Automated Non-repudiable Cloud Resource Allocation*

Kassidy Clark, Martijn Warnier, and Frances M.T. Brazier

Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, Delft, The Netherlands
{k.p.clark,m.e.warnier,f.m.brazier}@tudelft.nl

Abstract. This paper presents an Intelligent Cloud Resource Allocation Service (ICRAS) that assists consumers with the complex task of finding the optimal configuration of Cloud resources given a consumer's specific needs. The process of selecting a CSP becomes increasingly complex as the number of Cloud Service Providers (CSP) offering similar services continues to grow. Consumers can pick and choose between CSPs based on a growing number of options, including price, Quality of Service, reputation and so forth. The advent of dynamic pricing (based on real-time availability) further increases the complexity of CSP selection. ICRAS alleviates much of this burden from the consumer by automating the processes of service discovery, evaluation, negotiation and migration. Furthermore, ICRAS monitors Service Level Agreement (SLA) compliance using non-repudiable monitoring techniques.

1 Introduction

Cloud computing [2] provides the illusion of unbounded online resources, such as cpu or storage capacity. Companies that offer these resources are referred to as Cloud Service Providers (CSP). The Cloud is sometimes also called *elastic* since customers can easily increase or decrease resource usage, such as the amount of computing power, rented from a CSP.

Similar Cloud services are offered through a number of CSPs that compete on price and service levels. Several of these CSPs also offer a whole pallet of options to their customers who can customize their own service based on metrics such as price, Quality of Service (QoS), reputation and location. Note that most of these metrics are dynamic, i.e. they change continuously. For example, some CSPs, such as Amazon Web Services spot pricing, offer dynamic pricing. This enables that the price of resources changes constantly, which reflects underlying factors, such as Cloud utilization, fluctuating energy prizes or consumer demand [3,4].

In this environment, a consumer of Cloud services faces several challenges. First, to obtain the desired initial configuration of Cloud resources, a consumer must evaluate prices and configuration options (QoS levels, location, etc.) of all available CSPs.

* This is an updated an extended version of the paper "An Intelligent Cloud Resource Allocation Service - Agent-based automated Cloud resource allocation using micro-agreements" [1] presented at the 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012).

The task of finding the ideal configuration is further complicated as more CSPs implement dynamic pricing. When a consumer chooses the configuration that is currently the most appropriate, a better (cheaper) configuration may become available soon thereafter. Therefore, a consumer must periodically reevaluate configurations at all available CSPs. If a consumer chooses to move from his or her current CSP to a different CSP with a more suitable configuration, the consumer is then faced with the challenge of migration. Due to inoperability of CSPs and the tendency towards vendor lock-in, changing CSPs is not a trivial task. Finally, once a consumer chooses a CSP, the consumer must continually monitor the service to detect any violations to the service agreement. Moreover, the consumer must also give evidence, for example in the from of an audit trail, that a violation has actually taken place.

To assist a consumer with these challenges, this paper introduces an Intelligent Cloud Resource Allocation Service (ICRAS). ICRAS supports the consumer throughout the lifecycle of a Cloud service. This includes, (1) discovering all available resource configurations, (2) choosing the desired configuration, (3) negotiating a service agreement with the CSP, (4) assisting in the migration of services between CSPs and (5) securely monitoring the service agreement for violations.

ICRAS aggregates information describing the available services from multiple CSPs, including current price, availability, Quality of Service guarantees, location and reputation. When a consumer requires resources, it contacts ICRAS with a description of the computing needs. ICRAS then matches the resource request to the most appropriate configuration of Cloud resources from the CSPs. ICRAS facilitates the negotiation of the necessary Service Level Agreements (SLA) with the CSPs on behalf of the consumer and assists in the migration process.

ICRAS then monitors the services during the lifetime of the SLA to ensure that there are no agreement violations. If violations are detected, corrective action can be taken. Service monitoring is performed using secure modules at both the consumer and provider. Further steps are taken to generate an audit log of service message. Using several cryptographic protocols, this audit log can guarantee integrity and non-repudiation of service messages.

The main contributions of this paper are an Intelligent Cloud Resource Allocation Service (ICRAS) that (1) maximizes the utility of the consumer, (2) supports the consumer throughout the lifecycle of a Cloud service, (3) utilizes micro agreements in order to quickly react to changes in the Cloud service market (e.g. a lower price from a competing CSP), and (4) provides monitoring of the SLA which results in an audit trail that provides non-repudiation and integrity.

The remainder of this paper is organized as follows. Section 2 introduces the core concepts used in automated negotiation and service monitoring. The ICRAS architecture is detailed in Section 3. The ICRAS protocol is explained with a use-case in Section 4. Section 5 gives an overview of an prototype implementation of the ICRAS framework in the AgentScape platform. In Section 6, other automated service negotiation architectures are compared. Finally, the implications of this research are discussed in Section 7 and the paper is concluded in Section 8.

