

Mapping High-Level Application Requirements onto Low-Level Cloud Resources

Yih Leong Sun, Terence Harmer, Alan Stewart

School of Electronics, Electrical and Computer Science, The Queen's University of Belfast, Belfast, UK.
Email: ysun05@qub.ac.uk, t.harmer@qub.ac.uk, a.stewart@qub.ac.uk

Received October 13th, 2012; revised November 11th, 2012; accepted November 20th, 2012

ABSTRACT

Cloud computing has created a paradigm shift that affects the way in which business applications are developed. Many business organizations use cloud infrastructures as platforms on which to deploy business applications. Increasing numbers of vendors are supplying the cloud marketplace with a wide range of cloud products. Different vendors offer cloud products in different formats. The cost structures for consuming cloud products can be complex. Finding a suitable set of cloud products that meets an application's requirements and budget can be a challenging task. In this paper, an ontology-based resource mapping mechanism is proposed. Domain-specific ontologies are used to specify high-level application's requirements. These are then translated into high-level infrastructure ontologies which then can be mapped onto low-level descriptions of cloud resources. Cost ontologies are proposed for cloud resources. An exemplar media transcoding and delivery service is studied in order to illustrate how high-level requirements can be modeled and mapped onto cloud resources within a budget constraint. The proposed ontologies provide an application-centric mechanism for specifying cloud requirements which can then be used for searching for suitable resources in a multi-provider cloud environment.

Keywords: Cloud Computing; Resource Mapping; Cloud Ontology; Cost Model

1. Introduction

Cloud computing provides many potential benefits to business organizations. It provides access to scalable on-demand computing resources based on a pay-as-you-use pricing model. Many organizations currently use the cloud as an alternative platform for executing business applications. However, increasing number of cloud providers supply the cloud marketplace with a wide range of cloud resources. Different providers offer cloud resources in different formats using different pricing structures. Given the wide range of products available and the complexity of the cost structure for consuming cloud resources, it can be difficult to estimate the cost of running an application in the cloud. Searching for a suitable set of cloud resources that meet an application's requirement and budget is a challenging task. For many applications, a high degree of business continuity and service availability are of paramount importance. Many business organizations demand high availability of resources and require a solution that can deliver a highly resilient business service. In the event of service interruptions, caused by a provider's resource malfunctioning, such as the AWS incidents [1], organizations should have the option to migrate their applications elsewhere.

From the application developer's point of view, it is desirable to have a programming model for constructing cloud-based infrastructures on which to deploy business applications. This model should provide an application-centric mechanism for specifying high-level application requirements and a means of translating them onto the low-level cloud resources offered by different cloud providers in the marketplace. Many frameworks have been proposed for managing cloud resources [2,3]. However, these frameworks do not provide a mechanism for comparing and selecting cloud resources according to application's constraints. Papers [4-6] propose different approaches for describing cloud resources, but requirements are typically analyzed from the provider's perspective which is usually based on the resource capabilities offered by the providers. Currently there are no suitable mechanisms for describing high-level requirements from the application's point of view

The pricing structure for consuming cloud resources varies across cloud providers. Some providers, such as Amazon Web Services, offer compute resources in multiple geographical areas and charge different prices for each area. Certain provider uses a rate-tier structure for data transfer usage (or bandwidth) whereas others use a

flat rate structure. For example, HP Cloud charges \$0.12 per GB per month for usage up to the first 10 TB, \$0.09 for the next 40 TB, \$0.07 for the next 100 TB and so on; In contrast, Rackspace charges a flat rate of \$0.18 per GB. It is difficult to compare the costs of using cloud resources for running applications in a multi-provider cloud environment.

This paper proposes an application-centric, multi-layer ontological approach for specifying requirements. The ontologies enable application developers to formulate high-level domain-specific requirements and subsequently apply these descriptions to search for the most suitable set of resources in a multi-provider cloud environment. Cost ontologies are proposed to model the costs of cloud resources; these can be used for estimating and comparing the costs of running applications in the cloud.

