Improving the Reliability and Availability of IaaS Services in Hybrid Clouds

†Bernady Apduhan, ‡ Muhammad Younas, † Toshihiro Uchibayashi

† Faculty of Information Science, Kyushu Sangyo University, Japan

{bob, uchibayashi}@is.kyusan-u.ac.jp

‡ Department of Computing and Communication Technologies

Oxford Brookes University, Oxford, United Kingdom

{m.younas@brookes.ac.jp}

ABSTRACT

This paper investigates into IaaS service provisioning in hybrid cloud which comprises private and public clouds. It proposes a hybrid cloud framework in order to improve reliability and availability of IaaS services by taking into account alternative services which are available through public clouds. However, provisioning of alternative services in hybrid cloud involves complex processing, intelligent decision making and reliability and consistency issues. In the proposed framework, we develop an agent-based system using cloud ontology in order to identify and rank alternative cloud services which users can acquire in the event of failures or unavailability of desired services. The proposed framework also exploits transactional techniques in order to ensure the reliability and consistency of the service acquisition process. The proposed framework is evaluated through various experiments which show that it improves service availability and reliability in hybrid cloud.

Keywords: component; Hybrid cloud; IaaS; availability; reliability

1 Introduction

In hybrid cloud environment, organizations and companies can provide some of its services internally using private cloud, while other services are provided externally using public cloud [1]. For instance, mission critical applications and data can be deployed at private cloud while other part of applications can be delegated to public cloud. While hybrid clouds provide immense benefits and promising returns, they are still vulnerable to various kinds of failures which can be attributed to the unavailabil-

adfa, p. 1, 2011.

ity of required services, complexity of the underlying cloud infrastructure, large-scale, and widely distributed computing resources.

In this paper we propose a hybrid cloud framework in order to improve reliability and availability of IaaS (Infrastructure-as-a-Service) services (e.g., CPU, disk storage, memory, etc.) by taking into account alternative services which are available through public clouds. However, the process of provisioning and acquisition of alternative services in hybrid cloud is not trivial. First, it involves complex processing and intelligent decision making in order to decide on which alternative services should be provided to cloud users in the case of unavailability of desired services. Second, the required services should be acquired in a reliable way such that they are consistent and correct even in the event of failures such as system failures or network communication failures.

In the proposed framework, we develop an agent-based system using cloud ontology [4] in order to identify and rank alternative cloud services which users can acquire in the event of failures or unavailability of desired services. It includes a broker server within the private cloud that collects information about availability and status of services from different public cloud providers. Individual agents are stationed on public clouds that monitor and transmit (in real time) the availability and service status changes to the broker server. Using a heuristic algorithm the broker server ranks available services at public cloud in an order that best match specification of user's requests. Once services are ranked the broker server then starts acquiring the services. It exploits transactional techniques in service acquisition in order to maintain the correctness and consistency of user's requests and the required services. Transactions have been employed in various cloud services [5]. However, they have not been exploited in the acquisition of services in hybrid cloud environment. We believe that employing transactions in hybrid cloud service acquisition can guarantee correctness and consistency of applications and services despite failures of computing nodes, or network failures.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents architecture and an example scenario of IaaS service provisioning in a hybrid cloud. Section 4 describes the selection and ranking process of IaaS services. Section 5 presents service acquisition. Section 6 presents empirical evaluation and results. Section 7 describes the concluding remarks.