2 Automated Negotiation and Monitoring

Negotiation is the process by which one or more parties, with possibly conflicting goals, together search for a mutually acceptable agreement [5]. The negotiation process consists of proposals, counter-proposals, trade-offs and concessions, as each party attempts to maximize its own utilities (e.g. outcomes). A common utility function for consumers in the context of Cloud computing is to reduce costs while achieving the desired resources and maintaining reasonable Quality of Service (QoS)

Much research has been done in recent years on the area of automating the negotiation process using intelligent software agents [6,7,5,8,9]. In this paper, an agent is be defined as a piece of software that is capable of autonomous action [10].

In the marketplace, agents represent the individual parties of a negotiation. Given a user's preferences and a negotiation strategy, agents are able to communicate with other agents to autonomously negotiate agreements. Furthermore, agents can learn from past social interactions and improve their response to changes in the environment or even take proactive measures when opportunity arises. The agent model supports message passing and autonomous decision making useful for automated negotiation.

2.1 Service Level Agreements

The product of a successful negotiation session is an agreement between the parties that stipulates the terms and conditions of the service. This agreement is referred to as Service Level Agreement (SLA). An SLA contains the names of the parties involved, the services to be provided and the QoS guarantees that apply. Several standards have been proposed for formalizing the negotiation and creation of the SLA document, including the Web Service Agreement (WSAG) [11] and Web Service Agreement Negotiation (WSAN) [12] specifications.

The WSAG specification describes the steps taken during SLA negotiation, as well as how SLAs are represented. The objects used in negotiation are 1) Templates, 2) (Counter-) offers and 3) Agreements. *Templates* are used by service providers to describe the services they offer, including specific configurations of price, QoS guarantees and so forth. These services are listed in the template with constraints such as `ExacltyOne` and `OneOrMore`. Upon request, a service provider sends his or her templates to a service consumer. Based on the templates, the consumer makes one or more *Offers*. An offer is an instantiated template. This occurs when a consumer chooses a specific configuration of services from a template along with their associated guarantees. If both parties accept an offer, an *Agreement* is created. The final agreement lists the parties involved, the exact services being provisioned and the specific guarantees (QoS) that apply. If the offer is not accepted, either a counter-offer is created with a new configuration or the negotiation session is terminated.

2.2 Micro Agreements

Agreements specify the terms and conditions of a service for a defined period of time. For instance, home-owners typically make a long-term agreement with the power company for a period of one year or more. The agreement typically stipulates that the home-owner may not migrate to another energy provider until the end of the period. This fixed pricing period benefits the provider two fold. First, it provides a reliable income source for the period. Second, it improves the accuracy of the usage prediction used for buying or generating electricity. Energy providers can make more accurate assumptions about energy consumption if their customers cannot suddenly move to a different provider.

The disadvantage of long-term agreements is that the customer cannot react to changes in the market, such as new providers or cheaper products. In practice, prices are constantly changing due to the constant balance of supply and demand. However, these changes are not immediately reflected in the price the customer pays, due to long-term agreements. Furthermore, due to fixed pricing, customers have no incentive to shape their demand to conform to supply. This results in lowered market efficiency.

An alternative to a long-term agreement is a micro-agreement. A micro-agreement is a short-term agreement with a period on the scale of seconds, hours or days. By keeping the period of fixed-pricing short, consumers are able to benefit from dynamic pricing, also referred to as real-time pricing. Using micro-agreements, consumers are able to shape their demand on an hourly basis, in response to changes in price. This approach increases market efficiency, lowers price and reduces the amount of unconsumed (e.g. wasted) resources. Dynamic pricing has been investigated in the area of energy markets with promising results [13].

From a technical perspective, short term agreements differ only slightly from classical agreements. No fundamental changes to the negotiation protocol are required. Protocols, including WSAN described above, already support renegotiation of existing agreements. A micro-agreement is just an agreement with a much shorter time-to-live (TTL). Additional resources are needed to handle the high frequency of agreement (re)negotiation, including hardware resources. For instance, if a single CSP has 100 customers with month-long agreements, that CSP needs resources (e.g. memory, CPU and so forth) to handle 100 agreement (re)negotiations per month. However, if agreements expire after 1 hour, this CSP must process approximately 100 agreement (re)negotiations every hour.

2.3 Service Monitoring

Monitoring is used to detect SLA violations when they occur and to identify the offending party, if possible. In some cases, no responsible party can be identified (e.g. force majeur). For instance, if a lightning strike disables the communication lines between a consumer and a provider, the consumer may incorrectly conclude that the provider has violated the SLA. Monitoring data can be used to show that the provider was not responsible for the violation.

