A novel approach for services management and selection in intercloud system

Lohit Kapoor
Sreenidhi Institute of Science and Technology, India, Email: lohitkapoor@sreenidhi.edu.in,
Seema Bawa
Thapar University, Patiala, India, Email: seema@thapar.edu,
and
Ankur Gupta
Model Institute of Engineering and Technology, Jammu, India, Email: ankurgupta@mietjammu.in

Service deployment, orchestration, provisioning and SLA-compliance present a challenge in intercloud environments. A comprehensive and unified service management framework is required so that service providers and consumers can leverage the intrinsic benefits of services deployed across different cloud service providers exploiting latency and cost advantages while ensuring scalability, load-balancing and high-availability. This research paper presents architecture for unified services management in the intercloud environment allowing users seamless access to services through an optimal service selection mechanism providing on-demand ranking on several quality parameters. Experimental results establish the effectiveness of the proposed scheme.

Keywords: Intercloud, Services Management in Intercloud, Customized Service Ranking and Selection

1. INTRODUCTION

Services management in an intercloud environment encompasses the deployment, configuration, provisioning, operation and portability of cloud resources across different Cloud Service Providers (CSPs). The intercloud provides a strong motivation for the deployment of large-scale services which need to cater to diverse geographical locations of their users besides optimizing latency and cost while ensuring quality-of-service and complying with service-level agreements. Thus, a mechanism which allows users customized and seamless access to services in an intercloud scenario is required. A major concern for any inter-cloud services management framework is the orchestration of services across CSPs and allowing end-users fine-grained control over how they select and consume services without knowing the deployment and location details [Petcu et al. [2013]].

The intercloud has emerged as a logical evolution of the cloud computing paradigm allowing for the creation of a community of CSPs to offer greater value-add to the end consumer while facilitating enhanced elasticity, ensuring QoS even at peak loads, service/data portability and migration and collaborative services for mutual benefits. For small and medium CSPs this model is fairly intuitive from a resource sharing perspective [Gupta et al. [2011]]. For Large CSPs the intercloud offers them the possibility of offering services in the geographical proximity of their customers and improving responsiveness. The intercloud also alleviates the issue of data/vendor lock-in. In any case end-users want flexibility to shift from one cloud to another or from one service to another, due to various reasons like performance degradation, high cost, legal issues etc. Service portability or redeployment can solve these issues [Petcu et al. [2013]][Krintz [2013]], but a large numbers of CSPs makes it difficult for service providers and end users to decide which CSP will be a best-fit for their requirements.

Cloud brokers have emerged as intermediaries between service providers, users and the CSPs [CCB]-[Lucas-Simarro et al. [2013]]. However, current cloud brokers do not provide advanced service management capabilities across CSPs to allow end users to select the service instances that best meet their requirements [Lucas-Simarro et al. [2013]]. To enable such a selection mechanism, detailed service monitoring on various performance parameters over sustained periods shall be necessitated. Traditional performance management techniques mainly focus on vm-level monitoring within a CSP, but in the intercloud scenario we need to have a complete performance view of services across CSPs. Clearly a more comprehensive service-level monitoring mechanism is required to take informed decisions on service selection which meet user-defined criteria.

Consider a Service Consumer (end user) who requires access to a data-intensive service with constraints over latency and/or data transfer costs. Therefore it is logical to choose the CSP whose data-center (where the service resides) is located closest to the consumer of that service. Similarly, for compute-intensive services a user would like to choose a service instance which offers the lowest cost while delivering acceptable performance. Thus, the nature of services and the service deployment models play an important part in service selection. However, the service selection process in an intercloud environment can be expected to throw up some counter-intuitive results. This is because network and service performance can vary significantly over time-zones and periods of peak-usage [Lucas-Simarro et al. [2013]][Wu and Madhyastha [2013]]. There is a need to focus on how a service is performing over a period of time and at specific times. Moreover a performance comparison between different service instances would also provide greater insights for service selection facilitating delivery of the best possible cost to performance proposition to the end user. At the other end of the spectrum are the Service Providers who need insights into the consumption pattern of their services to facilitate dynamic deployment and scaling to maximize revenues while meeting customer requirements. Facilitating optimal service provisioning and consumption is therefore non-trivial and a major challenge in intercloud environments.

This research paper proposes a comprehensive Unified Services Management Framework for the intercloud environment - the “Services Cloud (SC)” , a distributed trusted third party framework which encompasses brokering services, service performance management and a service ranking and selection mechanism for end-users. The SC receives users request, provides a customized ranking of various available services (across CSPs) and allocates the service based on user selection. Further, in the case the user finds the services not meeting their expectations, it can select a different service instance on the intercloud. Moreover, a service provider can also initiate a migration of a service instance to some other CSP to better meet its SLAs with the end users. The aim of the proposed framework is to enable the users to consume customized services as per their choice in the intercloud ecosystem and present them a variety of services options which best meet their requirements.

The main contributions of this paper are:

—Proposes a Services Cloud for managing services in an intercloud environment.

—Presents a service ranking mechanism in an intercloud based on detailed performance monitoring and historical analysis.

—Presents a fine-grained control mechanism to the end users for optimal service selection based on different parameters.

—Presents model of automated service scaling and deployment for service providers in an intercloud based on dynamic service consumption patterns.

The rest of the paper is organized as follows: section 2 presents the background and related work while section 3 illustrates the use-cases for service deployment and consumption in the intercloud. The System Model of the proposed Services Cloud is described in section 4 with details of operation, modeling of services and performance metrics considered. Section 5 contains the experimental setup and presents the simulation results for different experiments conducted. Finally section 6 concludes the paper and presents some directions for future work.

2. BACKGROUND AND RELATED WORK

Intercloud service management is challenging yet essential in deploying the next-generation of large-scale global services which span multiple Cloud Service Providers (CSPs). Many of current vendors provide solutions to manage services across multiple clouds. TM forum (TMF) [ICS] presents a unified service delivery management model which focuses on deployment of federated services. It builds standards for services in intercloud environment known as Inter-cloud Services (ICS) which converges towards a singular approach to service orchestration of networks and data-centers of multiple providers. According to ICS, "*Service providers* require an integrated, flexible, automated, service-focused intercloud management system" . As per Forrester Researchs, Cisco commissioned research on Global Managed Services Opportunity (2009) the demand for managed "on-demand" services is identified as a key shift in user preferences [MSP]. Therefore the necessary intercloud infrastructure and middleware required to support service orchestration across CSPs needs to be in place to realize this market requirement.

Research in intercloud services management is in nascent stage. Existing literature in the field addresses some challenges of deploying and managing services in an intercloud environment, but the big picture seems missing. Authors in [Lucas-Simarro et al. [2013]] propose a broking mechanism in a multi cloud environment which looks at various aspects of pricing schemes, automatic decisions for service elasticity, optimization and finding the perfect cloud for new service deployment. However this work does not take into account latency between the service provider and the consumer which may result in sub-optimal service selection. The work presented in [Wu and Madhyastha [2013]] figures out the latency benefits for optimal service deployment and minimizes the service response time to user. The experimental results in this work are based on only one real-world cloud. Hence, the comparison between implementation of service instances in different CSPs is not made.

Several brokers exist in literature which can operate across CSPs and focus on specific issues such as interoperability, performance monitoring of virtual machines, data migration and orchestration. RightScale [RHP] is a real world cloud broker around a middleware which is adaptable and automated. It analyzes past events for better cloud control, administration, and life-cycle management of applications across multiple clouds. It relies on monitoring the performance parameters of instances of different clouds (in terms of virtual machine performance) but is service agnostic. Service performance can vary significantly in different production environments, but RightScale does not provide any mechanism to monitor the performance of different services across CSPs. Thus, optimal service selection remains a challenge. Zimory [ZHP] cloud management platform enables cloud brokers to create a cloud eco-system including non-cloud providers to provide intercloud services. However, it does not provide any scheduling mechanisms or any decision making technique to manage services or aid the end user in selection of service instances which best meet their requirement. Aeolus [Garg et al. [2013]] is an open source, Ruby-based cloud management software which allows users to choose between private, public or hybrid clouds, using DeltaCloud [DC] cross-cloud abstraction library. But it is not aware of monitoring, scheduling and pricing schemes of different clouds, making it tough for the users to decide on the most economical service for their workload. Rackspace [CM] cloud monitoring brokering service lets users monitor its websites whether located in Rackspaces own data centers or any other cloud and use graphs to analyze trends, outliners and patterns of their allotted servers but scheduling across CSPs is not handled. Cohesiveft [CHP] provides enterprise-grade virtualization and cloud migration services. Its main contribution is the transfer of applications comprising application template, operating system images, libraries and system components, across private, public or hybrid clouds in an automated manner but service-level monitoring and fine-grained control over service selection to the user is not considered.

