features of portability may be gained from this description. Primarily, the transfer from a particular cloud to a different one needed to be attained at the lowermost probable price, endeavor, and interval. Also, portability denotes the capability of transferring every constituent of all of the 3 facility prototypes through platforms of the cloud.

As stated by CSA (Cloud Security Alliance) [5], diverse requirements of portability exist at the 3 diverse prototypes of cloud service. Within IaaS, the prerequisite is the ability to effortlessly move the VMs (Virtual Machines) along with information from a particular supplier to a different one. As an example, an institute or corporation functioning numerous VMs in a particular supplier of cloud substructure needs to have the capability of simply transferring them to another provider. Within PaaS, the prerequisite is the ability to deploy programs through diverse platforms. As an instance, if a programmer produces and uses an application that's situational on a particular cloud platform, it needs to be practicable for the program to be moved to a dissimilar setting, with a nominal group of alterations, if at all. Lastly, within SaaS, the prerequisite, during swapping from a particular software program to a different one, is having the ability to extract the information from the primary and encumber them to the secondary one. As an instance, in case an organization implements a CRM program supplied by a cloud vendor consequently deciding to shift to a dissimilar supply, it is vital that every consumer information may unswervingly be loaded as well as handled by the novel CRM program.

Customers, if they perform on an IaaS, PaaS or SaaS prototype require the ability to effortlessly alter among suppliers of cloud along with having the freedom of choosing the particular one which works in an improved way towards fulfilling their needs along with being cost-effective and of high quality. The customers capability of simply migrating from one cloud facility supplier to a different one is even further serious if a cloud supplier's action is suddenly dismissed. A realistic instance for illustrating this dispute is the Coghead event [6] – an internet program creating a platform hosting the development and hosting of information- powered programs. The organization had been able to entice few hundreds of programmers before its sudden announcement regarding its terms of operation, asking every consumer to move their information which they saved within their programs, yet not providing them the choice of porting the programs.

Guaranteeing portability throughout suppliers of cloud would eradicate the problem of seller lock-in [7] along with allowing customers to shift among sellers as per their requirement. Consecutively, this would cause the growth of customers' faith in cloud computation as well as services by open cloud.

B. Interoperability in cloud computation

Within the appendix of IEEE [8], the definition of interoperability is ―the aptitude of 2 or extra structures or constituents to swap data along with using the data which has been switched over. As stated by Petcu, numerous descriptions of cloud interoperability may be located in the works. As an instance, interoperability has been described as the knack of non-concrete the application variances from a particular cloud to a different one, the capacity of translating within the constructs maintained by diverse clouds, to compliantly operate programs natively or within the cloud or combined‖, or for using similar implements of managing, images of server, programs in numerous clouds.

Interoperability influences each of the 3 facility prototypes along with having an explicit necessity in every one of those. Interoperability, in IaaS, can denote the capacity of a customer to flawlessly operate resources of the substructure from diverse sellers via a public supervision API [9]. As an example, a customer might have the ability to perform the identical group of processes on Virtual Machines from dissimilar suppliers (example, start, stop or delete) minus producing a dissimilar customer for every provider. Within PaaS, programmers can require the usage of APIs, libraries, or tools sourced from various suppliers of PaaS for developing their programs. As an instance, a cloud program can be made of numerous cloud facilities sourced from various sellers of the cloud. Lastly, within SaaS, interoperability stays in the aptitude of diverse programs to swap data. As an instance, a corporation could wish to subcontract the emailing facility to Google or the HRM (Human Resource Management) facility to Salesforce. This suggests that the information arrangement within the structure of e-mail (example, address book, calendar) requires to be well-matched with the facility of HRM.

Without interoperability, a barrier is created between the usage as well as joining resolutions from diverse suppliers, yet permitting on-spot programs to cooperate and swap info with cloud facilities as well. Improving interoperability between cloud suppliers would allow 2 modes of associations. Primarily, facilities from numerous cloud suppliers might be flawlessly united to offer the finest types of resolutions regarding the feature, worth, or characteristics. Furthermore, corporations would have the ability of ― pushing only a portion of their facilities to the cloud however maintaining the ones which are most critical on–location.

C. Association among interoperability and portability

In Section 3.1 and 3.2 portability and interoperability were explained as different or individual concepts. However, it is common for the phrase ―interoperability to be implemented to denote each of the two notions. As recorded by Petcu (2011) [11], writers occasionally describe interoperability as the capability of ―moving programs from a particular environment to a different one or functioning in numerous clouds, or the capability of ―moving facilities, procedures, amount of work, as well as information between clouds. Consistent with the description provided by us in section 2.1, this is a situation of portability. According to Dowell et al (2011), portability is an unusual type of trial of interoperability. Observed from this viewpoint, the unpredictable usage of the phrase interoperability for denoting each of the notions may be defensible [12].