A commonly used approach to monitoring is referred to as active monitoring. Active monitoring performs specific measurements at specified intervals. Active monitoring is used to monitor SLAs [14,15]. A service is monitored by periodically testing whether the terms of an SLA have been met by all parties. This may require measuring a single variable or a complex aggregation of variables. For instance, ‘Host is reachable.’ may be measured by a single request/response action. In contrast, ‘Host uptime is greater than 99%.’ is often measured by polling a host multiple times and calculating the average rate of success.

An important aspect of monitoring is safeguarding objectivity of monitoring results. A party to an agreement may have an incentive to manipulate the results to his or her advantage. For instance, an SLA may stipulate that a consumer receives financial compensation if a specified service is unreachable. Regardless of the actual status of the service, that consumer may want to manipulate monitoring results to make it appear unreachable and therefore collect financial compensation. To prevent this situation from occurring, a Trusted Third Party (TTP) is used to perform monitoring measurements [16,15]. A TTP is an independent party that can access all communication between the parties to the SLA. To prevent parties from manipulating the measurement results collected at their respective locations, a TTP install Trusted Monitoring Modules (TMM) at each partys location. The use of TMMs allows parties to have more equal access to the same QoS metrics. For instance, a consumer may not allow a provider to access sensitive client data directly. However, a TMM allows a TTP to access this data in a secure way. Thus, the provider has 'indirect' access to the data via the TTP and will be notified if it reveals any SLA violations. Which TMMs are required to access which QoS metrics depends on a specific SLA.

Active monitoring can be combined with an alternative monitoring technique, referred to in this paper as *passive monitoring* [17,18]. Passive monitoring uses cryptographic primitives to generate a secure audit log of all service messages. The cryptographic primitives offer integrity and non-repudiation of these messages between consumer and provider. If a conflict arises regarding SLA compliance, the audit log is analyzed to determine which party (if any) has violated the SLA.

3 ICRAS Architecture

The Intelligent Cloud Resource Allocation Service (ICRAS) requires an underlying architecture, consisting of three major components: 1) a consumer, 2) a CSP and 3) an ICRAS agent. These elements represent the three roles in the marketplace, which may contain multiple instances of each. Furthermore, this architecture provides the mechanisms and protocols that enable these parties to communicate with one another and autonomously negotiate micro-SLAs. SLAs are negotiated and created following the WSAN specification. This architecture is illustrated in Figure 1.

3.1 Consumer

Each consumer interacts directly with an ICRAS agent. A consumer specifies his or her requirements in an SLA offer. This document allows a consumer to specify 1) hard and 2) soft requirements, 3) priorities, 4) ranges of options, and 5) dependencies between requirements. For instance, a consumer requires 10 virtual servers with a combined CPU power of 20 GHz and a combined storage of 2 TB. Using the SLA notation, a consumer expresses that the CPU and storage requirements are strict, however, for a reduced price, the actual number or servers can change.

In addition to providing the initial resource requirements, a consumer is also responsible for updating these requirements. If resource requirements change, a consumer must inform an ICRAS agent of these changes. A change in requirements can occur

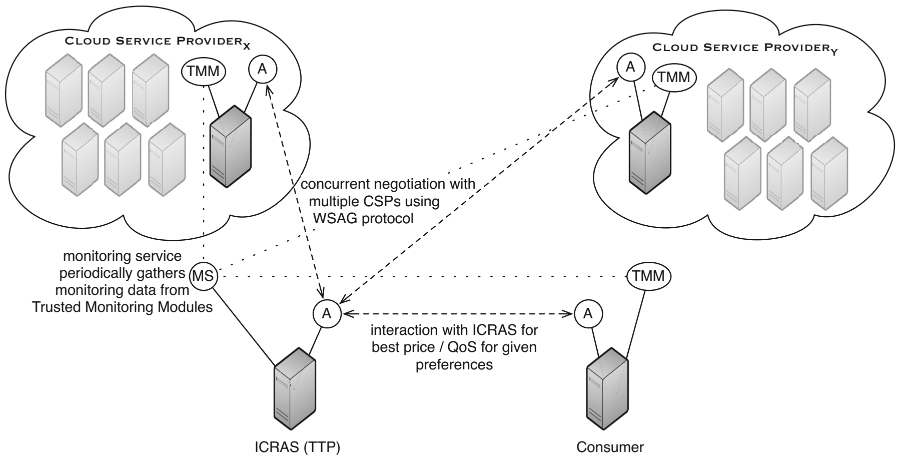
Cloud Service Provider$_X$
TMM
A
Cloud Service Provider$_Y$
A
TMM
concurrent negotiation with multiple CSPs using WSAG protocol
monitoring service periodically gathers monitoring data from Trusted Monitoring Modules
MS
TMM
A
A
interaction with ICRAS for best price / QoS for given preferences
ICRAS (TTP)
Consumer

Fig. 1. ICRAS architecture with a consumer negotiating with two competing CSPs

for several reasons. First, based on current events or past experience, a consumer can predict increases or decreases in computing needs. For instance, online retailers receive more traffic leading up to the holidays. Second, a change in business needs can prompt an immediate reconfiguration of the resource requirements. For instance, a company decides to remove some legacy applications. Finally, a company's resource requirements can change due to developments in the market, such as increased competition or lower consumer demand.