Authors in [Calheiros et al. [2012]] propose a "cloud coordinator" between multiple clouds which allows customers to dynamically scale their services for optimal performance. The introduction of middleware cloud coordinator allows a service to improve its performance, reliability and scalability. However this model does not consider the dynamic change in market price of resources. Also, the view is service-centric and not user-centric. Authors in [Itani et al. [2014]] present a routing technique for managing service collaboration among different cloud providers. It works for the stability and efficiency of overall routing processes. This protocol deals with the changing configuration and traffic overheads in real cloud but fails to take decision based on internal performance of the service. Authors in [Garg et al. [2013]] present a ranking mechanism of different services in order to provide a comparative view to users to chose a particular service over another under different use-cases. In this work authors proposes SMICloud Broker which performs service discovery and ranking on different Key Performance Indicators (KPIs). However this mechanism does not provide any autonomic technique to redirect the users request in the case of flash-crowd scenario, fault incidence etc. to ensure minimal SLA violations and revenue loss to the service provider.

Thus, most of the current cloud brokering mechanisms provide scheduling or routing of service requests based on monitoring the virtual machines on which the services are deployed. In most of the cases customers can select, monitor and migrate these virtual instances across CSPs without having a comprehensive service-view of the intercloud. The service-view of the intercloud is important to a) optimize service deployment and management for the service provider (auto-scaling, replication, migration) b) optimize service selection for the end-user (cost, response-time, QoS-compliance). This research paper provides details of the framework to meet the above requirements and proposes a Services Cloud (SC) with a services broker which manages service orchestration and consumption across different CSPs in a seamless manner.

3. USE-CASES: SERVICE DEPLOYMENT AND CONSUMPTION

To better understand the requirements for a Services Cloud for the intercloud the perspectives of all the entities need to be considered. We identify the following use-cases for the two main entities in an intercloud; the Service Consumer and the Service Provider:

(1) **Service Consumer**
Use Case 1: Service Selection
Consider the case where a user wants to execute a job. In the intercloud a large number of similar services which meet the user requirements may be available. Different services deployed across different CSPs will have different performance levels and might entail different costs. Therefore, to select the service which best meets the stated requirements of the end user is not trivial. The SC should facilitate optimal service selection for the end-user.
Use Case 2: Changing Service Provider
Consider a case where the Service Consumer is not satisfied with the services offered by the current Service Provider and wants to shift to other Service Provider. The SC should facilitate seamless service consumption across Service Providers without data lock-in.
Use Case 3: Fine-grained Control
An end-user might want greater control in how it consumes the services. For instance the end-user might impose cost constraints, time constraints, latency constraints, geographical constraints or other requirements on service characteristics which the SC should be in a position to satisfy. The SC should also provide detailed service-related information to allow the end-user complete control over the mechanism of service selection and consumption.
Use Case 4: Seamless Interaction
An end-user should not be concerned about the location of the CSP which hosts the service being consumed nor should it be aware of how its service requests are routed and serviced across the intercloud. The SC should therefore abstract the underlying details of the CSPs and service deployment from the end-user, which should focus on just selecting and consuming services through a standardized and seamless interface.

(2) **Service Provider**
Use Case 5: QoS Compliance

The intercloud is a highly dynamic environment which needs to cater to flash-crowd scenarios and volatile resource requirements. From a Service Provider perspective it is imperative that the SC ensures that deployed service instances are continuously monitored and dynamic load-balancing, fault-tolerance and elasticity be provided to ensure that the Service Provider agreed QoS thresholds with the end-user are not violated.

Use Case 6: Exploiting SC Locality

Responsiveness of a service is an important performance parameter for any SP. Thus, a SP needs to be aware of the geographical distribution of its end-users and dynamically deploy service instances across the intercloud to improve service responsiveness. A SC should facilitate such dynamic deployment in response to the geographical location of the end-users.

Use Case 7: Maximize SP RoI

A Service Provider needs specific insights into how its service is being consumed (average and peak load, user location, average cost) and its performance (response time, reliability, SLA violations etc.) besides the various service hosting costs across the intercloud to maximize its RoI. The SC should allow a Service Provider enough control to allow it to optimize its service deployment strategy.

Use Case 8: Geographical Aware Auto-Scaling

Geographical Location is very important when dealing with varied users requests. Further Service Provider have the option to deploy services such that they are near to users location. A Service Provider may need to dynamically scale-up or scale-down service instances in different geographic regions depending upon the number of users requests form a particular region. Therefore the SC should provide this facility.

4. SYSTEM MODEL

In this section the detailed system model of the proposed Services Cloud (SC) is discussed. A Cloud Service Provider (CSP) can have multiple data-centers at different geographical locations. Data-Centers belonging to a CSP are managed by a central broker which distributes the resource requests within the CSP. Each CSP participates in a federation of CSPs i.e. the intercloud. The schematic of the proposed Services Cloud (SC) is shown in Fig 1 and has its own broker for scheduling services across CSPs. Thus the SC broker talks to individual CSP brokers for deploying/scheduling services across the intercloud. Each service may consist of multiple instances which can be deployed at any CSP in the intercloud.

Each Service Provider which is desirous of hosting services in the intercloud needs to register its services with the SC by specifying the service characteristics (name, type, category, cost etc.), resource requirements, constraints and the deployment policy (fixed or geographically-aware auto-scaling). There are several standardized service description formats available such as [SPF][WSD][TF]. Based on the inputs provided by the SP, the SC broker proceeds to deploy the service to a specific CSP or across multiple CSPs. The SC is also responsible for monitoring all the active deployed service instances in terms of their performance (load, resource usage, response times, latency, reliability, availability etc.) over sustained periods of time. There exists several standard service performance monitoring frameworks in practice [ICS]. This is needed to build a historical performance profile of each service and its usage patterns. These insights are used both by the SP in fine-tuning its deployment and provisioning strategies and by the end-user in selecting the services which best meet its requirements. Thus, the SC is responsible for orchestrating services across CSPs. Service orchestration can be achieved by using commonly used APIs of different CSPs. For example Appscale [Krintz [2013]] is a private PaaS which supports and maintains the commonly used APIs from Google App Engine stack [GCA]. Therefore any app/service which works on Google App Engine will run on Appscale as well.