As Petcu (2011) suggested [11], it is possibly valuable to consider according to horizontal and perpendicular interoperability. The Horizontal interoperability may be described as the capability of two facilities of the cloud of similar facility prototype (IaaS/PaaS/SaaS) for communicating with one another – a description uniform to our interoperability concept as conversed in section 3.2. Conversely, perpendicular interoperability may be described as the capability of a cloud facility to be implemented on a cloud facility of a subordinate facility prototype. As an instance, permitting a SaaS program to be arranged on numerous offerings of PaaS. This description is constant to our concept of portability as conferred in section 3.1, i.e. the transporting aptitude of a program through numerous cloud podiums.

IV. PORTABILITY IN CONTENTION WITH PAAS

According to section 3.1, there are portability prerequisites throughout each of the 3 facility replicas (SaaS/PaaS/IaaS). The range of this study, though, is the exploration of the subject of program portability at the PaaS level.

A. Cloud portability concerns at the level of platform

Platforms of Cloud assure the easing of and speeding up of the cycle of program formation by proposing a comprehensive group of implements to develop, deploy as well as maintain the program. There is currently a comprehensive selection of PaaS suppliers, like Google App Engine, Force.com and Microsoft Azure, that present a varied array of facilities for the development of the program, comprising analytics, information and file storage, queuing, messaging, managing of the amount of work, etc. [13]. The amalgamation of the huge amount of platform assistances along with the diversity of cloud facilities adds to the experiment portability, as it raises the complexity of relocating a program through cloud platforms.

The pre-mentioned declaration turns well-defined with the consideration that the facilities of cloud, concurring to every supplier, can implement diverse technologies along with being suggested to customers by numerous exclusive APIs. As an instance, Microsoft offers the SQL Azure database for storing the information, whereas Google App Engine offers, inter alia, the App Engine Data Store. Along with that, program portability can be delayed by the datum that particular cloud facilities suggested by a particular platform are unavailable in a different one. As an instance, the facility of mailing that Google App Engine proposed might not be suggested by a different supplier. Therefore, the diversity of present day's platform contributions adds meaningfully to the trial of program portability.

B. Overall methods for tackling cloud portability in PaaS

Few general methods, as well as policies, exist which might be espoused for tackling the subjects delineated in section 3.1, along with ultimately easing program portability through platforms.

An apparent tactic is the description of the customary group of criterions for PaaS assistance. The implementation of these principles by every cloud supplier would permit programmers to produce their programs autonomously, without particular platform settings following which they are deployed to the cloud platform chosen by them. This group of principles might comprise a consistent API for accessing the facility presented by the platform, regular setups for demonstrating file arrangements, typical information stocks, etc.

Calibration appears to be quite a competent method for achieving cloud portability. Nevertheless, for causes not essentially linked with technology, it is quite hard for every platform of cloud to ultimately approve of a collective group of criterions. Every big vendor of cloud uses exclusive APIs as well as file setups as a method of securing clienteles to their facilities. The endeavor obligatory for re-engineering a program for porting it to a different platform is dispiriting clienteles to transfer. Additionally, a group of collective criterions would avert platform suppliers from proposing the singular, platform-particular characteristics which permit sellers to distinguish from their contenders.

Intermediation is an alternative method in the direction of attaining portability among platforms which is, presenting a transitional stratum which de-couples development of the program from APIs of a particular platform and compatible setups. Within these situations, programmers generate their programs with the use of a transitional API that is skeptical to platforms along with being able to ― withhold or ―cover the exclusive APIs of specific sellers. The transitional stratum averts programmers from being connected to particular computing languages, setups of file or information stocks. As an instance, a program might be programmed in a language-autonomous style, and afterward, via prototypical alterations, be rendered into the specific encoding language compatible with a PaaS supplier (like, C#, Java, or Python), or the databank inquiry language specific to one platform (like, MySQL or Microsoft SQL).

For cases as such, regarding presenting a transitional level for de-coupling program creation from particular platforms, agreement of platforms sellers is not necessary. Nevertheless, the interesting bit of this is developing the rubrics of translation as well as the prototypical alterations among the transitional level as well as every platform seller particularly.

C. Present works concerning cloud portability at platform level

A few works exist that endeavor tending to the subject of portability crosswise over cloud stages, through the adoption of single or uniting 2 of the methods exhibited in section 3.2.