To enable such changes, a consumer monitors the level of activity on his or her Cloud resources and informs the ICRAS agent if a threshold is crossed and a new configuration is necessary.

The consumer must also host a Trusted Monitoring Module (TMM) as described in Section 2.3. This module gives the ICRAS agent, acting as the de facto Trusted Third Party (TTP), access to relevant service metrics. The ICRAS agent can thus accurately assess the user experience of the service. Passive monitoring is supported by extending the TMM to include the necessary cryptographic protocols.

3.2 CSP

To enable participation in the ICRAS architecture, a CSP must offer a compatible interface that is accessed by the ICRAS agent. This interface must support two main functions: negotiation and migration. For negotiation, a CSP must generate SLA templates. For this, a CSP requires access to internal information of its Cloud. This includes real-time pricing data, Cloud utilization and system health (QoS) information, if available. On the basis of this information a CSP generates SLA templates describing the available resources. Due to the dynamic nature of CSP resource availability and pricing, these SLA templates are updated regularly.

Upon request, the SLA templates are delivered to the ICRAS agent. When the ICRAS agent makes an offer, the CSP enters a negotiation session. The strategy that drives this negotiation is determined by the CSP negotiation policy. This policy includes functions for evaluating an offer, threshold values for acceptance or rejection of an offer and rules governing the creation of counter-offers.

To support data migration to and from its Cloud, the CSP interface must support the import and export of virtual disk images. After creating an SLA with a consumer, the CSP must support the uploading and import of the consumer's virtual disk images. Likewise, these virtual disk images are exported and downloaded upon request.

To support monitoring, the CSP must also host a Trusted Monitoring Module (TMM) that gives the ICRAS agent access to relevant service metrics, such as network latency and so forth. The TMM can also include support for the cryptographic protocols required for passive monitoring.

3.3 ICRAS Agent

This paper assumes that an ICRAS agent is maintained by an independent, trusted third party (TTP). This service has no loyalty to any particular CSP and therefore can operate fully on behalf of participating consumers. The ICRAS agent has five major responsibilities: 1) discover CSP resource offerings, 2) evaluate these offerings, 3) negotiate an SLA with a CSP on behalf of a consumer, 4) monitor the provisioning of the new Cloud resources to detect SLA violations and 5) assist in migration to the new CSP.

Discovery. The process begins when an ICRAS agent receives a resource request from a consumer. The agent then queries all CSPs for one or more SLA templates describing their available resource offerings. This process is repeated at a regular interval to discover more appropriate configurations even after an SLA has been created. Depending on a consumer's preferences, he or she is notified if a new and better suitable configuration is discovered. The consumer is then given the option to renegotiate a new SLA.

Evaluation. Once received, the agent compares the CSP templates to the consumer's request. If a CSP cannot provide any of the requested resources, this CSP is removed from consideration. The remaining templates are then evaluated and ordered using the preferences of the consumer. For instance, if a consumer specifies that price is the most important attribute, the remaining templates are arranged by price. Depending on a consumer's requirements, templates from multiple CSPs can be selected for separate resource requirements. For instance, a consumer may allow processing and storage to be handled by two separate CSPs, if this meets the price and QoS needs.

Negotiation. Once the best template has been selected, the ICRAS agent contacts the responsible CSP to begin negotiations. If multiple templates from competing CSPs are considered to be acceptable, these CSPs are contacted for simultaneous negotiations. If a negotiation session results in an offer that is acceptable by both a CSP and the ICRAS agent (according to a consumer's request), this is sent to the consumer for final approval. If acceptable, the consumer contacts the CSP directly to create a micro-SLA. A micro-SLA is used so that a consumer can migrate to a new configuration or renegotiate the current configuration if the opportunity arises.

Migration. Once a consumer decides to migrate, the consumer services are migrated to the new CSP. In the most straightforward case, migration involves stopping the cloud instances at the current CSP, converting these instances (e.g. disk images) to the format used by the new CSP, transferring them to the new CSP and starting them again. The conversion process is not necessary if CSPs adopt the same industry standard, such as the Open Virtualization Format [19].

