AN INTELLIGENT CLOUD RESOURCE ALLOCATION SERVICE
Agent-based Automated Cloud Resource Allocation using Micro-agreement

Kassidy Clark, Martijn Warnier and Frances M. T. Brazier
Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, Delft, The Netherlands

Keywords: Cloud Computing, Service Level Agreements, Agent Technology, Automated Negotiation, Micro-agreements.

Abstract: The Cloud refers to hardware and software resources available across the Internet. The number of competing Cloud Service Providers (CSP) continues to increase as companies outsource their computing infrastructure to the Cloud. In this environment, consumers face several challenges, including finding the least expensive Cloud service configuration, migration between CSPs and dynamically changing resource offerings. To assist consumers in this environment, this paper proposes an Intelligent Cloud Resource Allocation Service (ICRAS). This service maintains an overview of current CSP resources offerings and evaluates them to find the most appropriate configuration given a consumer's preferences. The service then negotiates a short term micro service agreement with the CSP and monitors the service for any violations. Finally, the service can assist in the migration of the consumer's data between CSPs.

1 INTRODUCTION

The Cloud refers to hardware and software resources available across the Internet (Armbrust et al., 2010). Companies that offer these resources are referred to as Cloud Service Providers (CSP). Cloud services can be roughly categorized as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). This categorization is based on the complexity of the service, from raw compute resources, such as storage or processing power, to refined software services, such as databases or other applications. This paper focusses mainly on the first of these categories: IaaS.

The Cloud computing model allows end users to rent computing infrastructure as needed, rather than requiring them to purchase their own resources. Moreover, consumers can increase or decrease the size of their Cloud resources in a straightforward and timely manner, depending on their computing requirements. Consumers use these services following a pay-as-you-go model, only paying for the specific amount of time or level of service used.

Much research into efficient use of Cloud resources focusses on increasing utility of the CSP. For instance, techniques have been proposed for load-balancing techniques aimed at reducing energy costs (Baliga et al., 2011) or dynamic pricing models that maximize revenue (Pueschel et al., 2009). In contrast, this paper proposes a service to maximize utility from the perspective of the consumer. With this service, a consumer can find the most appropriate balance between low cost and high quality.

The number of CSPs offering similar services on the market continues to grow. This increased level of providers allows consumers to pick and choose between options, based on price, Quality of Service (QoS), reputation and location. The values of some selection options are also constantly changing. For instance, in addition to set prices, some CSPs offer dynamic pricing (e.g. Amazon Web Services spot pricing). Dynamic pricing allows the price of certain resources to change to immediately reflect underlying factors, such as Cloud utilization and consumer demand (Pueschel et al., 2009, Anandasivam and Premm, 2009).

In this environment, a consumer is faced with several challenges. First, to obtain the desired initial configuration of Cloud resources, a consumer must evaluate prices of all available CSPs. The task of finding this configuration is further complicated as more CSPs implement dynamic pricing. When a consumer chooses the configuration that is currently the most appropriate, a better 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

Clark K., Warnier M. and M. T. Brazier F..
AN INTELLIGENT CLOUD RESOURCE ALLOCATION SERVICE - Agent-based Automated Cloud Resource Allocation using Micro-agreement.
DOI: 10.5220/0003928300370045
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 37-45
ISBN: 978-989-8565-05-1
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

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.

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

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. Furthermore, ICRAS continues to evaluate the available CSP resources and their prices and can recommend new configurations when they benefit the consumer. By negotiating micro-SLAs (e.g. one hour), ICRAS is able to respond to newer, and hopefully better, configurations.

ICRAS can be used by a consumer for two different stages: initial migration to the Cloud and migration between CSPs. After negotiating the initial service agreement, ICRAS constantly monitors for newly available configurations. If a better one is found, the consumer is notified and given the option to migrate to the new CSP.

The rest of this paper is organized as follows. Section 2 introduces the core concepts used in automated negotiation. The ICRAS architecture is detailed in Section 3. The ICRAS protocol is explained with a use-case in Section 4. In Section 6, other automated service negotiation architectures are compared. Finally, the implications of this research are discussed in Section 8.