MOSAIC: MOSAIC is a structure that assures the easing of program portability across different platforms by offering a range of APIs which are seller-autonomous [12], [13]. During the time of designing, programmers are implementing those APIs for creating programs that comprise numerous cloud constituents, all of them executing a particular utility. A cloud constituent may, for instance, be a Java program. Currently, the program is not restricted by any particular platform. Thus, during runtime, the platform of mOSAIC crumbles the program into the numerous cloud constituents along with deploying all of them on the platform of cloud which offers the finest execution for the functionality of cloud constituents. The collection of the existing cloud facilities that are usable is computerized and implemented by the platform of mOSAIC. Thus, programmers may concentrate on creating their programs in a style that's platform- nonaligned, and afterward, they may resolve on the supplier for the cloud which they hope to install them. The mOSAIC API performs as a transitional stratum among the programmers as well as the authentic cloud platforms, plus programmers need not implement exclusive APIs of the objective cloud platforms. Therefore, mOSAIC might be categorized as a transitional method that attempts at decoupling program development from specific technologies of the platform.

A sample program, as defined in Petcu (2011), might be a facility for checking out for purchasing online items [11]. The program might be divided into 4 central actions: a) recovering consumer expense particulars along with a list of merchandise, b) estimating the entire sum of money to be credited from the customer, c) putting a charge on the credit card via communicating with the bank, d) storing the particulars of transaction. During the time of development, all of those functions are designated to a constituent of the cloud, plus during the run-time, every constituent may be arranged on a platform that can execute the operation in the best way.

OCCI (Open Cloud Computing Interface): It is a group of stipulations that permit the evolution of implements for the execution of everyday cloud responsibilities such as autonomic scaling, deployment, as well as observing through diverse cloud facilities suppliers. It presents an API that is sustained by various stacks of cloud computation, like Open Nebula, Open Eucalyptus, and Open Stack. Thus, OCCI might be categorized as a calibration method. It needs to be stated that Open Cloud Computing Interface began in the form of an API to manage substructure of the cloud. Though, according to the statement in its certified online site, the OCCI will ultimately aid additional prototypes as well, other than IaaS, viz. SaaS and PaaS.

PSIF (PaaS Semantic Interoperability Framework). Loutas et al (2011) suggested the PaaS Semantic Interoperability Framework, which purposes at forming semantic interoperability clashes which can ensue while relocation or positioning of a program on a cloud platform is taking place [14]. The structure is organized in line with 3 scopes, (i) the diverse structural units in the setting of PaaS, (ii) the semantics type in a PaaS unit's description viz. serviceable, non-serviceable as well as performance semantics, (iii) and the stratum where clashes of semantic take place, that is, the stratum of the data prototype as well as the stratum of information. Semantic clashes are recognized and categorized in line with those 3 scopes.

Two samples of the structure's set-up were delivered by Loutas et al (2011) [14]. In the primary instance, a program is moved from a particular platform to a different one. A clash takes place once programs make an effort at connecting to a databank, as the 2 platforms function using different operations calls (for example ―connect db vs. ―insert db). This clash of semantic takes place because of the management interface descriptions of the 2 platforms and particularly because of how the semantics are structured. The clash is elevated at the informational stratum, as it is a result of the diverse designation of the identical operation. One more clash of semantic can ensue because of variations in the structuring of the offerings of PaaS. As an instance, a certain supplier makes use of a subject programming language for describing the version and the language, for example, Java 1.6, whereas additional platform offering makes use of 2 dissimilar subjects. This clash is caused because of the variations at the offering units of PaaS, plus particularly to how non-functional semantics are structured. The clash takes place at the level of data structure stratum, as it is a result of the diverse rational demonstration of the identical data. Similarly, additional clashes of semantic that can ensue during the relocation of programs across platforms are categorized.

After modeling thoroughly, the essential PaaS units in a specific offering of PaaS, a semantic stratum will be applied for providing an Application Model and a PaaS Offering Model for the usual explanation of accessible offerings of PaaS [15]. PaaS suppliers will have the ability to publish their contributions according to these general prototypes. By permitting suppliers into approving a general prototype for their offerings, the program portability through the platforms will be improved. As we understood, along with the available works regarding the PSIF, classification this work as a method of describing a range of general criterions may easily be done by us.