If services cannot be stopped during migration, live migration is required. Live migration of cloud instances can be possible if both CSPs are using the same virtualization layer [20]. However, the heterogeneity of current CSPs complicates the migration process.

Monitoring. The task of the ICRAS agent does not stop after an SLA has been created and a service is being used. The ICRAS agent also assumes the role of Trusted Third Party (TTP) and monitors the service to detect SLA violations by either party. Using TMMs at each party, the ICRAS agent periodically measures service performance at both the source (CSP) and end user. The ICRAS agent uses a dedicated Monitor Service (MS) to monitor the SLA for QoS violations, such as slow network response [16]. If a violation is detected, parties are notified so corrective action can be taken.

When using passive monitoring, the ICRAS agent acts as the mediator of any conflicts that occur. As mediator, the agent requests audit logs from all parties. These logs are then analyzed to determine which, if any, party has violated the SLA. The full mediation process is explained in detail in [17].

4 ICRAS Protocol

This section gives an example scenario to demonstrate the process of ICRAS mediated negotiation. This example involves two competing CSPs, a single ICRAS agent and a single consumer. Service requests, SLA templates and offers are presented in generic format rather than their official XML format.

Step 1. A consumer requires Cloud resources. A consumer specifies these needs using an SLA offer. This request is summarized in Figure 2. In this request, a consumer indicates that it needs 10 servers with CPU power between 1.5. and 3.0 GHz, at least 2 TB of storage and at least 1 GB of traffic. Furthermore, the consumer prefers the Windows OS, requires an availability of between 95 and 100 percent and a price below 1000 Euro. This resource request is sent to the CSP .

Step 2. The ICRAS agent receives the request of the consumer and queries all participating CSPs for their SLA templates.

Step 3. Each CSP receives the query and responds by sending SLA templates that describe the current resource offering to the ICRAS agent. If the templates have not yet been generated or are outdated, they are (re)generated at this point. The SLA template is generated following the WSAG specification. Example templates from two competing CSPs are shown in Figure 3. In these templates, each CSP displays the current resource offering.

Step 4. Upon receiving the templates, the ICRAS agent evaluates each template using the consumer's request. If a template cannot meet the requirements, it is immediately removed from consideration. In Figure 3, the template from CSP$_x$ is removed because

```
              RESOURCE REQUEST
Num. of Servers = (10)
CPU GHz         = (1.5 - 3.0) | CD:C1, VI:V1
Storage  (GB)   = (2000 - *) | CD:D100, VI:V1
Traffic (GB)    = (1 - *) | CD:D1, VI:V1
Operating Sys.  = <Windows, Linux> | PC:YES
Availability    = [95 - 100) | CD:C2, VI:V1
Price (EUR)     = [0 - 1000) | CD:D2, VI:V1
```

Fig. 2. Consumer generated resource request

```
   SLA TEMPLATE CSPx                    SLA TEMPLATE CSPy
Num. of Servers  = 100       Num. of Servers  = 50
CPU GHz          = 2.0       CPU GHz          = 3.0
Storage  (GB)    = 8000      Storage (GB)     = 4000
Traffic (GB)     = 1000      Traffic (GB)     = 500
Operating System = Linux     Operating System = {Windows OR Linux}
Availability (%) = 90        Availability (%) = 99
```

Fig. 3. SLA template from two competing CSPs

the availability offering is outside of the range specified by the consumer. In the case that more than one template remain after the first selection, the ICRAS agent evaluates them again to determine the most appropriate option. This evaluation is done by comparing key attributes, such as CPU or Availability.

Step 5. At this point, the ICRAS agent has selected the best matching CSP. The ICRAS agent generates an initial SLA offer, as shown in Figure 4. The ICRAS agent then contacts the selected CSP to begin negotiations. Following the WSAN specification, the negotiation consists of rounds of offers and counter-offers. If no mutually acceptable offer can be found, negotiation terminates and the ICRAS agent selects a different CSP. However, in the event that a mutually acceptable offer is found, this offer is sent on to the consumer.

Step 6. Once the consumer receives the offer, it re-evaluates the offering and, if acceptable, contacts the CSP directly to create a micro-SLA. After the SLA has been created, the service can be used.

Step 7. Upon successful creation of an SLA, the consumer migrates his or her services to the new CSP. This involves converting the virtual disk images to the format used by the new CSP and then transferring these images to the new CSP.