2 Related work

The work in [2] introduces an agent-based cloud service search engine for cloud resource management. However, SLA and network latency were not considered. Authors in [3] identified the characteristics of IaaS and used the analytic hierarchy process (AHP) as the multi-criteria decision-making technique to compare IaaS providers. In our case, we used ontology and semantics to find the provider services that best-fit the user requirements. While we share similar goals with the above efforts, we focus on the infrastructure level and used ontology and agents for discovery and selection of IaaS services to provide resiliency support in hybrid cloud computing. Our

work does not intend to replace existing or classic cloud computing resiliency techniques, but can be considered as a complement. Authors in [5] study various alternative architectures of cloud computing for transaction processing in web and database applications. Based on alternative architectures, this work investigates into performance and cost of running such applications on different commercial cloud services such as Google and Amazon AWS. The findings are that all major cloud service providers have adopted a different architecture for their cloud service provisioning. Thus depending on the workload, there are significant differences in the cost and performance of cloud services. Paper [7] proposed a scalable transaction management approach which is based on snapshot isolation. The objective is to achieve high scalability by decoupling transaction management functions from storage systems and integrating them with application-level processes. Paper [6] presents a Deuteronomy system that ensures ACID properties in data anywhere or cloud environment. This work also separates transactions from underlying data such that the former are managed by transactional component and the latter by data component. Such separation is claimed to result in improved performance while still maintaining ACID properties. The aforementioned approaches do not exploit transactions in service acquisition in hybrid cloud. Instead, they are concerned with transaction processing in cloud data management.

3 The hybrid cloud Framework

A generic architecture of the proposed hybrid cloud framework is depicted in Figure 1. It is comprised of a private cloud, broker server and a number of public clouds. Users can acquire services from a private cloud. If required services are not available from the private cloud then they can be acquired from one or more public clouds. In this paper, we focus on the IaaS services such as CPU, memory, disk space, and so on.

Consider a scenario where a highway control office wish to run a cloud application in order to collect large volume of data (e.g., cameras, sensors, traffic vehicles, etc) about current traffic conditions on roads; and process such data in order to assess the impact of various events such as traffic jams, accidents, etc. Based on the assessment and analysis of data the highway control office can then provide highway operation staff with useful information so they can respond to events in a timely manner and ensure smooth traffic flow on roads. In order to process such a huge data in a timely manner, such application will require various IaaS services such as CPUs, memory, disk space, and so on. Highway control office may use private cloud to run such an application. But if a private cloud does not provide all the required services then this application will not be able to run and will eventually fail. In this situation, hybrid cloud provides alternative provisioning of services from public clouds. That is, in the case of unavailability of some services, the highway control office can acquire services from public clouds in order to run their application.

In the proposed framework, a broker server is deployed (on the private cloud side) which facilities the alternative service provisioning. Broker server provides various functionalities which we classify in the following classes:

Service selection and ranking: As shown in Figure 1, there are various public clouds which provide similar alternative services but with varying level of quality. Thus it is important to identify (search), rank and select the alternative services. The proposed broker-server is built around multi-agents and cloud ontology in order to carry out searching, ranking and selection of services.

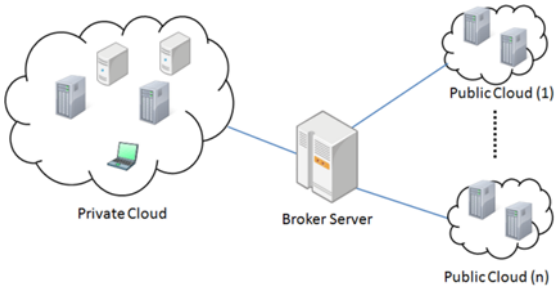

Fig. 1. Architecture of the Hybrid Cloud

Service acquisition: In order for a cloud application (e.g., highway application) to run, it is important to guarantee that all required services are acquired in a consistent and reliable manner. For a cloud application, it would not be acceptable, if a CPU service is acquired but not a memory or a disk space. The broker server uses transactions in order to avoid such situation.