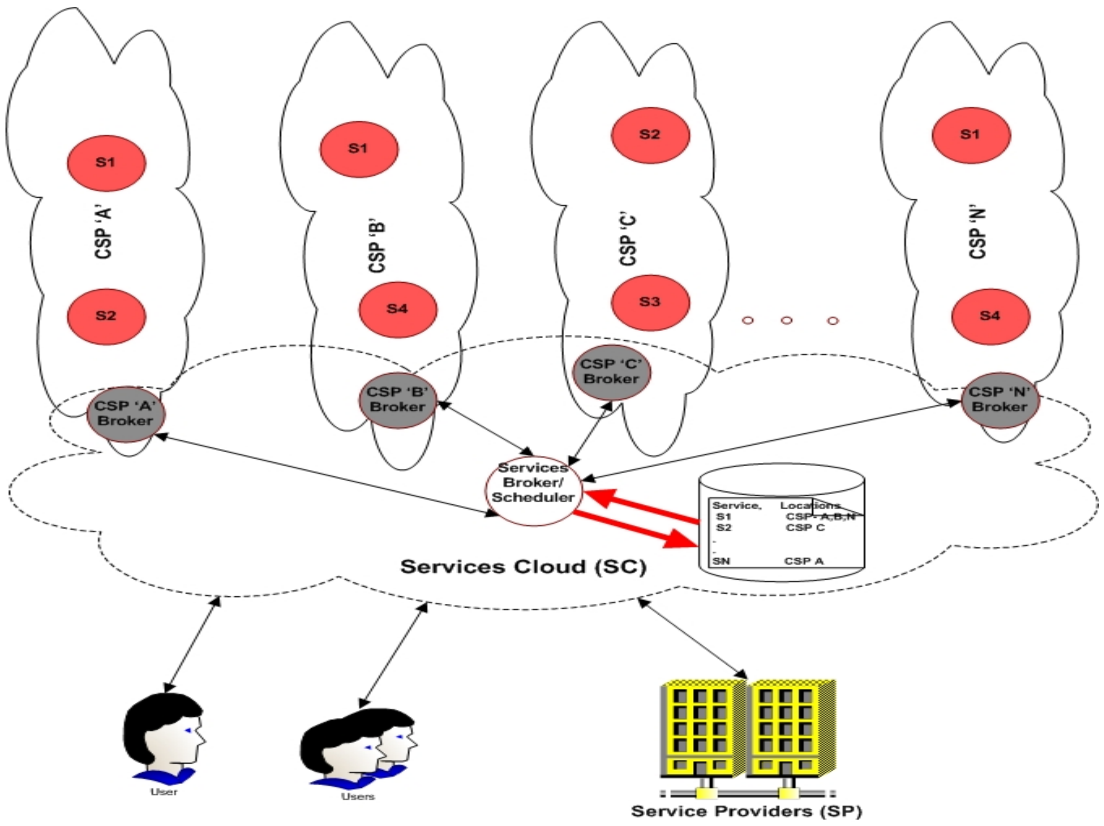

Figure 1: Conceptual Model of Services Cloud (SC).

The proposed Services Cloud (SC) architecture for the intercloud environment is depicted in Fig 2. At a broad level the architecture consists of four entities: a) Services Cloud b) Cloud Service Providers c) Service Providers and d) Service Consumers. The SC is not monolithic, but composed of several sub-components. Prominent among these are the user-management module, services performance monitoring module, services deployment and management module and the broker module (for routing end-user requests to appropriate/selected service) through the CSPs broker.

4.1 Major Operations

A) Service Registration

Service Providers (SPs) need to register their service offerings with the Services Cloud (SC) through a well-defined interface. A service registration request contains complete service description including generic information, service components, related files and dependencies which facilitate automated service deployment. Upon successful registration services are reflected in the central service directory maintained by the SC.

B) Service Deployment

The process of service deployment is based on the resource requirements and constraints of a particular service. For instance a specific platform may be a dependency for service deployment. Also, the physical resources (number and desired configuration of virtual machines), their location and cost ($ per hour per virtual machine) can be important constraints for the SP. In that case the SC deploys the service at the CSP which best meets the requirements of the SP. The SP can also enable the dynamic provisioning functionality in the SC, which allows for auto-scaling (service duplication or provision of additional physical resources) to meet volatile service requests.

C) Service Monitoring

Once a service is deployed by the SC, the Service Monitor (a Real-time monitoring Tool) monitors each service. It obtains the service handle from the service directory and keeps sending heartbeat messages to the deployed service instance after a pre-defined interval for checking uptime, availability and response times. It also obtains detailed performance information from the local service monitor deployed at the CSPs servers which monitor the resource usage of each deployed service instance. Service monitor also entails keeping track of individual service queues maintained by the SC. This gives indication of the service load and allows the correlation between load and service performance to be established. Thus, trend analysis and performance profiling of each service instance is performed. Thus, a strong basis for service ranking and selection and further optimization is created by the service monitor.

D) User Registration

User registration involves account creation and generation of authentication credentials. The user management module maintains a profile for each user with services consumed, feedback/rating and billing information included.

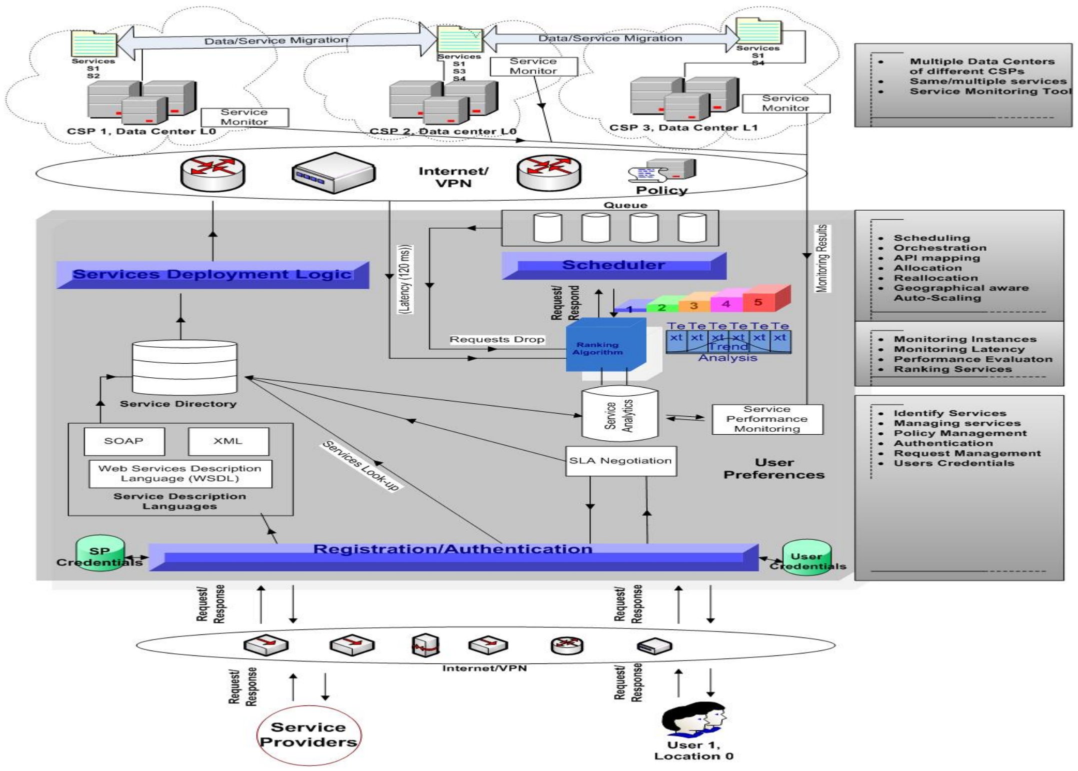

Figure 2: Schematic of intercloud service management system.

E) Service Ranking and Selection (SRS)

Since detailed performance profiles of each service instance are created by the service performance monitoring module it is straightforward for the SC to rank comparable services (belonging to the same category). The service performance monitoring module is also responsible for providing feedbacks to Service Analytics for evaluating different services. This is done through a weighted formula based on different parameters such as latency, reliability, availability and user rating while remaining within the overall cost constraints specified by the consumer. Hence Ranking Algorithm ranks the services in ascending order keeping the parameters into consideration. Although the SC maintains these internal rankings based on the weighted formula, the end-users are free to specify their own weighted formula allowing customized rankings to be computed on demand for individual end-users. This is a major contribution of the proposed framework. Subsequently, the user selects the appropriate service and negotiates the SLAs with the selected service. The details of SRS are presented in Section 4.3.

F) Service Consumption and Management

All the preceding operations contribute to major real-time optimizations during service consumption phase. If the SC detects that the service request queue has reached its threshold (can be determined by observing past correlation between service queue length and response time) then the SC can perform auto-scaling including duplication of service instance. Similarly if service requests from a particular geographic region are pervasive and the SLA agreement contains a latency threshold then an additional service instance can be deployed at a CSP in that geographical region. Similarly, if the resource costs at the CSP hosting the deployed service goes beyond the cost constraint specified by the SP, the SC can migrate the service to a CSP which offers resource cost within the specified cost constraint of the SP. Similar optimizations are possible at the user level as well allowing users to switch between service instances or even competing service providers depending upon which better meets their requirements. Thus, SC allows multiple optimizations to be performed dynamically both in the context of the SP and the end-user giving all stakeholders flexibility in meeting their objectives.