Step 8. Upon successful creation of an SLA, the ICRAS agent takes on the new task of monitoring the service on behalf of the consumer. Monitoring is done by periodically measuring key service metrics and storing the result. If a violation is detected (e.g. Availability is less than promised.), the consumer is notified and corrective action (e.g. fines, credits, and so forth) is taken. In addition to SLA monitoring, the ICRAS agent also periodically requests and evaluates SLA templates from all CSPs. If a new offering is more appropriate than the current one, the consumer is notified and migration can take place.

```
            SLA OFFER
Num. of Servers    = 10
CPU GHz            = 3.0
Storage  (GB)      = 3000
Traffic (GB)       = 10
Operating System = Windows
Availability (%) = 99
Price (EUR)        = 500
```

Fig. 4. ICRAS agent generated offer

5 Prototype Implementation

The ICRAS architecture is implemented using the AgentScape distributed middleware platform [21]. AgentScape is a distributed platform for mobile agents designed to be open, scalable, secure and fault-tolerant. This middleware provides mechanisms for SLA negotiation, inter-agent communication and migration. Software (Java) agents are used to represent the three major components: Consumer, ICRAS agent and CSP.

Two CSPs are chosen that fullfil the minimum standards of interoperability to support the example: Amazon Web Services[1] and CloudSigma[2]. On each of these CSPs a server instance is used to host a software agent running on AgentScape. Each agent uses their respective API to query price information and generate an SLA template describing each CSP's resource offerings.

An ICRAS agent runs on an instance of AgentScape on a local server. This agent collects templates from the agents running at each CSP. When the ICRAS agent has found the most suitable configuration, it is sent to the consumer agent, running on a separate instance of AgentScape on a separate local server. If a new CSP is chosen by the consumer, migration is assisted by the ICRAS agent. Virtual disk images are downloaded from the old CSP, converted to their target format using QEMU [22] and then uploaded to the new CSP.[3]

6 Related Works

Agent technology is being applied to the task of automated resource negotiation in many areas, including the area of Grid computing. Despite minor differences, Grid computing is an area that closely resembles Cloud computing in that both provide a paradigm of utility computing [23]. Tianfield uses agents to automate the task of resource negotiation in Grid computing [24]. As in the ICRAS architecture, agents are used to represent resource providers and brokers in a market. Agents apply a set of strategies to negotiate an agreement for resources. Agents are able to span multiple administrative domains to negotiate access to the necessary resources for a specific job. As with ICRAS, this

[1] `http://aws.amazon.com/`

[2] `http://www.cloudsigma.com/`

[3] Note that due to lack of standardization, a separate ad hoc solution for disk image migration is required for each unique pair of CSPs.

allows for the possibility that a single SLA includes resources from several different providers.

Sim proposes a similar architecture for automating negotiation of SLAs for Cloud resources [25]. Similar to ICRAS, this architecture supports multi-level, concurrent negotiation between multiple consumers, brokers and providers. A major difference between these two architectures and the approach used by ICRAS is the notion of time. These architectures negotiate per job, rather than per unit of time. Once an SLA is created, there is no way to dynamically respond to changes in price, utilization, and so forth. These architectures lack the benefits of micro-SLAs. There is also no impartial service to monitor the provisioning of resources according to the agreement.

Instead of agents, intelligent mapping of SLA templates is used in [26] to increase the success rate of matching Cloud service offerings to service requests. A set of public SLA templates is used as the basis of matching providers to consumers. Providers link their own template to the public template that most closely matches. The consumer then searches for a public template that matches his or her needs and contacts the related provider. To account for discrepancies between templates, users can add metadata that specifies mappings between their template and a public template. Furthermore, public templates slowly evolve to match market trends.

This approach aims to offer consumers an increased chance of finding the most appropriate resource configuration, but does not actively assist the consumer. There is no party that works on behalf of the consumer to navigate the large number of resource offerings and dynamic prices to find the most suitable CSP and negotiate an SLA. Moreover, the service migration and SLA monitoring process are left entirely to the consumer.

In contrast to the works described above, the ICRAS framework attempts to handle the entire lifecycle of a Cloud service, rather than only one or more pieces. ICRAS matching service offers to requests, such as [26] and negotiates agreements, such as [24] and [25]. In addition, ICRAS handles migration between CSPs and monitors agreements to detect possible violations.

7 Discussion

CSPs typically offer multiple interfaces to their Cloud resources, including a web interface for human access, as well as a scriptable, Application Programming Interface (API) for automated access. The API allows the consumer to purchase, launch, control and terminate Cloud resources. Furthermore, the API often gives the consumer access to pricing information. There are efforts to standardize the Cloud interface. Such efforts include the Eucalyptus [27] and OpenStack [28] open source APIs.

Another aspect that requires standardization is the data format used by clouds. CSPs use virtual disk images to encapsulate a consumer's data. These disk images use varying formats, including Amazon's AMI, Microsoft's VHD and VMWare's VMDK. If data is stored in one of these formats, there is no straightforward process to migrated to a different CSP using a different format. Each image must first be converted, following a sometimes slow and complex conversion process. While each format has its supporters, a standardized format can be used to increase the level of interoperability. The Open Virtualization Format [19] has been suggested for this purpose.