Simple Cloud: Simple Cloud is an API that permits programmers to make use of facilities of storage free from specific platforms of the cloud. Inter alia, it proposes 2 basic facilities: (i) File Storage Facility and (ii) Document Storage Facility. The File Storage Service permits for execution processes on documents such as metadata storing, reading, storing, copying, deleting, and so on. All of these are performed through offering the supposed ― adapters of storage facility which permit creators to use storage facilities by Microsoft Azure, Rack space, and Amazon along with different ones, with the use of the identical code of programming. The Document Storage Facility summarizes the interfaces of every chief database of documents, once more permitting creators to access diverse suppliers via one sole API. Microsoft, Rack space, IBM, Go Grid, along with numerous other cloud facility providers maintains Simple Cloud. Through providing an API for stockpiling information that conceptualizes/hides every exclusive one, Simple Cloud may be thought of as a transitional stratum for decoupling programs from openly retrieving the storage contrivances of particular platforms.

V. DATA PORTABILITY IN CLOUD

The Data Portability between two clouds systems can be described by the model perspective with three facets as depicted in the following table:

Table 1.

Facet	Aim	Object	Examples
Policy	Meet pertinent laws, guidelines, and strategies	Laws, guidelines, policies	Individual information guidelines, Cross outskirt information move laws, Security strategies
Syntactic	Receive data in a readable structured format	Data	JSON, XML
Semantic	Understand the meaning of ported data	Information	OWL

To perform the data portability and cultivate good results, all three aspects should be met. Syntactic errors can be tackled through a syntax converter for mapping the original syntax to the targeted syntax. On the other hand, semantic issues can be corrected by altering the information to the target data format. It might be not possible based on variability's between the origin and the target data designs. Regarding the policy criteria, the portability of data may not be possible, for example, when the data movement isn't very legal. Otherwise, some extra ideas are necessary, for example, redaction of a few data parts).

VI. APPLICATION PORTABILITY IN CLOUD

There is a model to describe the cloud portability where 5 criteria are mentioned.

Table 2.

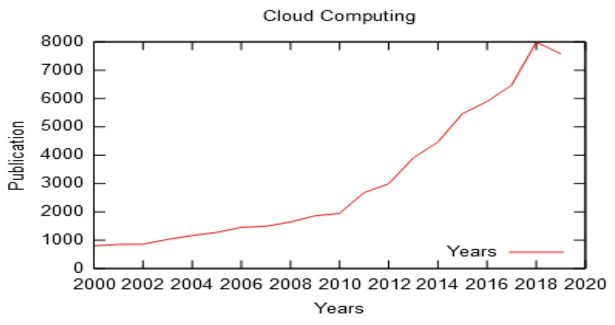
Facet	Aim	Object	Examples
Metadata	Realize as well as utilize the metadata that specifies environmental dependencies to execute the application	Metadata artifacts	YAML, JSON, Script, XML
Policy	Fulfill important rules as well as policies in association with application use	Laws, regulations, policies	Individual information guidelines, Cross fringe information move laws, Security strategies
Instruction	Perform application orders appropriately	Executable artifacts	Java, C++, BPEL
Behavior	Build and perform the desired outcomes when performing the application	Application operational as well as non-operational behaviors	Verified by test suites
Syntactic	Comprehend and utilize the format of application artifacts	All application artifacts	Zip, tar, jar

When it comes to the application syntactic aspect, it needs that the main system which is targeted has a capacity of comprehending as well as utilizes the format which is used. Otherwise, such a format needs to be converted. The metadata character is about the metadata of an application alongside its issues. It basically, is constituted of particular facets or features concerning the climatic considerations to execute that application. The targeted system should understand the metadata before operating and trying to perform the application. The particular metadata should be adapted to the capacities of the target structure. Regarding the application order feature, the target system needs to be able to understand as well as perform the orders mentioned in the performance-related order/instruction. Some order is global, especially for a few languages like Java, node.js, while the need is that the target system acquires the runtime engine concerning such orders (Java VM, and so on.). In the case of other languages, which are assigned to a native order set, the need revolves around the target system, dealing with the instruction series. Otherwise, the aspects need to be recompiled about the target system. The Scheme features demand the application portability in the genre of implementing rules, norms as well as other bindings. These aspects can cover export-related norms as well when the porting requires being across international boundaries. Commercial issues like licensing of applications, other dimensions like safety schemes describing the access regulation/control or encryption necessities are also included. An unsuccessful venture can be accountable for inhibited additional expenses or the metadata alterations so that scheme requirements can be matched. The behavioral facet is associated with the operational as well as non-operational application behavior which is described by the test suite. This test suite inspects the behavioral facets to confirm that all particular demands are met. When the ported application doesn't get success in the test suite, the application code, as well as metadata, might require remodeling or changes.