G) Accounting

Finally, after the services have been consumed, requests processed and violations noted, the accounting process is applied to compute the dues of the end-users. Similarly, the dues of the SP towards the CSP are also computed and the settlements processed. However this process is beyond the scope of this paper.

4.2 Service and Performance Modeling

An Intercloud (Ic) environment consists of federation of N CSPs given by

Ic = $\{ \text{CSP}_1, \text{CSP}_2, \ldots\ldots, \text{CSP}_N\}$

Each CSP consists of multiple data-centers located in M different geographical locations across globe i.e.

CSP = $\{ \text{DC}_1, \text{DC}_2, \ldots., \text{DC}_M\}$

Let S be the set of t total services offered by an Ic, such that ,

S = $\{ \text{S}_1, \text{S}_2, \ldots , \text{S}_\text{t}\}$

Let x be the total number of services components for each service S deployed in CSP$_N$, such that

S$_t$= $\{ \text{S}_{\text{c1}}, \text{S}_{\text{c2}},\ldots, \text{S}_x\}$

A service S can be deployed in multiple CSPs with individual components distributed in a way such that

$\{ \text{S}_\text{t}(\text{S}_{\text{c1}}\varepsilon\ \text{CSP}_1), (\text{S}_{\text{c2}}\ \varepsilon\ \text{CSP}_2), ..,(\text{S}_{\text{c(x-1)}}\ \varepsilon\ \text{CSP}_{N\text{-}1}), (\text{S}_{\text{cx}}\ \varepsilon\ \text{CSP}_N)\}$,

or single monolithic service is deployed/replicated at multiple CSPs, such that y multiple service instances exist at several CSPs, such that

$\{ \text{S}_\text{t}\text{S}_{\text{t1}}\ \varepsilon\ \text{CSP}_1, \text{S}_{\text{t2}}\ \varepsilon\ \text{CSP}_2,\ldots , \text{S}_{\text{ty}}\ \varepsilon\ \text{C}_N\}$,

or a single service component is replicated at multiple CSPs

$\{ \text{S}_\text{t}\text{S}_{\text{c1}}\ \varepsilon\ \text{CSP}_1, \text{S}_{\text{c1}}\ \varepsilon\ \text{CSP}_2, \ldots , \text{S}_{\text{c1}}\ \varepsilon\ \text{CSP}_N\}$,

And finally where a monolithic service or all components of a service are deployed at the same CSP and maybe even at the same data center within the CSP.

Thus, multiple deployment scenarios are supported. Services are ranked as per their performance which constitutes of various performance parameters.

In the proposed model users can communicate their service preferences through: a) Cost (c), and

b) Service Quality (p),

$$0 \leqslant p \leqslant 1.$$

The cost parameter c is the amount that a user is willing to pay for consuming a service. Service Quality p is a cumulative parameter which allows users to customize their requirements.

4.3 Performance Metrics

Service Measurement Index (SMI) [AS] is a set of business-relevant Key Performance Indicators (KPI's) that provide a standardized method for measuring and comparing a business service regardless of whether that service is internally provided or sourced from an outside company. SMI enables individual preferences to be the basis for what defines a good service. We have identified five parameters as shown in Table 1 from SMI as a basis for defining service quality *(p) for the proposed framework.*

Table 1. Identified Parameters

Service Quality Parameters	Standard Weightage
Network Latency (ϕ_1)	0.20 (wt_1)
Processing Time (ϕ_2)	0.20 (wt_2)
Reliability (ϕ_3)	0.20 (wt_3)
Reputation (ϕ_4)	0.20 (wt_4)
Availability(ϕ_5)	0.20 (wt_5)

Network Latency: We measure the round trip delay between sending a service request and receiving the response. Latency varies based on geographical location; therefore we continuously measure latency for each time slot n as:
$\phi_1 = (^{n}\sum_{t=0} (\alpha / \tau)) / n$
Where α = minimum latency observed during ideal time
τ = real latency observed at that time,
n = total number of requests.
Processing Time: It is the measure of time a service takes to process a job request. This depends on several factors including the service architecture, speed of CPU and available cores, load on the sever hosting the vm which hosts the service and service load etc. Therefore, we calculated processing time as
$\phi_2 = (^{M}\sum_{j=1} (\beta / \lambda)) / M$
Where β = minimum processing time observed during ideal time
λ = real processing time observed at any time,
M = total number of requests in a time period.
Reliability: Reliability is an important factor in computing service quality. In this case we monitored service instances during peak and lean hours and observed the service responses received and calculated the Mean Time Between Failures (MTBF). Once MTBF is calculated we can find out the reliability of a service by using following equation [McClusky and Mitra [2004]]:
$\phi_3 = \exp(-t/MTBF) \leqslant 1$
Where e = exponential function,
t = minimum expected processing time for node to deal for any tasks execution,
MTBF = the failure rate of the node at the give time.
Reputation: Reputation of a service is a multi-faceted concept. Reputation is calculated based on the feedback provided by the user community about their previous experiences which is ranked between 0 and 1. On completion of every service usage cycle, users rate the service as 0 (not recommended), 0.25 (poor) 0.5 (acceptable), 0.75 (good) and 1 (excellent). Our Ranking algorithm calculates the reputation score similar to proposed by [Wishart et al. [2005]] which includes service ID, consumer ID, timestamp (used to determine the aging factor of a particular service rating).

Therefore, $\phi_4 = {}^{N}\sum_{i=1} S_i \lambda^{di} \leqslant 1$
Where N = number of ratings for a service,
S_i = ith service ratings
λ = inclusion factor i.e. $0 \leqslant \lambda \leqslant 1$
di = age of the ith service ratings in days.
The inclusion factor λ used to signify the recent ratings of the service. For example smaller value of λ means more recent ratings which have more impact on reputation and larger λ means more of the reputation influences the reputation scores.
Availability: This parameter is simply obtained by observing the total time duration for which the service remains down relative to the total time the service is offered. Therefore,
$\phi_5 =$ 1- $(t_{down} / t_{up}+t_{down}) \leqslant 1$
t_{up} = time for which service remains up during a partuclar period of time
t_{down} = time for which a service remains down during a particular period of time
Based on their contribution under SRS, a weight age is given to all the Service Quality factors and as a final point aggregated to compute ranking score (R) of a service is given by:
$R = {}^{n}\sum_{i=1} wt_n \phi_n \leqslant 1$
The Service Monitor at each vm where the service is deployed sends performance information to the Service Performance Monitoring module in the Services Cloud every five minutes. Service Analytics uses this information to compute the Ranking Scores (R) for each service instance. By default equal weights for the five service quality parameters are used in the computation of ranks, although individual service consumers can define their own weights to derive customized service rankings as per requirement. After computing the rank, a service list is presented to the user and final negotiation process is initiated. Service ranks can potentially be recomputed every five minutes when new data comes in from the service monitor. The Service Consumer is notified if the service ranks changes or a service quality parameter changes significantly since the last ranking cycle for taking suitable action including possible selection of a new service. Fig 3 presents an indicative snapshot of sample data and computed ranks for different service instances deployed at different CSPs/locations.

	Date:10:15:2013 Time: 01:00:00 – 01:05:00						Date:10:15:2013 Time:01:06:00 - 01:10:00						Date:10:15:2013 Time:01:11:00-01:15:00					
	Individual Score Weight (*Wt*) = 0.20					Total Score	Individual Score Weight (*Wt*) = 0.20					Total Score	Individual Score Weight (*Wt*) = 0.20					Total Score
	$\phi_1.Wt$	$\phi_2.Wt$	$\phi_3.Wt$	$\phi_4.Wt$	$\phi_5.Wt$	$R= {}^{5}\sum_{i=1} wt\,\phi_n$	$\phi_1.Wt$	$\phi_2.Wt$	$\phi_3.Wt$	$\phi_4.Wt$	$\phi_5.Wt$	$R= {}^{5}\sum_{i=1} wt\,\phi_n$	$\phi_1.Wt$	$\phi_2.Wt$	$\phi_3.Wt$	$\phi_4.Wt$	$\phi_5.Wt$	$R= {}^{5}\sum_{i=1} wt\,\phi_n$
S1.L3	0.08	0.12	0.134	0.108	0.174	0.40	0.08	0.12	0.134	0.108	0.174	0.40	0.08	0.12	0.134	0.108	0.174	0.40
S1.L2	0.09	0.09	0.09	0.112	0.068	0.45	0.09	0.09	0.09	0.112	0.068	0.45	0.09	0.09	0.09	0.112	0.068	0.45
S1.L1	0.174	0.108	0.086	0.068	0.134	0.57	0.174	0.108	0.086	0.068	0.134	0.57	0.174	0.108	0.086	0.068	0.134	0.57
S1.L4	0.174	0.108	0.086	0.068	0.134	0.59	0.174	0.108	0.086	0.068	0.134	0.59	0.174	0.108	0.086	0.068	0.134	0.59
S1.L6	0.45	0.65	0.67	0.112	0.87	0.63	0.45	0.65	0.67	0.112	0.87	0.63	0.45	0.65	0.67	0.112	0.87	0.63
S1.L5	0.10	0.07	0.10	0.112	0.068	0.66	0.10	0.07	0.10	0.112	0.068	0.66	0.10	0.07	0.10	0.112	0.068	0.66

Fig 3: Indicative sample snap shot of hourly score on the basis of various parameters for service instances across CSPs.