If widely adopted, these standards will make data and service migration between CSPs more straightforward. However, the main obstacle to their adoption is vendor lock-in [29]. CSPs have no incentive to make the process of service migration possible, let alone straightforward; therefore, migrating away from a CSP remains a difficult task. A consumer does not always have the option to export or download their virtual disk images from a CSP. This means, once a consumer has migrated to a particular CSP, the cost and hassle of leaving that CSP prohibits them from doing so, even if a better configuration is found at a different CSP. Note that complete state-full migration, i.e., where a snapshot of a running image is migrated and the state of the newly migrated image is updated, is still an open research question. The discussed solution would only preserve the state until the snapshot is made, so some state is lost (when the image is migrating).

Finally, wider adoption of dynamic pricing in Clouds is needed to allow users to react to changes in real market forces, including Cloud utilization. Some providers have begun offering dynamic pricing models to reflect the actual fluctuation of resource supply and demand. Dynamic pricing is beneficial to both consumers and providers of Cloud resources. Consumers can shift demand to cheaper time slots, such as evening or weekend processing, to save on costs. CSPs can take advantage of demand shifting to lower costs during peak periodes. For instance, a CSP can reduce the cost of cooling a data center at noon on a hot day by making it cheaper to use the data center at night.

Cloud computing was originally envisioned as a utility, similar to the electricity grid, where users can simply plug in to their computing needs. To enable this vision, more standardization and openness is required in the Cloud interface and data format.

The incompatibility of CSPs as discussed above greatly limits the ability to evaluate ICRAS. Preferably, an (exhaustive) evaluation would be performed to test and compare various metrics, such as migration delay, negotiation success rate and so forth. However, vendor lock-in of data formats prevents this. The scenario described in Section 5 uses two CSPs that are specifically chosen for their (limited) interoperability. However, even with these two carefully chosen CSPs, the experiment can only function in one direction. Migration is possible from CSP_1 to CSP_2 but not the other way around. CSPs must adopt open standards, as discussed above, before more extensive evaluation of ICRAS is possible.

8 Conclusions

As the Cloud continues to grow and attract users, the number of providers and Cloud resources increases. Consumers need new mechanisms to make the process of finding the most appropriate Cloud resource configuration as straightforward and automated as possible. There are many attributes that must be compared when choosing the right Cloud provider, including QoS, reputation and price.

The Intelligent Cloud Resource Allocation Service (ICRAS) gives Cloud consumers a straightforward interface to finding the most suitable Cloud resource configuration. This service compares all available offers and monitors the current price information. In addition, the service mediates the creation of micro-SLAs. Using micro-SLAs allows consumers to respond to changes in the market and renegotiated their services for lower prices or different providers. The ICRAS also monitors the service for any SLA violations.

ICRAS benefits the consumer by relieving them of the task of constantly monitoring all CSPs to find the lowest price. CSPs also benefit from ICRAS by gaining more market visibility. For instance, by participating with ICRAS, a small CSP can compete directly with larger, established CSPs, as consumers compare resource offerings independent of name recognition.

ICRAS also offers both parties the assurance that any SLA violations will be detected and reported. As an impartial party, ICRAS can monitor services without fear of bias. The passive monitoring techniques employed also generate a secure audit log that records all service messages. This log can be used to guarantee integrity and non-repudiation of these messages.

Future work will investigate the creation of complex SLAs. For instance, a consumer requires storage and compute power. One CSP offers the lowest price for storage, while another the lowest price for compute power. In this case, two separate SLAs are needed.

Additional CSP attributes will also be researched, including reputation and location. The geographical location of a CSP determines the laws that apply to the data [30].

Acknowledgements. This work is supported by the NLnet Foundation (`www.nlnet.nl`).

References

1. Clark, K.P., Warnier, M., Brazier, F.M.T.: An intelligent cloud resource allocation service - agent-based automated cloud resource allocation using micro-agreements. In: The Proceedings of the 2nd International Conference on Cloud Computing and Services Science, CLOSER 2012 (2012)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Communications of the ACM 53, 50–58 (2010)
3. Pueschel, T., Anandasivam, A., Buschek, S., Neumann, D.: Making Money With Clouds: Revenue Optimization Through Automated Policy Decisions. In: 17th European Conference on Information Systems (ECIS 2009), Verona, Italy, pp. 355–367 (2009)
4. Anandasivam, A., Premm, M.: Bid Price Control and Dynamic Pricing in Clouds. In: 17th European Conference on Information Systems (ECIS 2009), Verona, Italy, pp. 328–341 (2009)
5. Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., Sierra, C.: Automated negotiation: prospects, methods and challenges. Group Decision and Negotiation 10, 199–215 (2001)
6. Koritarov, V.: Real-world market representation with agents. IEEE Power and Energy Magazine 2, 39–46 (2004)
7. Jonker, C., Treur, J.: An Agent Architecture for Multi-Attribute Negotiation. In: International Joint Conference on Artificial Intelligence, vol. 17, pp. 1195–1201. Lawrence Erlbaum Associates LTD (2001)
8. Brazier, F., Cornelissen, F., Gustavsson, R., Jonker, C., Lindeberg, O., Polak, B., Treur, J.: A multi-agent system performing one-to-many negotiation for load balancing of electricity use. Electronic Commerce Research and Applications 1, 208–224 (2002)
9. Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., Krishnakumar, K.: A Multi-agent Infrastructure and a Service Level Agreement Negotiation Protocol for Robust Scheduling in Grid Computing. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 651–660. Springer, Heidelberg (2005)