This paper is organized as follow. Section 2 describes an ontology-based resource mapping model. Section 3 defines ontologies for specifying cloud requirements and resources. A use case example is given in Section 4. Section 5 discusses related research and conclusions are drawn in Section 6.

2. A Resource Mapping Model: An Ontological Approach

An ontology provides a means of modelling a domain of knowledge. It provides formal descriptions of a set of entities and the relationships between them. In this paper, we propose a mechanism for mapping high-level application's requirements onto low-level cloud resources using an ontological approach (see **Figure 1**).

In the proposed model, high-level application requirements are expressed using domain-specific ontologies. These ontologies provide a semantic mechanism for capturing high-level requirements in a language or terminology familiar from a user's application domain. Domain-specific ontologies focus on user's needs. Once the high-level domain-specific ontologies are constructed, it is up to developers to translate these domain-specific ontologies onto infrastructure deployment ontologies. Knowledge databases or historical databases can be used to facilitate the translation process by providing application-specific performance and stability data. For example, in [7] several tasks are run simultaneously on a multi-core and high memory resource in order to meet a delivery deadline.

The infrastructure deployment ontologies provide constraints for resource selection. For example, in [8], a high availability and high resilience constraint results in an application being deployed on mirror infrastructures that are located at different locations. This ontological layer is resource-agnostic and gives an abstract view of infrastructure requirements from the application's perspective.

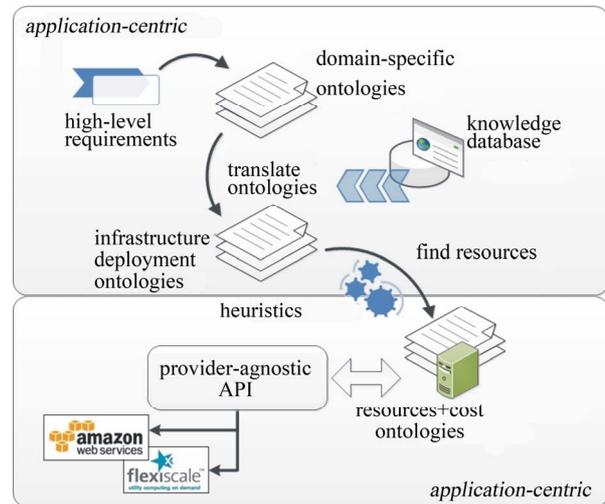


Figure 1. An ontology-based resource mapping model.

In the proposed model, infrastructure deployment ontologies are then used to search for the most appropriate set of resources from the resource layer.

The resource layer contains a pool of cloud resources offered by various providers. Resource ontologies, such as [5,6], are used to describe the features or capabilities of resources offered by cloud providers. Cost ontologies are proposed for specifying the costs of resources. The choices of cloud resources (or provider) depend on the user's preferences. Multiple cloud providers can be used as long as appropriate financial constraints are satisfied.

A two-phase resources discovery approach is used for selecting the most appropriate set of resources for a given application [9]. In the first phase, possible resources which meet an application's requirements are identified. In the second phase, suitable heuristics, such as cost or performance, are used for filtering the initial sets of resources. Once the best set of resources is identified, the resources can be instantiated and managed using a cloud management API, such as [10,11].

3. Cloud Ontologies

In this section, we describe an ontological approach for specifying cloud application requirements. OWL 2 [12] is used to specify ontologies.

3.1. Domain-Specific Ontology

A *domain-specific* ontology defines the high level requirements that are needed (or desired) for an application. These ontologies are application-specific and expressed using the terminology familiar in the user's application domain. An example of a *domain-specific* ontology is given below.