5. EXPERIMENTAL SETUP AND RESULTS

We collected real-world data pertaining to a sample deployed service across three popular CSPs - Amazon EC2 [AHP], Windows Azure [WS], and GoGrid [GG2]. The experimental data served as a basis for designing our simulator built on top of Cloudsim [R. Buyya]. The real-world test environment had a total of 3 CSPs, 6 data centers at 6 different physical locations and 10 service instances deployed at various CSPs locations. We evaluated a Photo Storage service, which is implemented on each CSPs datacenter location. We further consider only one type of virtual machine instance in different CSPs since in [Lucas-Simarro et al. [2013]] authors have concluded that both medium and large vm instances gets overloaded with the similar level of concurrent requests. Therefore in order to save experimental cost without compromising on service performance measurements we use instances shown in Table 1 for service deployment. We replace actual CSP names with "CSP_1, CSP_2 and CSP_3" and replace actual data-center locations with generic locations "L_1, L_2, L_3, L_4, L_5 and L_6" .

Table 1: Virtual Machine configuration

OS	CPU	Cores	Memory(MB)	
Linux/ubuntu	AMD Opteron 2 GHz	1	1024	
Windows Server 2012	Intel Xeon E52670	1	1024	

We used HTTP HEAD Requests using curl for latency measurements instead of ICMP ping command because MS Azure has barred incoming/outgoing ping requests [OPA]. We take measurements for every 5 minute interval and compute averages for each parameter every 1 hour. Intuitively network latency is dependent on the geographical locations of the user with respect to the data center, but routing inefficiencies, time zone differences and usage of multiple data centers in an intercloud environment can lead to counter-intuitive results. The results obtained in the test setup are shown in Fig 4 which concurs with those obtained in [Wu and Madhyastha [2013]]. It can be seen that the average percentage variation for one location is up to 18% i.e. latency varies significantly over 24 hours for the same location which implies that any QoS-complaint service deployment scheme requires factor these latency variations to effectively meet defined SLAs

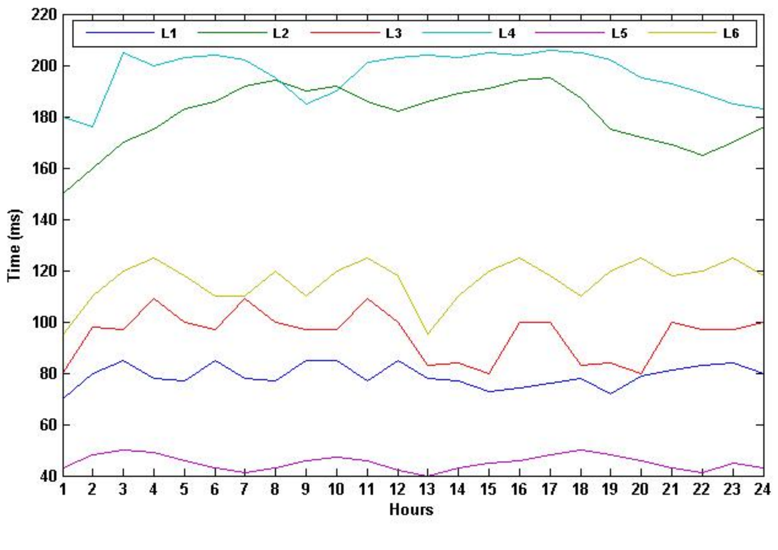

Fig 4: Observed latency for different service instances over a 24 hour period.

Further we use Manage Engines Application Manager [ME] to measure service performance in a virtualized environment. We also used httperf [HPL] as our benchmark to generate the workload for service instances deployed in different locations. Since our service is web-based, therefore detailed information on the number of connections, rate of connections, request size, request rate, reply time/size and rate etc. is required. We perform our benchmark tests on each deployed service instance in our test environment.

We measure the variations in the number of requests processed per minute and results are depicted in Fig 5. The number of requests processed per minute is dependent on the latency between the user and location of the service and the variations of up to 60% are observed. Here the location of the user remains the same. In the real world varied users locations can result in greater variations in service instances performance. We use the photo-storage service to look-up and download a 5KB file in all these tests.

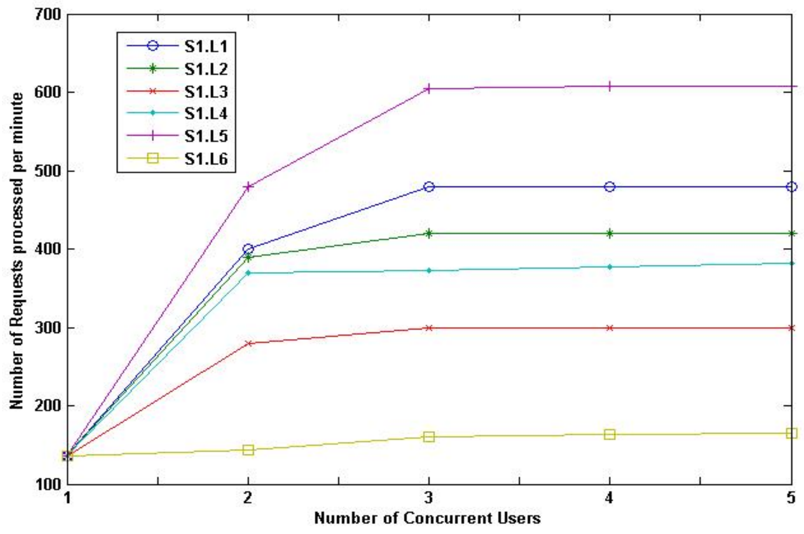

Figure 5: Observed Number of Requests processed/minute by service instances deployed in different locations

This test setup served the following purpose:

—Benchmarking service behavior with varying workloads in a real world scenario.

—Provide strong basis for the simulation of service behavior in the proposed Services Cloud framework

5.1 Simulation Results

To validate the proposed framework we used cloudsim [R. Buyya] works on real-world data obtained in Section 5 as its base input. We have built the service ranking logic on top of base classes and use the results obtained from service ranking as an input to route service requests to cloudlets in different datacenters. We conducted experiments to assess the impact of our scheme on both the Service Providers and Service Consumers We consider three CSPs with two data centers each for a total of 6 data centers. Each datacenter is located in a different geographic location. A total of 25 service instances with varied deployment strategies and 100 service consumers are considered for simulation purposes. The service consumers are located in "20" different geographical locations including the six locations of the data centers with different latencies. Each CSP follows different pricing policies for resource usage. Further, we assume that each vm can host one service instance with different processing time, reliability and availability and the vm cost per hour varies from 0.10$ -0.50$.

5.1.1 *Evaluating geographical implications for service usage.* The aim of this experiment is to assess the impact of service location on the response time. We measured the average response time for six instances of the same service deployed in each of the 6 data centers. In Figure 6 we can observe that the average response time obtained by different instances of the same service to download a 5KB image file for increasing number of user requests. We can see that the same service is responding differently if deployed in different locations and variations of up to 600% are observed in the response times. The major component in this variation is the actual latency between the service consumer and the service location, while variations due to differences in the processing capabilities of vm's belonging to different CSPs is only upto 10% . This experiment results agree with the conclusion mentioned in [Wu and Madhyastha [2013]] which highlighted the latency factor for effective service response.