VII. RESEARCHES CONCERNING PORTABILITY OF APPLICATION IN CLOUDS

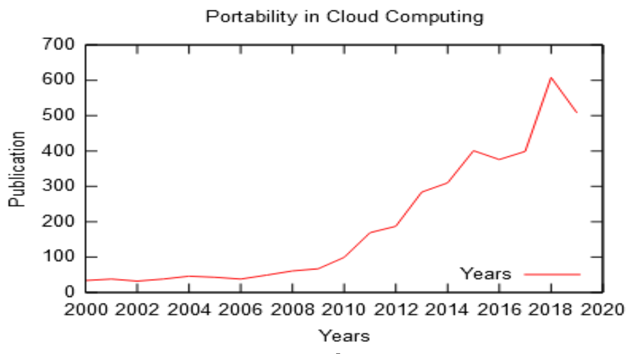
Although the seeds of cloud computing germinated in 1950 with the implementation of Mainframe computers via static clients, the rise of personal computers in the 1960s and growth of client-server architecture in the 1990s boosted the concept. The Increase in growth of virtualization around the year 2000 and then the emergence of service beyond the year 2010 made cloud computing concept the feature of utmost importance.

According to the data available from science direct, we have plotted the graph for estimating the research done after the year 2000 to date in the field of cloud computing. Graph 1 depicts the number of publications in the field of cloud computing which clearly shows how the concept has been the researches choice to develop the IT sector worldwide.



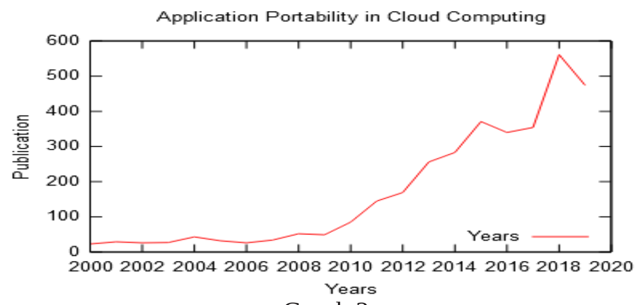
Graph.1

In graph 2 we have been concise with only portability issues in cloud computing and the graph shows the level of research in this field as vendor lock-in issue is the biggest hurdle in the growth of cloud computing. As discussed earlier portability in cloud computing can be of many types our area of concern is only application portability in cloud computing.



Graph.2

Graph 3 shows the number of publications done in this regard. We have also raised certain issues that are taken up by various researches and depicted that in Table 3.



Graph.3

Ref[26]	The Initial perspective on cloud application versatility is displayed concentrating on the best way to inspect the heterogeneous idea of cloud stages. The Approaches to handle the recognized difficulties were offered through a cross-stage application procedure.
Ref[27]	The Demand and requirement for movability and interoperability in distributed computing are offered. The methodology utilized is an open-source model that improves conveys ability and interoperability in the different cloud stages. This, in the long run, diminishes the expense of cloud relocation.
Ref[28]	The Semantics-driven answers for information and application versatility in distributed computing are introduced. The Paper centers on the issue of accomplishing information and application compactness in the cloud. The semantic web network has offered a few answers to beat various parts of cloud convey ability.
Ref[29]	The Portable cloud administrations utilizing Topology and Orchestration Specification for Cloud Applications (TOSCA) are offered. The emphasis is on the relocation of administrations between cloud suppliers to keep away from seller lock-in issues. This is done at the administrative and operational level utilizing TOSCA.
Ref[30]	A convenient cloud application from the hypothesis to practice is displayed. The methodology is to plan an open-source application that is equipped for dealing with numerous cloud utilization. It manages the issues of use, information, and administration versatility in the distributed computing condition.
Ref [31]	This is a similar investigation of administering the ascribe of the ideal to information convey ability in Europe. It talks about the challenge forestalled in cloud specialist organizations and its aversion through lock-in.
Ref [32]	The methodology is to plan a cloud convenience system utilizing a connector model. This methodology influences on TOSCA for application convey ability.
Ref [33]	The Survey on application portability in inescapable figuring is displayed. The Primary center is to inspect diverse relocation strategies for application convenience. A system was proposed alongside four measurements to permit the simple versatility of uses in distributed computing conditions.
Ref [34]	The Paper centers around rising difficulties from future cloud application situations are exhibited. Different parts of distributed computing were analyzed. The issues of combination and interoperability were likewise talked about and proposals made in transit forward.
Ref [35]	Empowering transportability in cutting edge data-driven administrations over-organized distributed frameworks are proposed in this paper. Three-layer engineering is proposed to upgrade convey ability. The middleware in the structure is flexible for a wide assortment of utilizations.
Ref [36]	The paper proposes a structure for coordinating diverse applications on the cloud which is likewise executed and approved. It utilizes TOSCA as an institutionalized meta-model that streamlines DevOps mechanization for a cloud application.
Ref [37]	The Paper approaches application conservativeness in the stage as an organization. It analyzes smallness options to the extent PaaS provider and the organic framework capacities. The offered model is executed with another educational list and particular PaaS merchants.