2 AUTOMATED NEGOTIATION

Negotiation is the process by which one or more parties, with possibly conflicting goals, together search for a mutually acceptable agreement (Jennings et al., 2001). The negotiation process consists of proposals, counter-proposals, trade-offs and concessions, as each party attempts to maximize its own goals. 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 (Koritarov, 2004, Jonker and Treur, 2001, Jennings et al., 2001, Brazier et al., 2002, Ouelhadj et al., 2005). In this paper, an agent is be defined as a piece of software that is capable of autonomous action (Jennings and Wooldridge, 1998).

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) (Andrieux et al., 2007) and Web Service Agreement Negotiation (WSAN) (Waldrich et al., 2011) 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. The basic structure of these objects is shown in Figure 1. *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

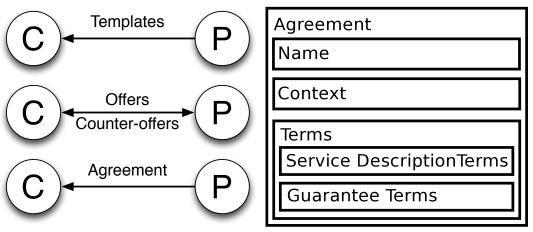

Figure 1: WSAN protocol and SLA specification.

along with their associated guarantees. If both parties accept and 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, homeowners typically make a long-term agreement with the power company for a period of one year or more. The agreement typically stipulates that the homeowner 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 (Borenstein, 2005).

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 2.

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 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.

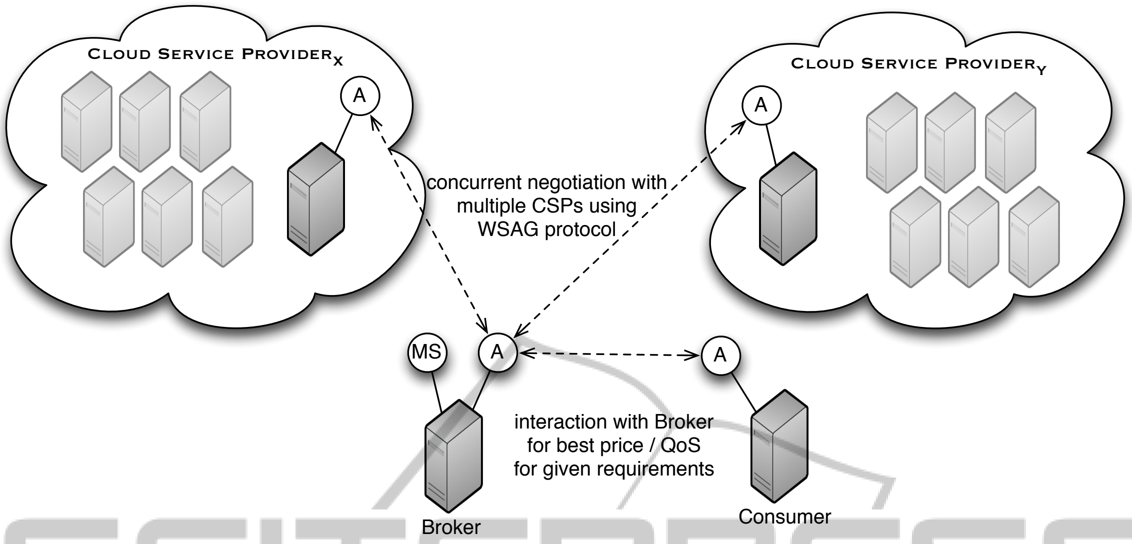
Cloud Service Provider$_X$
Cloud Service Provider$_Y$
A
A
concurrent negotiation with multiple CSPs using WSAG protocol
MS
A
A
interaction with Broker for best price / QoS for given requirements
Broker
Consumer

Figure 2: ICRAS architecture with a consumer negotiating with two competing CSPs.

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 realtime 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.

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.