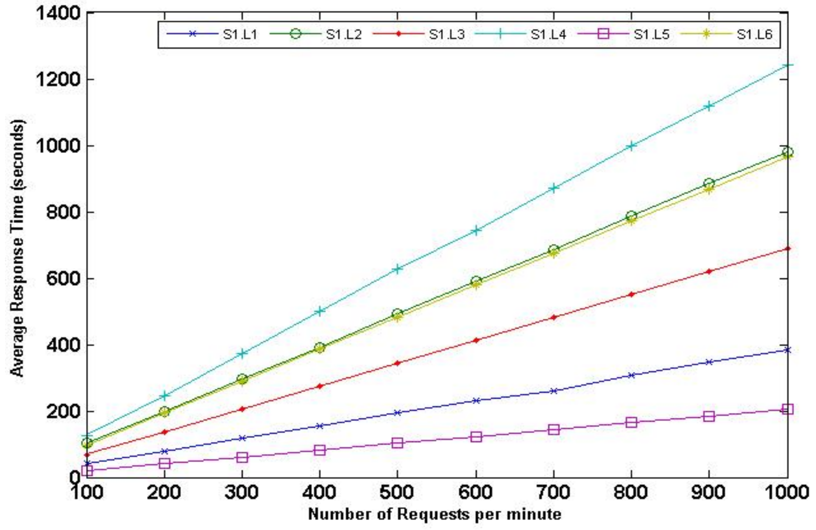

Fig 6: Variations in average file download time from 6 different geographical locations (same photo storage service).

5.1.2 *Evaluating Deployment Scenarios.* In this experiment we evaluated different deployment scenarios as discussed below and measure the impact on the observed average response times, requests drops and profits for increasing number of requests.

—**Single Instance Single Location (SISL):** It represents a service which is deployed in only one location and has got only single service instance to serve users requests.

—**Multiple Instances Single Location (MISL):** Same service and associated components replicated at single CSP and same data center (physical location). In this scheme all the requests comes to a single instance and when this instance get overloaded another instance is created on the same physical location to handle users requests.

—**Multiple Instances Multiple Locations (MIML):** Multiple instances of service components (distributed replicated services) deployed at different data centers (physical locations) of 1 or more CSPs. In this scheme the requests are distributed across different instances in inter-cloud environment. Different strategies of load balancing can be applied to serve users requests due to huge availability of service instances. For example consider a scheme in which load balancing is done to exploit geographical proximity. However, initially this type of deployment scenario is costly as compared to SISL.

Here we use the same configuration service instance, but deploy them in different strategies (SISL, MISL and MIML). In the cases of MISL and MIML three service instances are used to cater to users request. In Figure 7 shows the cumulative distribution function (CDF) of Average Response Time over the file size 5 KB that were generated during the experiment as results. It has been observed that in the case of MISL and MIML the response time has been decreased by 62% and 78.75% respectively as compared to SISL. This is obviously due to SIML and MIML having more service instances and MIML is exploiting geographically proximity.

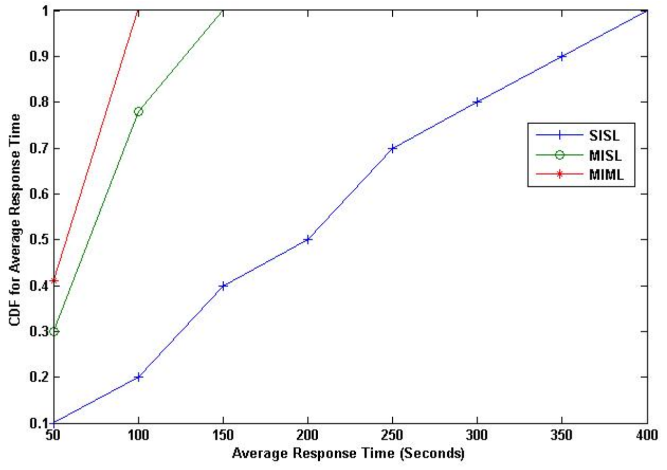

Fig 7: CDF for average response time obtained for different service deployment strategies

The processing time for a service request is dependent upon VM's operating system/hypervisor, hardware configuration of server, scheduling logic, web-server performance, service queue length etc., but shows a maximum variation of 10% between CSPs. In terms of latency since SISL and MISL are located at same location they offer almost same latency for each service instance but in case of MIML deployment the average observed latency is reduced significantly as service instances are deployed at different geographical locations to serve scattered user requests more efficiently. Therefore the main reason behind the performance degradation of MISL as compared to MIML is its static service location. This results in 37% lower average response time for a large number of geographically dispersed user requests.

We assume the maximum waiting time for the response is 100 seconds before the request is considered dropped, Therefore the request drops for increasing number of user requests were measured for each of the service deployment scenarios. Results are displayed in Figure 8. As expected, SISL suffers from high request drop due to its single instance getting overwhelmed sooner followed by MISL and MIML. MIML due to its lower overall response time and latency is able to service approximately 22% more requests compared to MISL without requests getting dropped.

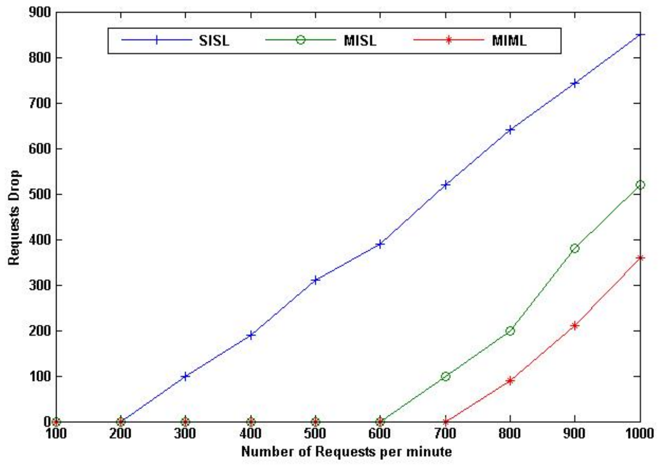

Fig 8: Number of request drops per minute for a single service under different deployment scenarios

It is noticeable that for up to 200 requests per minute, SISL, MISL and MIML have negligible request drops but beyond it SISL drops nearly 90% of received requests due to increasingly longer wait time. Further MISL with its 3 service instances starts dropping requests at close to 600 requests/minute but MIML remains stable till 700 requests per minute.

Moreover, the service deployment scenarios have a direct impact on the overall revenue generated for the SP. For up to 200 requests/minute, SISL is the most cost-effective deployment strategy, but when request/minute goes beyond 200 requests/minute SISL starts dropping requests. The impact on overall profit under various scenarios is depicted in Table 2. The potential revenue loss for a SP comprises of a) SLA violations and b) request drops since both these cases result in a penalty [Lee and Snavely [2006]][Emeakaroha et al. [2012]].

Table 2 shows that SISL is cost effective as compared to MISL and MIML when the frequency of request is small (in our case up to 200 requests/minute) but when the frequency of requests increase MIML and MISL are naturally more effective as they provide more service instances and scale better. Therefore more the number of service instances lesser the response time since more instances are available to handle the requests. Further, with increase in the number of service instances in intercloud environment the profitability can be increased by a) exploiting lower latencies through geographical proximity b) selecting the most cost-effective CSPs. We can also notice that with more number of service instances, SLA violations decreases significantly. Further, auto-scaling by the Services Cloud provides a practical mechanism to increase service instances to handle peak-load and reduce service instances proportionately as requests drop while adhering to SLA agreements. Thus, optimal deployment from the SP perspective is ensured. Therefore the shifting from SISL to MISL or MIML is dynamic and depends upon the frequency and geographical origin of requests.

It was observed that MIML performance is up to 40% better when requests are from geographically diverse origins and request loads are high. MIML can thus exploit time-zone differences and benefit from heavily discounted non-peak hour prices for compute resources at different CSPs.