10. Agent Technology: Foundations, Applications, and Markets. In: Jennings, N., Wooldridge, M. (eds.) Applications of Intelligent Agents, pp. 3–28. Springer (1998)
11. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) GFD-R-P.107. Technical report, Global Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG (2007)
12. Waldrich, O., Battre, D., Brazier, F.M.T., Clark, K.P., Oey, M.A., Papaspyrou, A., Wieder, P., Ziegler, W.: WS-Agreement Negotiation: Version 1.0 (GFD-R-P.193). Technical report, Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG (2011)
13. Borenstein, S.: The long-run efficiency of real-time electricity pricing. The Energy Journal 26, 93–116 (2005)
14. Ludwig, H., Dan, A., Kearney, R.: Cremona: an architecture and library for creation and monitoring of WS-agreements. In: 2nd International Conference on Service Oriented Computing, pp. 65–74. ACM, New York (2004)
15. Quillinan, T.B., Clark, K.P., Warnier, M., Brazier, F.M.T., Rana, O.: Negotiation and monitoring of service level agreements. In: Wieder, P., Yahyapour, R., Ziegler, W. (eds.) Grids and Service-Oriented Architectures for Service Level Agreements. CoreGRID, pp. 167–176. Springer, New York (2010)
16. Clark, K.P., Warnier, M., Quillinan, T.B., Brazier, F.M.T.: Secure monitoring of service level agreements. In: IEEE Fifth International Conference on Availability, Reliability and Security (ARES 2010), pp. 454–461 (2010)
17. Khader, D., Padget, J., Warnier, M.: Reactive monitoring of service level agreements. In: Grids and Service-Oriented Architectures for Service Level Agreements, Core GRID, pp. 13–22. Springer (2010)
18. Clark, K., Warnier, M., Brazier, F.M.T.: Self-adaptive service monitoring. In: Bouchachia, A. (ed.) ICAIS 2011. LNCS (LNAI), vol. 6943, pp. 119–130. Springer, Heidelberg (2011)
19. Crosby, S., Doyle, R., Gering, M., Gionfriddo, M., et al.: Open virtualization format specification 1.1.0. Technical report, DSP0243, Distributed Management Task Force, Inc. (2010)
20. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, NSDI 2005, vol. 2, pp. 273–286. USENIX Association (2005)
21. Overeinder, B., Brazier, F.: Scalable Middleware Environment for Agent-Based Internet Applications. Applied Parallel Computing. State of the Art in Scientific Computing 3732, 675–679 (2005)
22. Fabrice, B.: Qemu, a fast and portable dynamic translator. In: USENIX 2005 Annual Technical Conference, FREENIX Track, pp. 41–46 (2005)
23. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, GCE 2008, pp. 1–10. IEEE (2008)
24. Tianfield, H.: Towards agent based grid resource management. In: IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2005, vol. 1, pp. 590–597. IEEE (2005)
25. Sim, K.M.: Towards complex negotiation for cloud economy. In: Bellavista, P., Chang, R.-S., Chao, H.-C., Lin, S.-F., Sloot, P.M.A. (eds.) GPC 2010. LNCS, vol. 6104, pp. 395–406. Springer, Heidelberg (2010)
26. Breskovic, I., Maurer, M., Emeakaroha, V., Brandic, I., Altmann, J.: Towards autonomic market management in cloud computing infrastructures. In: International Conference on Cloud Computing and Services Science, CLOSER (2011)
27. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009, pp. 124–131. IEEE (2009)

28. OpenStack: Openstack: Open source software for building private and public clouds (2011), http://www.openstack.org
29. Weiss, A.: Computing in the clouds. NetWorker 11, 16–25 (2007)
30. Ruiter, J., Warnier, M.: 17. Computers, Privacy and Data Protection: an Element of Choice. In: Privacy Regulations for Cloud Computing, Compliance and Implementation in Theory and Practice, pp. 293–314. Springer (2011)