4 Service Selection and Ranking

This section describes an agent-based system which is used to collect and monitor the availability and status of services provided by the selected public cloud IaaS service providers. The system comprises a broker server and various agents — i.e., each public cloud has an agent deployed on its side. Such agents send to the broker server all updated information about the user-required services of public clouds. In other words, all updated services of each cloud provider are in place in the broker server database. This helps to instantly process user's requests (or transactions) which are explained in the subsequent section. As an example we consider the following two cases:

All Services in one Public Cloud Provider:

The broker server, based on the requirements from the user application (e.g., load spikes), will conduct a first search of all member public clouds providing a first-list of public clouds which meets the minimum requirements demanded by the user application. With the first-list and using a heuristic algorithm, the broker server will conduct

an individual search of each cloud service (e.g., OS, Cost, CPU, memory, storage, SLA) and rank the providers according to the level of services availability (i.e., the more memory the provider offers, the many points it gets) that it can provide based on the user requirements.

The provider having the most number of high service availability or having the most cumulative points of its services will be ranked top and others follow accordingly. The rational idea is to have all required cloud services to be provisioned by the same provider so as to minimize communication overhead.

Some Services in other Public Clouds:

Considering that other clients or customers are also vying to avail of the same public cloud service(s), there's a possibility that one or more requested public cloud services in the top-ranked provider may not be available by the time the broker server made its selection decision. The broker server will then proceed to the next-ranked cloud provider, and so on.

However, if the set of services cannot be provisioned by one cloud provider, the broker server will then search in the next rank provider the unavailable requested service(s). If the broker server cannot find a match, the same procedure is repeated to the next lower providers until the first-list is used up. Up to this point when no match is found, the broker server repeats the same procedure from the beginning, i.e., looking up to its updated database.

Results and Observations:

Tables 1 and 2 show the first-list of public cloud providers (top five) which have met the requirements of two users, and ranked according to the procedures described above. Each cloud provider in the list offers all the required services. The tables include the SLA (System Level Agreement) guarantee offered by each public cloud provider and slightly differ from each other. These SLAs are used to break up whenever two or more providers have the same accumulated scores for ranking. If a tie score persists, then a random selection is employed.

Since some users may be requesting the same service(s) concurrently on the same provider, one or some services may not be available by the time the broker server made its final acquisition decision. The broker server will then tap the lacking services from other cloud providers.

A possible dilemma is that if one or more services will be provided by a different public cloud, such operation may incur a large communication overhead (latency) which may be detrimental to the application performance. The broker server may send a ping message to the public cloud providers to determine the latency which can be considered as a parameter to deal with so as not to compromise the service transaction process.

To this, the user will now have a full knowledge on the best-fit public cloud providers ranked accordingly. It is now up to the user to select the desired provider for optimum benefits.

5 Service Acquisition

This section describe hybrid cloud transactions which are executed as part of users' requests in order to correctly and consistently acquire the cloud services which are ranked by the agent-based system.

To run an application, user makes a request (or initiate a transaction) in order to acquire required services from a hybrid cloud, for example, 4_Core CPU, 2GB memory, 200GB disk space, Ubuntu OS. All these four services are necessary for a user to run an application. If some of these resources (e.g., disk space) are not available from one cloud provider then it can be acquired from an alternative public cloud provider.

We define hybrid cloud transaction as an execution of a user's request which can be divided into well defined units (or component transactions) that provide correctness and consistency of cloud services. A hybrid cloud transaction can be considered as a unit of different component transactions each of which is used to acquire service — e.g. one component transaction can be used to acquire CPU service, another to acquire memory, and so on.

In order to ensure that cloud resources are consistently acquired we define the following rules (properties) [8] which must be followed by hybrid cloud transactions.

Semantic atomicity requires that a hybrid cloud transaction should execute as an atomic unit of work. That is, its component transactions should be run completely such that they can acquire those services which are essential for running a user's application. If any of the essential resource (e.g., CPU) is not available then other resources (e.g. OS) are not required.

Consistency: This requires that a hybrid cloud transaction should maintain the consistency of information related to the availability of cloud resources and user's application requirements.

Durability requires that effects of a completed hybrid cloud transaction must be made permanent in their respective data sources, in order to ensure recovery in the event of failures.

Resiliency is the ability to complete a hybrid cloud transaction in spite of failures or service unavailability. Resiliency is achieved by providing alternative services from public clouds. For example, if a disk space cannot be acquired from one cloud provider then it can be acquired from another one as there exist various alternative cloud providers.