Table 3.

VIII. CONCLUSION

To harbor some advantages in a competitive field of present financial services, cloud computing technology is a boon. More and more providers are coming to this genre where the real hurdle is putting up a competitive price. Lucrative cost, the capacity of freeing up the employees and assign them other responsibilities, as well as the capability to give payment for required services, will be beneficial for the entire scenario. In the last, it can be inferred that cloud interoperability on IaaS as well as PaaS levels is answered. Along with other partial or incomplete resolutions, the cloud interoperability on the SaaS system is in its budding stage.

The computing and application portability is quite significant in this regard. The customer or user is inclined towards a lock-in due to the intrinsic issues in association with data as well as application portability.

PaaS cloud technology is more relevant in this aspect, mainly concerning substructure or platform. All these are built to permit the portability in the cloud to be standardized and intermediated.

REFERENCES

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica and MateiZaharia, Above the Clouds: A Berkeley View of Cloud Computing‖, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, 2009
2. P. Mell and T. Grance, The NIST definition of cloud computing‖, National Institute of Standards and Technology, Gaithersburg, 2011.
3. Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf, NIST Cloud Computing Reference Architecture‖, National Institute of Standards and Technology, Gaithersburg, 2011.
4. O. T. TC. (2016) Topology and orchestration specification for cloud applications version 1.0. [Online]. Available: http://docs.oasisopen.org/tosca/TOSCA/v1.0/csprd01/TOSCA-v1.0-csprd01.pdf
5. G. Juve and E. Deelman, Automating application deployment in infrastructure clouds, in Cloud Computing Technology and Science (Cloud-Com), 2011 IEEE Third International Conference on, 2011, pp. 658665.
6. O. C. T. Members (2016) Cloud application management for platforms, version 1.0. [Online]. Available: https://www.oasis-open.org/committees/download.php/47278/CAMP-v1.0.pdf
7. OpenTOSCA. (2016) Opentosca initiative. [Online]. Available: Http: //www.iaas.uni-stuttgart.de/OpenTOSCA/
8. OpenTOSCA. (2016) Opentosca ecosystem. [Online]. Available: http://les.opentosca.de/v1/
9. OpenStack. (2016) Heat - OpenStack Orchestration. [Online]. Available: https://wiki.openstack.org/wiki/Heat
10. Aparna Vijaya, Pritam Dash, and Neelanarayanan V., Migration of Enterprise Software Applications to Multiple Clouds: A Feature-Based Approach, Lecture Notes on Software Engineering, Vol. 3, No. 2, 2015
11. D. Petcu, Portability, and interoperability between clouds: challenges and case study‖, in Proceedings of the 4th European conference on towards a service-based internet, Poznan, 2011, pp. 62–74.
12. S Dowell, A. Barreto III, J.B Michel, and M.T. Shing, ―Cloud to Cloud Interoperability‖, in Proceedings of the 2011 6th International Conference on Systems Engineering, 2011, Albuquerque, New Mexico, pp. 258-263
13. S. Charrington, "Don't Pass on PaaS in 2010," (ebizo), [online] 2010, http://www.ebizq.net/topics/cloud_computing/features/12279.html?page=2.
14. N. Loutas, E. Kamateri, and K. Tarabanis, A Semantic Interoperability Framework for Cloud Platform as a Service‖, in 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), Athens, 2011, pp. 280–287.
15. N. Loutas, E. Kamateri, and K. Tarabanis, ―Cloud Semantic Interoperability Framework‖, Cloud4SOA, D1.2, 2011.
16. O. T. TC. (2016, Nov.) Topology and orchestration specification for cloud applications version 1.0. [Online]. Available: http://docs.oasisopen.org/tosca/TOSCA/v1.0/csprd01/TOSCA-v1.0-csprd01.pdf