Media Transcoding and Distribution Services

Media transcoding is the process of converting media

files from one format to another; media distribution is the process of distributing media contents (using streaming media technology) to many online users (viewers or listeners) simultaneously over the internet. Media transcoding and distribution services require high CPU, high memory, large storage space and fast bandwidth. Consider a company that is offering an online training service. It needs to broadcast a series of training videos simultaneously to 500 users in UK and to 500 users in Singapore. These videos use *avi* format and are made available 5 hours before the broadcast schedule. In order to provide a good viewing experience and achieve high user satisfaction, these videos need to be transcoded into high quality formats in different screen sizes for different devices (such as *PC*, *iPad*, *iPhone*). In addition, there is a budget constraint of £5 per user. These requirements may be specified in high-level notations as follows:

Video sources: avii;

Video quality: high definition;

Playback devices: PC, iPhone, iPad;

Broadcast schedule: 9 am next morning;

Simultaneous users: 500 (UK) and 500 (SG);

Budget: £5 per user.

We have used this example and other examples of transcoding to devise a generic ontology for specifying media transcoding and distribution requirements. The ontology includes:

- Budget requirements which specify monetary constraints on leasing the infrastructure. These can be expressed as the maximum cost per day, per hour, or per user.
- Format requirements which specify the container format of the media. For example, transcoding a video from *avi* format to high quality *windows media* format.
- Device requirements which specify the destination devices (*i.e.* viewer's devices) that the media will be played on; for example, *iPad* or *iPhone*.
- Delivery time requirements specify the time when the transcoded media will be ready for broadcast.
- Capacity requirements estimate the number of users that will use the service simultaneously.
- Location requirements specify the geographical location of users that will use the service.

The ontology adopts media terminology. Many media users would be familiar with such high-level specifications rather than the details of low-level infrastructure resources (hardware or software) which support the media service. By using the proposed model, media users can specify media services' requirements using appropriate terminology. Application developers could utilize knowledge databases, such as historical records, CPU benchmarks of transcoding tasks or performance benchmarks of media servers, to map high-level *domain-*

specific ontologies onto *infrastructure deployment* ontologies.

3.2. Infrastructure Deployment Ontology

The next layer in our programming model provides a means of specifying a generic infrastructure which can support the high-level user requirements. The generic infrastructure model should have the potential to be instantiated using a wide range of cloud products, supplied from the multi-provider marketplace.

In the proposed ontology, *infrastructure requirements* define the capabilities, features or qualities that are necessary (or desired) for an infrastructure on which to execute the application. In general, *infrastructure requirements* are divided into the following categories (see **Figure 2**):

- Cost requirements specify the budget for deploying cloud infrastructure.
- Performance requirements refer to quality and performance of the infrastructure. These can be further categorized as:

1) Network latency performance, which indicates the delay incurred in the processing of data across the network;

2) Bandwidth performance, which indicates the speed of the network bandwidths including incoming and outgoing bandwidths.

- Resource requirements describe the specifications of resources, such as hardware, software and operating system. Three categories are identified:

1) Hosting environment defines the operating system requirements of the host, such as Windows 7 or Ubuntu Maverick;

2) Hardware capability defines the hardware components, such as CPU core, CPU architecture, RAM, storage space;

3) Software stack indicates the list of software or services that need to be installed on a resource.

- Geographical requirements refer to the location of resources, including data. For example, data must be processed and stored within UK.

- Compliance code requirements refer to the name or code of regulatory, industry or security standard that the infrastructure must comply with, such as HIPAA or ISO27002.

An *infrastructure requirement* can be either *hard* or *soft*. A *hard requirement* is compulsory and remains invariant over the application's lifecycle; for example, legislation regulations are compulsory; a *soft requirement* is desirable and can change or be re-prioritized; for example, budget or performance characteristics may have a degree of flexibility. Each requirement has a *priority* level, which indicates how importance the