5.1 Hybrid Cloud Transaction Protocol

In the following, we describe a protocol that enforces the above rules in hybrid cloud transactions. The proposed protocol is implemented as one Hybrid Cloud Coordinator (HCC) and several Component Transaction Coordinators (CTC). HCC is deployed as part of the broker server while each CTC is deployed on individual public cloud (as in Figure 1). The main functionality of the HCC is to coordinate the execution of the overall hybrid cloud transaction in relation to the component transactions. Each CTC is responsible for coordinating its individual component transaction (e.g., acquiring CPU or disk space). HCC and each CTC maintain log files in order to record the required information about the execution of hybrid cloud transaction.

In order to acquire required resources for user's requests, HCC needs to execute hybrid cloud transaction. During the execution of hybrid cloud transaction, HCC communicates various messages with CTCc. Transaction execution is accomplished through the following steps:

1. A new hybrid cloud transaction is assigned to a HCC, which records the start of the transaction in a log file, and send 'start' message to CTCi to initiate component service transaction.

2. CTCi records the start of a component service transaction in the log file. After processing, CTC sends either a 'local success' or 'local fail' decision to HCC. 'local success' decision is made if component service transaction can acquire the required cloud resource (e.g., CPU, etc). 'local fail' decision is made if it cannot acquire the required cloud resource.

In case of 'local fail' decision, CTC records the failure of component service transaction, and declares it to be locally failed. Otherwise, CTC writes a 'local success' decision, and awaits from HCC final decision.

3. HCC receives all decisions from CTCc. If all the decisions are 'success' (i.e., all resources, CPU, disk space, memory are acquired), then HCC records 'global success' decision in the log file and inform all CTCs of the success decision. The CTCs commits, by forcibly writing the commit decision and then terminates the transaction and starts the processing of a new one.

Table 1. User1 Requirements

Table 1. User1 Requirements:	cpu	memory	storage	cost	os
	2_Core	2GB	200GB	10000JPY	Ubuntu

rank	point	provider	service	cpu(core)	memory(GB)	storage(GB)	SLA(%)	cost(JPY)	OS
1	3	Softbank_Telecom	Type_Dual	2	2	100	99	15750	WindowsSever2012 Ubuntu CentOS
2	3	NIFTY	Spec_Type6	2	2	30	99.99	25410	WindowsServer2012 Ubuntu RedHatEnterpriseLinux CentOS
3	2	GOGRID	Large	4	25	200	100	13140	Ubuntu Debian RedHatEnterpriseLinux WindowsServer2012 CentOS
4	2	NTTCommunications	Plan_v2	2	4	10	99.99	7560	WindowsSever2012 Ubuntu CentOS
5	2	IDC_Frontier	Type_M2	2	8	15	99.999	21000	WindowsServer2012 Ubuntu RedHatEnterpriseLinux CentOS

Table 2. User2 Requirements

Table 2. User 2 Requirements:	cpu	memory	storage	cost	os
	4_Core	2GB	200GB	10000JPY	Ubuntu

rank	point	provider	service	cpu(core)	memory(GB)	storage(GB)	SLA(%)	cost(JPY)	OS
1	3	GOGRID	Large	4	25	200	100	13140	Ubuntu Debian RedHatEnterpriseLinux WindowsServer2012 CentOS
2	2	Softbank_Telecom	Type_Dual	2	2	100	99	15750	WindowsSever2012 Ubuntu CentOS
3	2	NTTCommunications	Plan_v4	4	8	10	99.99	15120	WindowsSever2012 Ubuntu CentOS
4	2	IDC_Frontier	Type_L	4	4	15	99.999	29400	WindowsServer2012 Ubuntu RedHatEnterpriseLinux CentOS
5	2	NIFTY	Spec_Type3	1	2	30	99.99	18144	WindowsServer2012 Ubuntu RedHatEnterpriseLinux CentOS

4. If any ‘failed’ component service transaction is replaceable (having alternative service from public cloud), MTC initiates the alternative component service transaction and awaits the CTC’s decision regarding ‘success’ or ‘fail’ of the alternative component service. If component service transaction is not replaceable HCC records ‘global fail’, and informs all CTCs about its decision. In this case, the overall hybrid cloud transaction has failed and cloud resources cannot be acquired.