17. Daniel Oliveira and Eduardo Ogasawara. Article: Is Cloud Computing the Solution for Brazilian Researchers?. International Journal of Computer Applications 6(8):19–23, September 2010.
18. Aparna Vijaya, Pritam Dash, and Neelanarayanan V., Migration of Enterprise Software Applications to Multiple Clouds: A Feature-Based Approach, Lecture Notes on Software Engineering, Vol. 3, No. 2, May 2015
19. ISO/IEC 19941 Cloud computing - Interoperability and Portability https://www.iso.org/standard/66639.html
20. Kuyoro S.O., Ibikunle F., Awodele O., ―Cloud Computing Security Issues & Challenges‖, IJCN, Vol. 3 Issue 5: 2011, pp. 247-255.
21. Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang, "A survey of Mobile Cloud Computing: Architecture, Applications and Approaches", Wireless Communications and Mobile Computing, 2011
22. Cloud Standards Customer Council (2013). Migrating Applications to Public Cloud Services: Roadmap for Success. http://www.cloud-council.org/deliverables/migrating-applications-to-public-cloud-services-roadmapfor-success.htm
23. Cloud Technology Partners: Driven by Multi-Cloud, the Public IaaS Market Begins to Equalize. http://www.cloudtp.com/2014/06/27/driven-multi-cloud-public-iaas-cloud-market-begins-equalize
24. National Institute of Standards and Technology: US Government Cloud Computing Technology Roadmap Volume II Release 1.0 (Draft) – Useful Information for Cloud Adopters. http://www.nist.gov/itl/cloud/upload/SP_500_293_volumeII.pdf
25. Cloud Standards Customer Council (2017). Practical Guide to Cloud Computing. http://www.cloud-council.org/deliverables/practical-guide-to-cloud-computing.htm.
26. Gonidis F, Paraskakis I, Simons AJH, Kourtesis D. Cloud application portability: An initial view. BCI'13; September 19-21, 2013. ACM. 978-1-4503-1851-8/13/09.
27. Bozman J. Cloud Computing: The Need for Portability and Interoperability. IDC Executive Insights. www.idc.com/gms; 2010.
28. Ranabahu AH, Sheth AP. Semantics centric solutions for application and data portability in cloud computing. In: Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science; 2010.pp.234-241. http://corescholar.libraries.wright.edu/knoesis/766.
29. Binz T, Breiter G, Leyman F, Spatzier T. Portable Cloud Services Using TOSCA. IEEE Internet Computing. Vol. 16. no. 3. pp. 80-85. May-June 2012. DOI: 10.1109/MIC.2012.43
30. http://doi.ieeecomputersociety.org/10.1109/MIC.2012.43.
31. Petcu D, Macariu G, Panica S, Crăciun C. Portable cloud application-from theory to practice. Future Generation Computer Systems. 2012; **29**(2013):1417-1430.
32. Van der Auwermeulen B. How to attribute the right to data portability in Europe: A comparative analysis of legislation. Computer Law & Security Review. 2017; **33**(2017):57-72.
33. Kostoska M, Gusev M, Ristov S. A new cloud services portability platform. In: 24th DAAAM International Symposium on Intelligent Manufacturing and Automation 2013; 2014.
34. Yu P, Ma X, Cao J, Lu J. Application mobility in pervasive computing: A survey. Pervasive and Mobile Computing. 2012, 2013; **9:2**-17.
35. Jeferry K, Kousiouris G, Kyriazis D, Altmann J, Ciuffoletti A, Maglogiannis I, Nes P, Suzic B, Zhao Z. Challenges emerging from future cloud application scenarios. Procedia Computer Science. 2015; **68**(2015):227-237.
36. Pujo-Ahulló J, López PG. Enabling portability in advanced information-centric services over structured peer-to-peer systems. Journal of Network and Computer Applications. 2010; **33**(2010):556-568.
37. Wettinger J, Bietentúcher U, Kopp O, Leymann F. Streamlining DevOps automation for cloud application using Tosca as a standardized metamodel. Future Generation Computer Systems. 2016; **56**(2016):317-332.
38. Kolb S, Wirtz G. towards Application Portability in Platform as a Service. In Proceedings of the 2014 IEEE 8th International Symposium on Service-Oriented System Engineering (SOSE '14). Washington, DC, USA: IEEE Computer Society; 2014:218-229.