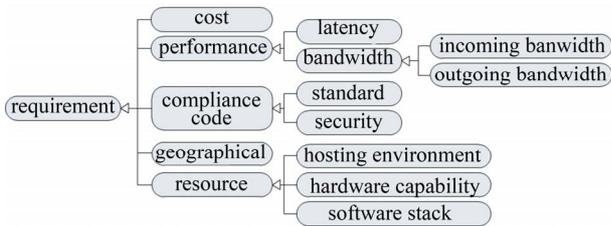


Figure 2. Categories of infrastructure requirements.

requirement is. This can be used during the requirements prioritization and resources filtering phases. *Requirements* may be inter-dependent. For example, the UK Data Protection Act (a compliance requirement) indicates that no data can be processed or stored outside the UK. This translates to a dependency relationship on geographical requirement. **Figure 3** provides an overview of the ontology for *infrastructure requirements*.

A class *restriction* is defined to identify the relevant conditions or constraints associated with a requirement (see **Figure 3**). Each *requirement* is constrained by at least one *restriction*. The detailed ontology relationships between *infrastructure requirements* and *restrictions* are given below:

- A cost requirement is constrained by cost restrictions. A cost restriction can be a total cost or it can be subdivided into compute costs, software costs, storage costs, or bandwidth costs. Each cost restriction is associated with cost frequency (per hour, per day) and financial cost (amount and currency).
- A performance requirement is constrained by performance related restrictions. Network latency performance is constrained by latency restrictions. Incoming and outgoing bandwidth performances are constrained by bandwidth restrictions. Bandwidth restriction indicates the minimum amount of bandwidth required.
- A resource requirement is constrained by resource-related restrictions.
 - 1) The hosting environment is constrained by operating system restriction which specifies the operating system types.
 - 2) The hardware capability is constrained by various hardware restrictions, such as minimum number of CPU cores, CPU speed, CPU architecture type, RAM, and storage space restriction.
 - 3) The software stack is constrained by software restrictions which specify the list of software or services that need to be installed on the resource.
- Geographical requirement is constrained by location restrictions. Location restriction indicates the location of resource or data processing.
- Compliance code requirement is constrained by compliance restrictions, which can be industry’s standard restriction or regulatory restriction.

Deployment specifications are expressed using layers: *domain*, *site*, *group*, and *node* (see **Figure 4**). A *domain* represents the top-layer of the infrastructure deployment layout and has at least one *site*. A *site* is composed of one or more *groups*. A *group* contains a set of *nodes* which provide same functionality, such as web servers or databases. A *node* is a specific type of resource such as a computational unit or storage. The ontologies include requirements which apply to many different layers of the deployment structure. For example, if a *location requirement* is applied at the *domain* layer, all *sites*, *groups* and *nodes* within the *domain* must fulfil the same location constraint; *hardware requirements*, such as CPU and memory, can be applied at *group* level or at individual *node* level.

Once the *infrastructure deployment specification* is defined, it is then used to search for resources in the low-level resource pool.

3.3. Resource Ontology

The *resource* ontology defines the properties of the resources offered by cloud providers. This layer has been widely investigated elsewhere [5,6]. In the proposed model, we adopt a similar approach as [6] (using the concept of resource capabilities) for describing cloud resources.

A *cloud resource* is associated with different resource capabilities, which can be *storage capability*, *compute capability*, *memory capability*, *software capability* and *host capability*. *Storage capability* consists of amount of storage space and various storage types, such as local

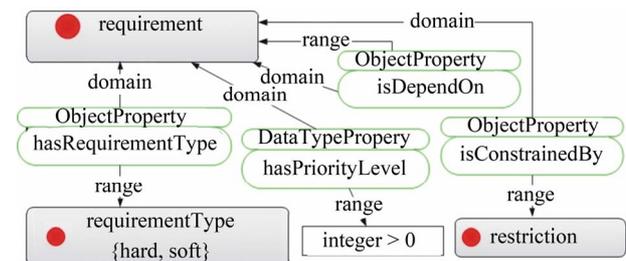


Figure 3. Overview of infrastructure requirements ontology.