5. After receiving a global decision, CTC writes the global success of component service transaction. If component service transaction is local-success and CTC receives a 'global fail' from HCC, then CTC must cancel that service. This situation occurs when one component service transaction acquires a resource (e.g. disk space) and another does not acquire (e.g. CPU). In this case the acquired resource needs to be cancelled as user will need both resources (CPU and disk). This is the constraint set by the semantic atomicity rule (as described above).

6 Empirical Evaluation

This section describes the evaluation of the proposed framework in terms of failure resilience and communication overhead. One of the main objectives of the proposed framework is to enhance the resiliency of cloud services through alternative public cloud services. We therefore evaluate the resiliency aspect of the framework. However, such resiliency may incur performance delay mainly due to the network communication delays. We therefore evaluate the overhead caused through the alternative services provisioning from the public clouds.

A. Failure Resiliency

Our evaluation criteria for failure resiliency are based on probability theory and are simulated through a prototype tool. In order to simulate the success/failure rate of a hybrid cloud transaction, we define the following set of probabilities:

private cloud success (PriCS): This refers to the probability that user can acquire required services from a private cloud by executing a hybrid cloud transaction.

private cloud failure (PriCF): This refers to the probability that user cannot acquire required resources from a private cloud by executing a hybrid cloud transaction. In other words, if the private cloud cannot meet the minimum requirement of user's request then it is considered as a failure of private cloud. In this case, the user's requests should be sent to the broker server (as described in the subsequent section).

alternative services availability (AltSA): This refers to the probability that there exist alternative services which user can acquire from public clouds. In other words, this depicts the situation where users cannot acquire first choice services from private cloud. But they have the opportunity to acquire alternative services from public cloud. As described above, we use a heuristic algorithm to rank different public cloud providers.

public cloud success (PubCS): This refers to the probability that user can acquire required services from a public cloud (first-list – see below) by executing a hybrid cloud transaction.

public cloud failure (PubCF): This refers to the probability that user cannot acquire required resources from a public cloud by executing a hybrid cloud transaction.

total success rate (TSR): This refers to the probability that a user can acquire required services from private and one or more public clouds.

total failure rate (TFR): This refers to the probability that user cannot acquire required services from private or any of the public clouds.

Based on the above probabilities, we design various mathematical expressions in order to calculate the total success rate (TSR) and total failure rate (TFR) of a hybrid cloud transaction. TSR and TFR are calculated using the probabilities PriCS, PriCF, AltSA, PubCS and PubCF.

We consider multiple alternative cloud services from public clouds. For example, if required storage is not available from a private cloud then it can be acquired from public clouds (see Section 4).

Table 3. Total Success and Failure Rates

	Case 1	Case 2	Case 3	Case 4	Case 5
PriCS	0.9	0.8	0.7	0.6	0.5
PriCF	0.1	0.2	0.3	0.4	0.5
TFR1	0.028	0.072	0.132	0.208	0.3
TFR2	0.0208	0.0464	0.0816	0.1312	0.2
TSR1	0.972	0.928	0.868	0.792	0.7
TSR2	0.9792	0.9536	0.9184	0.8688	0.8

As shown in Table 3 we conduct various experiments to evaluate the resiliency of the proposed framework. We assume that the probability of alternative services availability (AltSA) is 0.9 — showing that there exist 90% chance of acquiring alternative services from public clouds. In each case, different values of the PriCS and PriCF probabilities are also used in conjunction with AltSA. Various situations are considered. For example, scenarios where the failure probability is high, and also scenarios where the success probability is high.