DOI=http://dx.doi.org/10.1109/SOSE.2014.26.
39. Mell P, Grance T. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. National Institute of Standards and Technology, Information Technology Laboratory. 2011; **145:7**. https://doi.org/10.1136/emj.2010.096966.
40. Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop (GCE'08); 2008. DOI: 10.1109/GCE.2008.4738445
41. Stanoevska-Slabeva K, Wozniak T. Grid and Cloud Computing: A Business Perspective on Technology and Applications. Heidelberg/Dordrecht/London/New York: Springer; 2010. ISBN: 978-3-642-0 e-ISBN 978-3-642-05193-7. DOI: 10.1007/978-3-642-05193-7.
42. Martino BD, Cretella G, Esposito A. Classification and Positioning of Cloud Definitions and Use Case Scenarios for Portability and Interoperability. Rome: 3rd International Conference on Future Internet of Things and Cloud; 2015. pp. 538-544. DOI: 10.1109/FiCloud.2015.119
43. Munisso R, Chis AE. CloudMapper: A Model-Based Framework for Portability of Cloud Applications Consuming PaaS Services. 25th Euromicro International Conference on Parallel. St. Petersburg: Distributed and Network-based Processing (PDP); 2017. pp. 132139.DOI: 10.1109/PDP.2017.94
44. Pozdniakova O, MaZeika D. A Cloud Software Isolation and Cross-Platform Portability Methods. Open Conference of Electrical, Electronic and Information Sciences (eStream).Vilnius; 2017. pp. 1-6. DOI: 10.1109/eStream.2017.7950315
45. Martino BD, Cretella G, Esposito A. Applications Portability and Services Interoperability among Multiple Clouds. In: IEEE Cloud Computing; Vol. 2. no. 2. pp. 22-28. Mar.-Apr. 2015. DOI: 10.1109/MCC.2015.38
46. Brinkley J, Hoffman D, Tabrizi N. A Social Networking Site Portable Profile System for Blind and Visually Impaired Users Based on Cloud and Semantic Web Technologies.
47. Honolulu, HI: IEEE International Conference on Cognitive Computing (ICCC); 2017. pp. 104-111. DOI: 10.1109/IEEE.ICCC.2017.21
48. Parameswaran AV, Asheesh C. Cloud Interoperability and Standardization. in SETLabs Briefings. 2009;**7**(7):19-27
49. Moravcík M, Segec P, Papán J, Hrabovský J. Overview of Cloud Computing and Portability Problems. 15th International Conference on Emerging eLearning Technologies and Applications (ICETA). Stary Smokovec; 2017. pp. 1-6. DOI: 10.1109/ICETA.2017.8102511
50. Antoniades D et al. Enabling Cloud Application Portability. IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC). Limassol; 2015. pp. 354-360. DOI:10.1109/UCC.2015.56
51. Scandurra P, Psaila G, Capilla R, Mirandola R. Challenges and Assessment in Migrating IT Legacy Applications to the Cloud. IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA). Bremen, Germany: 2015. pp. 7-14. DOI: 10.1109/MESOCA.2015.7328120
52. Atayero AA, Ilori OA, Adedokun MO. Cloud security and the Internet of Things: Impact on the virtual learning environment. In: Edulearn15 7th International Conference on Education and New Learning Technologies; 2015. pp. 3857-3863
53. Martino BD, Esposito A, Cretella G. Semantic representation of cloud patterns and services with automated reasoning to support cloud application portability. IEEE Transactions on Cloud Computing. 2017;**5**(4):765-779
54. Martino BD, Cretella G, Esposito A. Semantic and Agnostic Representation of Cloud Patterns for Cloud Interoperability and Portability. IEEE 5th International Conference on Cloud Computing Technology and Science. Bristol; 2013. pp. 182-187. DOI: 10.1109/CloudCom.2013.123
55. Yangui S, Glitho RH, Wette C. Approaches to end-user applications portability in the cloud: A survey. IEEE Communications Magazine. 2016;**54**:138-145
56. Markoska E, Kostoska M, Ristov S, Gusev M. Using P-TOSCA to prevent vendor lock-in for cloud-based laboratories. 23rd Telecommunications Forum Telfor (TELFOR).Belgrade; 2015. pp. 982-985. DOI: 10.1109/TELFOR.2015.7377629
57. Kolb S, Röck C. Unified Cloud Application Management. IEEE World Congress on Services (SERVICES). San Francisco, CA; 2016. pp. 1-8. DOI: 10.1109/SERVICES.2016.7
58. Markoska E, Chorbev I, Ristov S, Gusev M. Cloud Portability Standardization Overview.38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija; 2015. pp. 286-291. DOI: 10.1109/MIPRO.2015.7160281.

AUTHOR PROFILE

Brijesh Pandey has completed his M.Tech from MNNIT Allahabad. He is pursuing PhD from Dr. APJ AKTU Lucknow. He has published more than 10 research papers in various International Journals. He has been guided 5 M.Tech students.

Dr. Sudhir Singh Soam has received his PhD from Dr. APJ AKTU Lucknow. He is currently workingin IET Lucknow. He has published many research papers and journals and delivered invited talks in reputed conferences. He is associated as a expert member of various universities. He is member of many regulatory bodies and panel member of RDC committees.

Dr. D. K. Yadav has done PhD from IIT Bombay. He is currently working in MNNIT Allahabad. He has published 25 international conference and journal papers. He has delivered many expert talks, invited talks and has been member of selection committee.He attended many short term courses and conferences. He has supervised 46 M.Tech students.