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 monitors system performance and detects SLA violations. The ICRAS agent uses a Monitor Service (MS) to monitor the SLA for QoS violations, such as slow network response (Clark et al., 2010). If a violation is detected, the consumer is notified so they can take corrective action.

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 (Crosby et al., 2010).

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 (Clark et al., 2005). However, the heterogeneity of current CSPs complicates the migration process.

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 3. 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 .

```
                 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
```

Figure 3: Consumer generated resource request.

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 4. In these templates, each CSP displays the current resource offering.

```
      SLA TEMPLATE CSPx
Num. of Servers  = 100
CPU GHz          = 2.0
Storage  (GB)    = 8000
Traffic (GB)     = 1000
Operating System = Linux
Availability (%) = 90

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

Figure 4: SLA template from two competing CSPs.

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 4, the template from CSP$_x$ is removed because 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 5. 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.

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

Figure 5: ICRAS agent generated offer.

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.

5 PROTOTYPE IMPLEMENTATION

The ICRAS architecture is implemented using the AgentScape distributed middleware platform (Overeinder and Brazier, 2005). Software agents are used to represent all parties. The application of this architecture is demonstrated with an example execution.

5.1 Environmental Setup

The three major components: Consumer, ICRAS agent and CSP are represented by (Java) software agents running on the AgentScape middleware. 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.

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. These software agents represent their respective CSPs. Upon request, each agent uses their respective APIs to query price information and generate an SLA template describing each CSP's resource offerings. This template is then sent to the inquiring ICRAS agent using the mechanisms of AgentScape. If an SLA offer is received, the agent evaluates it and counters with a `SendOffers()` or accepts with a `CreateAgreement()`.

An ICRAS agent runs on an instance of AgentScape on a local server. This agent uses the mechanisms of AgentScape to communicate with the agents running at each CSP. Messages are sent to the CSP agents to `RequestTemplates()`, `SendOffers()` and `CreateAgreement()`. When the ICRAS agent has found the most suitable configuration, this is sent to the consumer agent with `SendConfiguration()`. 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 (Fabrice, 2005) and then uploaded to the new CSP.

A single consumer agent runs on a separate instance of AgentScape on a separate local server. This agent communicates with the ICRAS agent using the mechanisms of AgentScape. Messages are sent to the

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

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

ICRAS agent to `RequestConfiguration()`. This message contains the consumer's resource needs and preferences. When an acceptable SLA offer is received from the ICRAS agent, the consumer responds with `CreateAgreement()`.

5.2 Example Execution

Step 1: The consumer agent currently uses CSP_1 to host a single, Linux-based web server. The consumer generates a resource request that indicates the need for a single virtual machine with at least 10 GB of disk space and 2 GB of memory. The lowest price is the sole criterium. This request is sent to the ICRAS agent.

Step 2: The ICRAS agent receives the request and queries both CSPs for their SLA templates.

Step 3: The CSP agents respond by generating and sending their SLA templates. CSP_1 can provide the requested resources for \$0.18 per hour. CSP_2 can provide the requested resources for \$0.12 per hour.

Step 4: The ICRAS agent evaluates the two templates based on price and chooses CSP_2. An SLA offer is sent to the agent of CSP_2. The CSP accepts and responds with an SLA agreement. This SLA agreement is forwarded to the consumer.

Step 5: The consumer accepts the agreement and returns it to the ICRAS agent. In turn, the ICRAS agent forwards this to CSP_2. The consumer may now use the service.

Step 6: The ICRAS agent now assists with migration of the disk image. CSP_1 supports export of disk images in RAW format. This image is downloaded and stored locally at the ICRAS agent. CSP_2 supports import of disk images in VMDK format. QEMU is used to convert from one format to the other. Once formated, the disk image is uploaded to the CSP_2. Once uploaded, the virtual machine at CSP_2 is started. Finally, the virtual machine at CSP_1 is terminated and the service is discontinued.

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