The total success and failure rates of hybrid cloud transactions are calculated in Table 3 and are graphically represented in Figures 2 and 3. The graphs clearly indicate that the proposed framework increases the resiliency of the service provisioning by successfully executing hybrid cloud transactions and acquiring the cloud services. For instance, the success rate of hybrid cloud transaction in Figure 2 is higher when there are services available from public clouds (TSR1 and TSR2). Similarly, Figure 3 shows that failure rate of TFR1 and TFR2 is lower than the PriCF when public cloud services are available.

B. Processing and Communication Delays

The improvement in failure resiliency may result in performance degradation. If one or more services are acquired from public clouds, then it may incur a communication overhead (latency) which may be detrimental to the application performance. In this section, we evaluate such overhead by taking into account some of the important parameters such as the network communication delay, the number of messages communicated between Hybrid Cloud Coordinator and several Component Transaction Coordinators (see Section 5-A), processing time of each component transaction spent

on acquiring services from public cloud, the total number of component transactions acquiring services from public cloud, and the probability that a particular service is acquired from a public cloud.

The average time taken to process a hybrid cloud transaction, T_{proc}, can be calculated as follows:

$$T_{proc} = \sum_{i=0}^{n} CTPR_i + \sum_{j=0}^{m} CTPUB_j + NET_d$$

T_{proc} is the sum of processing time of its component transactions ($CTPR_i$) acquiring resources from a private cloud plus component transactions ($CTPUB_i$) acquiring resources from a public cloud plus the network communication delay (NET_d).

Processing time of a component transaction is the time it takes to acquire a cloud service (e.g., disk storage, memory, etc).

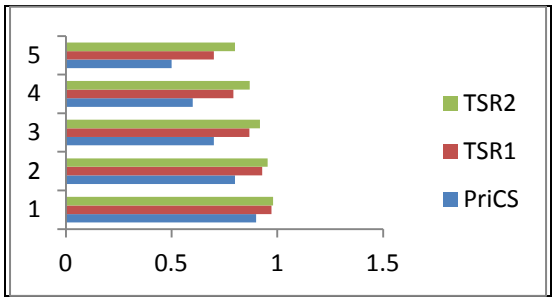

Fig. 2. Success Rate of Hybrid Cloud Transactions

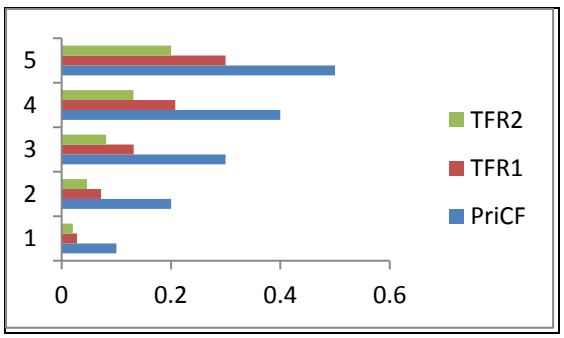

Fig. 3. Failure rate of Hybrid Cloud Transactions

NET_d is a network delay which is calculated as follows:

$$NET_d = (M_{pr} \times CD_{pr} \times N_{pr}) + ((M_{pub} \times CD_{pub} \times N_{pub}) \times P_{pub})$$

M_{pr} and M_{pub} are the number of messages communicated between hybrid cloud systems in order to respectively acquire services from private cloud and public cloud(s). CD_{pr} and CD_{pub} are the message delays for private and public clouds. N_{pr} and N_{pub} represent the number of component transactions respectively acquiring services from private cloud and public cloud. P_{pub} is the probability that one or more cloud services can be acquired from a public cloud. P_{pub} = 0, if all services can be acquired from a private cloud. In that case, there is no extra communication overhead of acquiring services from public clouds. P_{pub} = 1, if any of the service is acquired from a public cloud.