Table 2: Profit projections for a SP for varying number of requests/minute

Case 1: For 100 requests/minute		
Deployment Scenario	SLA violations	Average Profit ($/hour) = (Total fee earned - Expenditure of deployment of servi ce instance - Penalties)
SISL	0	59.9
MISL	0	59.7
MIML	0	59.75
Case 2: For 500 requests/minute		
Deployment Scenario	SLA violations	Average Profit ($/hour) = (Total Fee earned - Expenditure of deployment of service instance - Penalties)
SISL	30%	263.7
MISL	0	299.7
MIML	0	299.75
Case 3: For 1000 requests/minute		
Deployment Scenario	SLA violations	Average Profit = (Total Fee earned- Expenditure of deployment of service instance - Penalties)
SISL	80%	257.5
MISL	33%	551.7
MIML	26%	563.75

5.1.3 *Evaluation of Scheduling Schemes.* In this experiment we compare the SRS scheme with a hybrid service selection (HSS) scheme which attempts to minimize response time while reducing service consumption costs. It is similar to the scheme proposed in [Lucas-Simarro et al. [2013]] in which broker focused on performance optimization with a cost constraint. This policy takes care of both cost and workload of a service. It uses cost as an upper bound and finds out the least loaded service. We have compared the performance of the HSS policy with our SRS policy on several parameters and evaluated the benefits of SRS in optimizing end-user service selection. Further we use MIML deployment strategy for both the SRS and HSS and observe the overall response time and SLA violations. For this we deploy six service instances across different CSPs with varying cost and workload characteristics. The service instances are also assigned values for availability, reputation and reliability. SRS is clearly the more advanced selection policy since it takes into account the availability, reliability and reputation of a service instance while trying to optimize communication and resource usage costs. In our tests HSS selected services with low availability for 7% of the requests, services with low reliability for 6% of the requests and services with low reputation for 10% of the cases. This result in significantly higher SLA violations compared to SRS selection policy.

We observe in the Fig 9 that poor selection choices by the HSS result in the higher response times compared to SRS to the tune of 17% on average. SRS makes more credible choices based on availability, reliability and reputation of service instances.

Further, service selection schemes can also be based on:

(1) Cost based service selection (CSS): In this type of selection policy scheduler looks out for the cheapest service offer (in case of similar competing services). It selects the least cost service and allocates it to the user. This kind of service selection is best fit for the users who are not performance oriented and want to pay less. Here the probability of response-time based SLA- violations is very high.

(2) Work load based service selection (WSS): It monitors the work load on each service deployed in datacenter and find out the least loaded service with less service requests pending. This type of policy is optimal when performance is paramount but very prone to cost based- SLA violations.

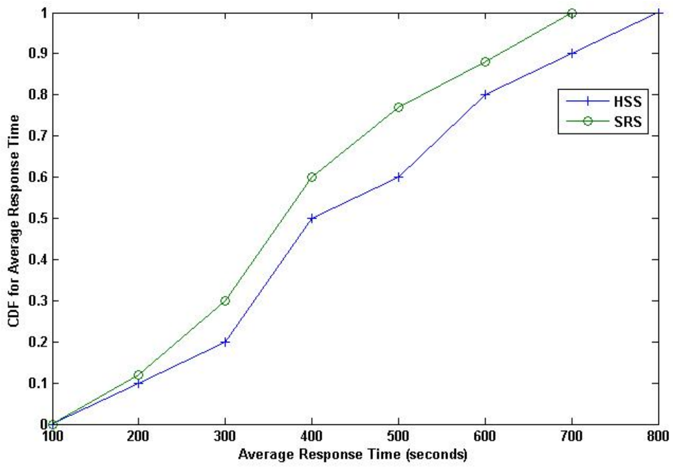

Fig 9: CDF for average response time for HSS and SRS service selection policies.

Therefore in Table 3(a) and Table 3(b) we compare HSS and SRS on cost optimization achieved and SLA violations observed with CSS and WSS selection policies as the base. SRS outperforms HSS due to: a) qualitatively better service selection and b) geographically-aware auto-scaling

Table 3(a):Optimization achieved by HSS and SRS over CSS service selection policy		
Parameters	**HSS**	**SRS**
SLA-violations optimization	31 %	58%
Cost optimization	9%	22 %

Table 3(b):Optimization achieved by HSS and SRS over WSS service selection policy		
Parameters	**HSS**	**SRS**
SLA-violations Optimization	22 %	47%
Cost optimization	19 %	34 %

5.1.4 *Service Instance Transition Behavior of SRS.* In this experiment we tracked service instance transitions (Service S1 deployed at locations L1 through L6) for varying number of user requests/hour as shown in Table 4. SRS selects the appropriate service instance combinations while optimizing the service hosting cost for the Service providers. We begin with one service instance at each location. At hour 1, S1.L6 is sufficient to handle up to 6200 request/hour with the associated hosting cost of 0.20 $ /hour. Between hours 2 to 4 due to flash crowd scenario (peak load) SRS performs service replication at L6 and L2 choosing a combination of service instances that lowers the hosting cost for the service provider in each case. The replicated services are also automatically decommissioned when not required.

Table 4: Service instance transitions in response to dynamic user requests.			
Hour	**Requests/hour**	**Service Instances**	**Total Cost/hour**
1	6000	S1.L6	0.20
2	10000	S1.L6+ S1.L6	0.40
3	17000	S1.L5+ S1.L2+ S1.L2	0.65
4	22540	S1.L6+S1.L5+ S1.L2+ S1.L2+ S1.L2	0.95
5	12000	S1.L6+ S1.L6	0.40
6	4000	S1.L2	0.10
7	3950	S1.L2	0.10
8	3900	S1.L2	0.10
9	4000	S1.L2	0.10
10	4799	S1.L2	0.10

11	4045	S1.L2	0.10
12	4000	S1.L2	0.10
13	4350	S1.L2	0.10
14	4298	S1.L2	0.10
15	4220	S1.L2	0.10
16	5700	S1.L6	0.20
17	6000	S1.L6	0.20
18	3900	S1.L6	0.20
19	5800	S1.L6	0.20
20	10000	S1.L6+ S1.L6	0.40
21	11000	S1.L6+ S1.L6	0.40
22	11098	S1.L6+ S1.L6	0.40
23	10086	S1.L6+ S1.L6	0.40
24	10035	S1.L6+ S1.L6	0.40

5.1.5 *Computation Time for SRS.* In this experiment computation time for SRS strategy is evaluated. The SRS strategy involves on-demand ranking of services based on customized weights provided by the end users which facilitates fine-grained control over service selection and consumption. This approach therefore involves computing service ranks in the context of the user-defined weights and incurs additional computational overheads compared to schemes which use a static weighted formula for determining service ranks. This is because the service ranks need to be recomputed for each user. Fig 10 provides the average processing time for computing service ranks for up to 500 service instances for a single user. The results show that with the increase in requests the computation time is growing linearly. This computation however needs to be performed only once before the service consumption phase for each user session.

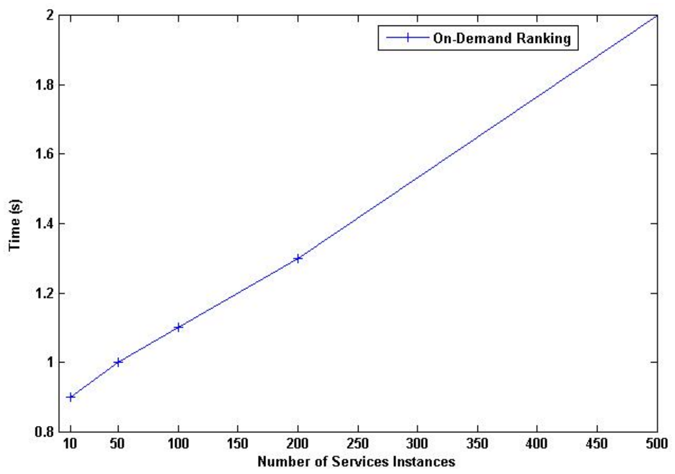

Fig 10: Average processing time for on-demand ranking of services instances

6. CONCLUSION AND FUTURE WORK

This paper proposes a Services Cloud which is a step towards a Unified Services Management Framework for an intercloud environment which facilitates a) optimal service deployment and geographically-aware auto-scaling and b) optimal service selection and consumption by the end-user in a seamless manner. We evaluated the effectiveness of the proposed framework using a custom simulator based on real world service performance measurements across different CSPs. The results indicate that from a service provider perspective achieving high performance at reasonable cost is dependent on the service deployment scheme. It was further observed that mere load-based auto-scaling is not effective to deal with flash crowd scenario but instead geographically-aware auto-scaling to exploit latency benefits in an intercloud environment is more efficient for global services. Further, a customized ranking mechanism for service consumers allows greater optimization at an individual level rather than with a one-size-fits-all approach.