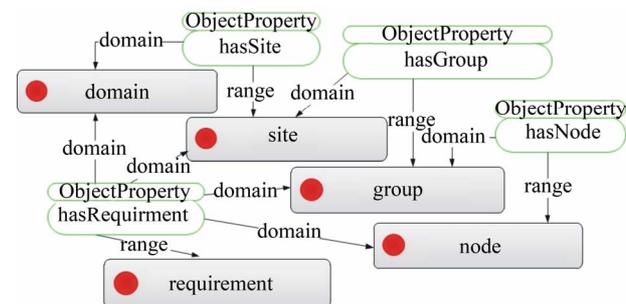


Figure 4. Ontology for infrastructure deployment.

storage or SAN storage; *compute capability* consists of CPU core, CPU architecture type, CPU speed; *memory capability* consist of memory size; *software capability* consist of software name, version, maker and features; *host capability* consist of operating system type and version. *Cloud resource* also has other properties such as *location* and *vendor* name. An overview of *cloud resource* ontology is illustrated in **Figure 5**.

Every *cloud resource* has a *price* structure which represents the resource consumption cost. Estimating costs of using cloud resources is a challenging task. Different providers have different means of charging cloud resources. In this paper, we consider normal pay-as-you-use pricing model. Discounted prices, such as Amazon Web Services' reserved instances or free-tier offers are not considered. We analyse the pricing structures of various providers and propose a simplified ontological pricing model. The *cost* ontology is shown in **Figure 6**.

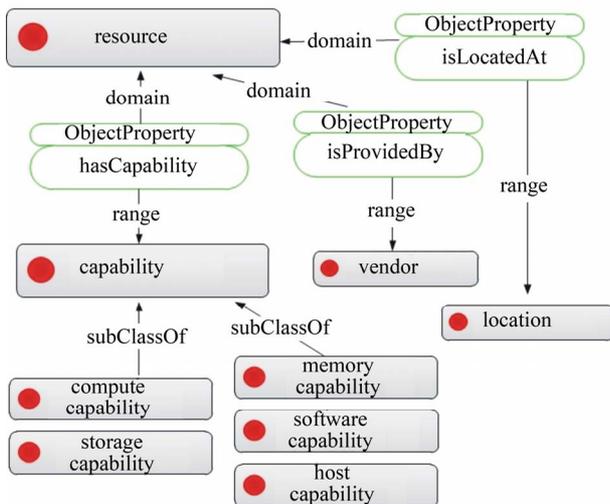


Figure 5. Overview of resource ontology.

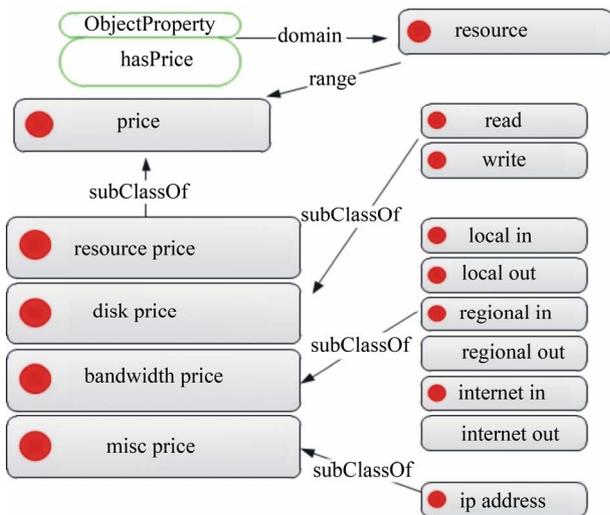


Figure 6. Cost ontology.

In general, prices of consuming cloud resources can be categorized as follows:

- Resource prices refer to the cost of allocating the resources and keeping the resources switched on (*i.e.* keep-alive). These include machine cost and storage cost.
- Bandwidth prices refer to the cost of data transfer in to and out from the resource. These include data transfer within local network, regional network or internet network.
- Disk prices refer to the cost of disk operations. For example, disk read request and disk write request.
- Miscellaneous prices refer to other costs not included in the above categories, such as the price of allocating additional IP addresses.