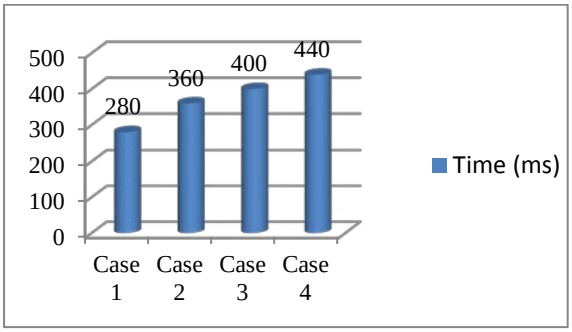

Fig. 4. Average Processing Time

We simulate an average time taken to process a hybrid cloud transaction, T_{proc}, by taking into account four different cases. We consider that as part of user request, a hybrid cloud transaction has to acquire four services by executing four component transactions.

The average time for four different cases is graphically represented in Figure 4. Case 1 represents the situation where all services are acquired from private cloud and thus average processing time, T_{proc}, is less than other cases. Case 4 represents the opposite that all services are acquired from public cloud showing steep increase in the processing time. Case 2 shows the situation where half of the services are acquired from private cloud and half from public cloud. Case 3 shows that three services are acquired from public cloud and one from private cloud.

7 Conclusion

We presented a hybrid cloud framework in order to improve reliability and availability of IaaS services through a synergetic approach of agents, ontology and transactions. The framework employed agents and ontology to gather current status of IaaS services from each member public clouds. The service information status is consolidated on a broker server which can provide, using a heuristic selection algorithm, the

first-list public cloud providers which meet the user requirements through the private cloud at runtime. Transactional techniques are then used in order to ensure the reliability and consistency of the service acquisition process. The proposed framework can be used by cloud users and providers in order to make intelligent decision in service consumption/provisioning in hybrid cloud by taking into account various crucial factors such as (i) ranking and selection of alternative cloud services from public cloud(s) (ii) enhanced failure resiliency and (iii) the performance and communication delay caused when services are acquired from private and/or public clouds.

Acknowledgment

This research was supported in part by the Japan Society for the Promotion of Science Grants-in-Aid for Scientific Research 24500100.

References

1. T. Uchibayashi, B.O. Apduhan, N. Shiratori, “An Ontology Update Mechanism in IaaS Service Discovery System”, Int. Journal of Web Information Systems, No.9(4), pp.330-343, 2013.
2. K.W. Sim “Agent-Based Cloud Computing”, IEEE Trans. on Services Computing, Vol. 5(4), pp.564-577, 2012.
3. S. Lee, K-K. Seo “A Multi-Criteria Decision-making Model for an IaaS Provider Selection Problem”, Int. Journal of Advancements in Computing Technology, Vol. 5(12), 2013.
4. D. Androcec, N. Vrcek, J. Seva “Cloud Computing Ontologies: A Systematic Overview”, Proc. of the 3rd Int. Conf. on Models and Ontology-based Design of Protocols, Architectures and Services, pp. 9-14, 2012.
5. D. Kossmann, T. Kraska, and S. Loesing, “An Evaluation of Alternative Architectures for Transaction Processing in the Cloud”. Proc. of SIGMOD, pp. 579-590, 2010.
6. J. J. Levandoski, D. B. Lomet, M. F. Mokbel, and K. Zhao, “Deuteronomy: Transaction Support for Cloud Data” 5th Biennial Conference on Innovative Data Systems Research January 9-12, Asilomar, California, USA, pp. 123–133, 2011.
7. V. Padhye and A. Tripathi, “Scalable Transaction Management with Snapshot Isolation on Cloud Data Management Systems,” Proc. of IEEE 5th Intl. Conference on Cloud Computing, pp. 542-549, 2012.
8. M. Younas, B. Eagelstone, R. Holton “A Review of Multidatabase Transactions on the Web: From the ACID to the SACReD” British National Conference on Databases (BNCOD), Exeter, UK, July 3-5, Springer LNCS, pp. 140-152, 2000.