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 (Foster et al., 2008). Tianfield uses agents to automate the task of resource negotiation in Grid computing (Tianfield, 2005). 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 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 (Sim, 2010). 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 (Breskovic et al., 2011) 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.

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 (Nurmi et al., 2009) and OpenStack (OpenStack, 2011) 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 (Crosby et al., 2010) 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 (Weiss, 2007). 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.

8 CONCLUSIONS

As the Cloud continues to grow and attract users, the numer 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 checking all CSPs and finding 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.

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 (Ruiter and Warnier, 2011).

ACKNOWLEDGEMENTS

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

REFERENCES

Anandasivam, A. and Premm, M. (2009). Bid price control and dynamic pricing in clouds. In *Proceedings of the European Conference on Information Systems*, pages 1–14.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007). Web Services Agreement Specification (WS-Agreement) GFD-R-P.107. Technical report, Global Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Baliga, J., Ayre, R., Hinton, K., and Tucker, R. (2011). Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167.

Borenstein, S. (2005). The long-run efficiency of real-time electricity pricing. *Energy Journal*, 26(3):93–116.

Brazier, F., Cornelissen, F., Gustavsson, R., Jonker, C., Lindeberg, O., Polak, B., and Treur, J. (2002). A multi-agent system performing one-to-many negotiation for load balancing of electricity use. *Electronic Commerce Research and Applications*, 1(2):208–224.

Breskovic, I., Maurer, M., Emeakaroha, V., Brandic, I., and Altmann, J. (2011). Towards autonomic market management in cloud computing infrastructures. In *International Conference on Cloud Computing and Services Science. CLOSER*.

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. (2005). Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA. USENIX Association.

Clark, K. P., Warnier, M., Quillinan, T. B., and Brazier, F. M. T. (2010). Secure monitoring of service level agreements. In *Proceedings of the Fifth International Conference on Availability, Reliability and Security (ARES 2010)*, pages 454–461. IEEE.

Crosby, S., Doyle, R., Gering, M., Gionfriddo, M., et al. (2010). Open virtualization format specification 1.1.0. Technical report, Technical Report DSP0243, Distributed Management Task Force, Inc.

Fabrice, B. (2005). Qemu, a fast and portable dynamic translator. In *USENIX 2005 Annual Technical Conference, FREENIX Track*, pages 41–46.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee.

Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., and Sierra, C. (2001). Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215.

Jennings, N. and Wooldridge, M., editors (1998). *Applications of Intelligent Agents*, chapter 1, pages 3–28. Agent Technology: Foundations, Applications, and Markets. Springer.

Jonker, C. and Treur, J. (2001). An Agent Architecture for Multi-Attribute Negotiation. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1195–1201. Lawrence Erlbaum Associates LTD.

Koritarov, V. (2004). Real-world market representation with agents. *Power and Energy Magazine, IEEE*, 2(4):39–46.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. (2009). The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE.

OpenStack (2011). Openstack: Open source software for building private and public clouds. http://www.openstack.org.

Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., and Krishnakumar, K. (2005). A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing. *Advances in Grid Computing-EGC 2005*, pages 651–660.

Overeinder, B. and Brazier, F. (2005). Scalable Middleware Environment for Agent-Based Internet Applications. *Lecture Notes in Computer Science*, 3732:675.

Pueschel, T., Anandasivam, A., Buschek, S., and Neumann, D. (2009). Making Money With Clouds: Revenue Optimization Through Automated Policy Decisions. In *17th European Conference on Information Systems (ECIS 2009), Verona, Italy*, pages 355–367.

Ruiter, J. and Warnier, M. (2011). *Privacy Regulations for Cloud Computing, Compliance and Implementation in Theory and Practice*, chapter 17, pages 293–314. Springer.

Sim, K. (2010). Towards complex negotiation for cloud economy. *Advances in Grid and Pervasive Computing*, pages 395–406.

Tianfield, H. (2005). Towards agent based grid resource management. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 590–597. IEEE.

Waldrich, O., Battre, D., Brazier, F. M. T., Clark, K. P., Oey, M. A., Papaspyrou, A., Wieder, P., and Ziegler, W. (2011). WS-Agreement Negotiation: Version 1.0 (GFD-R-P.193). Technical report, Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG.

Weiss, A. (2007). Computing in the clouds. *netWorker*, 11(4):16–25.