The proposed *resource* and *cost* ontologies provides a semantic mechanism for annotating cloud resources offered by different providers.

4. A Media Transcoding and Distribution Example

In this section, a media example is used to illustrate how a *domain-specific* ontology can be translated to a set of *infrastructure deployment* constraints which are in turn mapped onto actual *cloud resources*. Media developers need to provision an infrastructure which fulfils transcoding and distribution requirements as well as satisfying cost and delivery time constraints.

4.1. Specifying High-Level Requirements Using Domain-Specific Ontology

Based on the example given in previous section, high-level media requirements are represented using the proposed *domain-specific* ontology:

- Converting *avi* to high definition *Windows Media* format is a type of *Format* requirement.
- Supporting playback devices, such as *PC*, *iPad* and *iPhone*, is a type of *Device* requirement.
- Serving 500 users in UK and 500 users in Singapore are types of *Capacity* and *Location* requirement.
- Making the media available before 9 am next morning is a type of *Delivery Time* requirement.
- Having a budget constraint of £5 per user is a *Budget* requirement.

Table 1 summarizes the representation of high-level media requirements using *domain-specific* ontology.

4.2. Translating Domain-Specific Ontology onto Infrastructure Deployment Ontology

After the *domain-specific* ontology is identified, it is then mapped onto middle layer *infrastructure deployment* ontology. Knowledge databases such as historical records

Table 1. Specifying high-level requirements using domain-specific ontology.

| High-level requirements | Domain-specific ontology |
|---------------------------------|--------------------------|
| 500 users in UK | Capacity/Location |
| 500 users in Singapore | Capacity/Location |
| Convert AVI to HD Windows Media | Format |
| PC, iPad, iPhone | Device |
| 9am next morning | Delivery Time |
| £5 per user | Budget |

and performance benchmark reports are used to facilitate the translation process.

Transcoding requirements, such as format and device, indicate the features or capabilities of transcoding software that need to provide. Particular software, such as FFmpeg, Rhozet, or Microsoft Expression Encoder, can be used to perform the transcoding task. These domain-specific requirements are translated onto software stack requirements at the infrastructure layer. However, different software has different system requirements. For example, Microsoft Expression Encoder must be run on Windows operating system, whereas FFmpeg can be run on Linux. In the proposed ontology, these infer a dependency requirement on hosting environment.

Media distribution requirements, such as high quality high-definition format and capacity of simultaneous users, indicate the hardware constraints (CPU, memory, bandwidth, number of servers) for the media distribution servers. For example, a high-definition video (*windows media*, 720 p) requires a bit-rate of 2 mbps. In order to serve 500 users simultaneously, at least 30 *Windows Media Servers* would be required to service this demand (based on the streaming server benchmark [13]). In addition, tight delivery deadline and high quality requirements may necessitate the use of multi-core, high-CPU or high-memory resources. Multiple country distribution channels indicate the needs of deploying media servers at different locations. These media distribution requirements are translated onto infrastructure requirements-performance, hardware and geographical (see **Table 2**). Based on the media requirements, the *infrastructure deployment specification* is formed as follow:

- Transcoding *domain* is provisioned as a single *site* containing a *group* of transcoder resources. An incoming bandwidth requirement of 2 mbps is required for receiving the video source. A minimum outgoing bandwidth of 4 mbps is required for sending the transcoded media to the distribution *domain* (with two different *sites*).

- Distribution *domain* is provisioned as two different *sites* (one in UK and another in Singapore). Each *site* contains a *group* of 30 media servers and requires an incoming bandwidth of 2 mbps to each *site*. In order to serve 500 simultaneous users, a minimum outgoing bandwidth of 1000 mbps (500×2 mbps) is required on each *site*, split among 30 media servers.