We believe that the realization of a Unified Services Management Framework for an intercloud environment will result in enhanced capability to deliver services to end users in a customized manner while maximizing service provider revenue at the same time. This contributes to a more efficient management of cloud services and also reduces cost for users as well as service providers through comprehensive measurement of services performance and their ranking. Hence, the present work has potentially far-reaching consequences on the evolution of intercloud services.

Future work shall involve more real-world testing for different service types and categories, service deployment scenarios and varying user requirements over sustained periods to create formal benchmarks for intercloud services. Integrating the SC logic with some real-world cross-cloud brokers is also on the anvil. We believe that in the times to come planetary-scale services catering to geographically dispersed users and deployed in an intercloud environment shall become the norm and more research efforts shall be required to overcome the inherent complexities involved in a collaborative global system such as the proposed Services Cloud.

References

(ahp)amazon ec2 home page. http://aws.amazon.com/ec2.

(as)about smi. http://www.cloudcommons.com/about-smi.

(ccb)cloud computing brokers: a resource guide. http://www.datacenterknowledge.com/archives/2010/01/22/cloudcomputingbrokers-a-resource-guide.

(chp)cohesiveft home page. http://www.cohesiveft.com.

(cm)cloud monitoring. http://www.rackspace.com/cloud/monitoring.

(dc)deltacloud. http://deltacloud.apache.org/index.html.

(gca)google cloud app engine. https://cloud.google.com/products/app-engine.

(gg)gogrid. http://www.gogrid.com.

(hpl)hpl-hp http://www.hpl.hp.com/research/linux/httperf.

(ics)inter cloud service. https://www.tmforum.org/InterCloudService/8480/home.html.

(me)manage engine http://www.manageengine.com/products/applications_manager.

(msp)managed services paradigm. http://www.sbtpartners.com/_etc/managed_services_paradigm.pdf.

(opa)outbound ping on azure vm. http://social.msdn.microsoft.com/Forums/windowsazure/en-US/e9e53e84-a978-46f5-a657-f31da7e4bbe1/icmp-outbound-ping-on-azure-vm?forum=WAVirtualMachinesforWindows. Virtual Machines for Windows.

(rhp)rightscale home page. http://www.rightscale.com.

(spf)services protocols formats. https://marinemetadata.org/conventions/services-protocols-formats.

(tf)tosca-faq. https://www.oasis-open.org/committees/tosca/faq.php.

Wsdl. http://www.w3.org/TR/wsdl.

(ws)windowsazure. http://www.windowsazure.com/en-us.

(zhp)imory home page. http://www.zimory.com.

Calheiros, R. N., Toosi, A. N., Vecchiola, C., and Buyya, R. 2012. A coordinator for scaling elastic applications across multiple clouds. *Future Generation Computer Systems 28,* 8 (oct), 1350–1362.

Emeakaroha, V. C., Ferreto, T. C., Netto, M. A., Brandic, I., and De Rose, C. A. 2012. Casvid: Application level monitoring for sla violation detection in clouds. In *2012 IEEE 36th Annual Computer Software and Applications Conference*. IEEE, 499–508.

Garg, S. K., Versteeg, S., and Buyya, R. 2013. A framework for ranking of cloud computing services. *Future Generation Computer Systems 29,* 4 (jun), 1012–1023.

Gupta, A., Kapoor, L., and Wattal, M. 2011. C2c (cloud-to-cloud): An ecosystem of cloud service providers for dynamic resource provisioning. In *Advances in Computing and Communications*. Springer Berlin Heidelberg, 501–510.

Itani, W., Ghali, C., Bassil, R., Kayssi, A., and Chehab, A. 2014. ServBGP: BGP-inspired autonomic service routing for multi-provider collaborative architectures in the cloud. *Future Generation Computer Systems 32*, 99–117.

Krintz, C. 2013. The AppScale cloud platform: Enabling portable, scalable web application deployment. *IEEE Internet Computing 17,* 2 (mar), 72–75.

Lee, C. B. and Snavely, A. 2006. On the userscheduler dialogue: Studies of user-provided runtime estimates and utility functions. *The International Journal of High Performance Computing Applications 20,* 4, 495–506.

Lucas-Simarro, J. L., Moreno-Vozmediano, R., Montero, R. S., and Llorente, I. M. 2013. Scheduling strategies for optimal service deployment across multiple clouds. *Future Generation Computer Systems 29,* 6 (aug), 1431–1441.

McClusky, E. and Mitra, S. 2004. *Computer Science Handbook*. Vol. 2nd. CRC Press, Chapter Fault Tolerance.

Petcu, D., Macariu, G., Panica, S., and Crăciun, C. 2013. Portable cloud applications—from theory to practice. *Future Generation Computer Systems 29,* 6 (aug), 1417–1430.

R. Buyya, R. N. Calheiros, A. B. S. G. Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services. `http://www.cloudbus.org`. The cloud computing and distributed systems laboratory, University of Melbourne.

Wishart, R., Robinson, R., Indulska, J., and Jøsang, A. 2005. Superstringrep: reputation-enhanced service discovery. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., 49–57.

Wu, Z. and Madhyastha, H. V. 2013. Understanding the latency benefits of multi-cloud webservice deployments. *ACM SIGCOMM Computer Communication Review 43,* 2 (apr), 13–20.

Lohit Kapoor received the B.E. degree in Information Technology, M.Tech Degree in Information Communication and the Ph.D. degree in Computer Science and Engineering (Cloud Computing) from Thapar University, Patiala, Punjab. During his Doctorate, he has published a number of papers focused on Cloud Computing and Machine Learning in various reputed journals. His research interests cover the Cloud Computing, Data Science, Machine Learning and Artificial Intelligence.

Dr. Seema Bawa holds M.Tech (Computer Science) degree form IIT Khargpur and Ph.D. from Thapar Institute of Engineering and Technology, Patiala. She is currently Professor, Computer Science and Engineering and Dean (Student Affairs) at Thapar University, Patiala since September 2010. As Dean (Student Affairs) she has made students excel in diverse skills and areas with determination and conviction. She has demon strated wonderful managerial skills by heading the department for more than six years.
The department has grown in all dimensions including academics, research, man power development and finances. Her areas of research interests include Parallel, Distributed Grid and Cloud Computing, VLSI Testing, Energy aware computing and Cultural Computing. Dr. Bawa has rich teaching, research and industry experience. She has worked as Software Engineer, Project Leader and Project Manager, in software industry for more than five years before joining Thapar University. She has been Coordinator of two national level research and development projects sponsored by Ministry of Information and Communication Technology. She is the author/co-author 111 research publications in technical journals and conferences of international repute. She has served as Advisor / Track chair for various national and international conferences. She has supervised eight Ph.D. and forty four M.E theses so far. Prof. Bawa is an active member of IEEE, ACM, Computer Society of India, and VLSI Society of India. She has been rendering her services across the globe as an editor and reviewer of various reputed journals of these societies.

Prof. Ankur Gupta is the Director at the Model Institute of Engineering and Technology, Jammu, India, besides being a Professor in the Department of Computer Science and Engineering. Prior to joining academia, he worked as a Technical Team Lead at Hewlett Packard, developing software in the network management and e-Commerce domains. He obtained B.E (Hons) Computer Science and MS Software Systems degrees from BITS, Pilani and his PhD from the National Institute of Technology in India. His main areas of interest include peer-to-peer networks, network management, software engineering and cloud computing. He has published over 65 peer-reviewed papers in reputed international journals and conferences and is a recipient of the AICTE's (All India Council for Technical Education) Career Award for Young Teachers. He has 17 patents pending in diverse technical domains and is a senior member of both the IEEE and ACM.