Another high-level requirement is the budget constraints of £5 per user. This budget includes the cost of uploading video source, provisioning resources in transcoding domain and distribution domain and cost of data transfer to serve 500 UK users and 500 Singapore users. This is formulated in Equation (1) and used in the resource filtering and selection process at the later stage.

$$B \geq (R_t + D_{up}) + (R_{da} + D_{upa} + S_{da}) + (R_{db} + D_{upb} + S_{db}) + D_s + D_u \quad (1)$$

where

$$B = \text{budget} = £5 \times (500 + 500) = £5000;$$

$$R_t = \text{compute cost in transcoding domain};$$

$$D_{up} = \text{cost for uploading video source};$$

$$R_{da} = \text{compute cost in UK distribution domain};$$

$$S_{da} = \text{storage cost in UK distribution domain};$$

$$D_{upa} = \text{cost for uploading transcoded video to UK site};$$

$$R_{db} = \text{compute cost in SG distribution domain};$$

$$S_{db} = \text{storage cost in SG distribution domain};$$

$$D_{upb} = \text{cost for uploading transcoded video to SG site};$$

$$D_s = \text{data transfer cost to SG users};$$

$$D_u = \text{data transfer cost to UK users}.$$

4.3. Specifying Cloud Resources Using Resource Ontology

In the proposed model, developers generate a pool of infrastructure resources as an integral part of the development process. Although this incur additional development effort but it simplifies and automates the process of selecting resources in a multi-provider environment. In the experiment, Amazon AWS EC2 is chosen as the infrastructure provider. It is being considered as a few sub-providers (US Northern Virginia, US Oregon, US Northern California, EU Ireland, AP Singapore, AP Tokyo and SA Sao Paulo) because it offers resources across multiple geographical regions. About half a million of resources are generated (by combining different AWS instance types with different AMI images in different regions) and annotated using the proposed *resource ontology*. **Table 3** illustrates an example on how an AWS resource is annotated with the *resource ontology*.

Table 2. Mapping domain-specific ontology onto infrastructure requirements ontology.

| Domain-specific ontology | Infrastructure ontology | Infrastructure ontology (inferred) |
|---|---|--|
| Format (avi-HD wmv) | Software Stack (MS Expression) | Hosting Environment (Windows) |
| Device (PC, iPad, iPhone) | Software Stack (MS Expression) | Hosting Environment (Windows) |
| Format (high quality) | Hardware-Transcoder (8 cpu, 3 GHz, 8 GB RAM) Bandwidth (2 mbps-in/4 mbps-out) Hardware-Media Server (2 cpu, 3 Ghz, 4 GM RAM), | |
| Capacity/Location (500 SG/500 UK users) | Bandwidth (2 mbps \times 500-out) Geographical (UK, SG) | |
| Delivery Time (9 am) | Hardware (high cpu, high memory) | |
| Budget (£5 per user) | Total Cost (£5 \times 1000) | Storage Cost (UK, SG) Resource Cost (UK, SG) Data Transfer Cost (UK, SG) |

Table 3. Annotate AWS resource using resource ontology.

| AMI | Instance | Region | Resource ontology |
|---------------------------------------|--------------------|------------|--|
| Microsoft Windows Server 2008, 64 bit | m1.xlarge (normal) | EU Ireland | Compute Capability (4 cores, \times 86_64) |
| | | | Memory Capability (15 GB) |
| | | | Storage Capability (1690 GB) |
| | | | Host Capability (Windows Server 2008 R2 SP1, \times 86_64) |
| | | | Software Capability (Windows Media Streaming) |
| | | | Location (Ireland) |
| | | | Vendor (AWS EU Ireland) |
| | | | Resource Price (\$0.92/hour) |
| Bandwidth Price (\$0.12/GB/out) | | | |

4.4. Selecting Cloud Resources

Once the *infrastructure deployment* ontology is defined, eligible cloud resources are selected from the resource pool using a two-phase resource discovery approach [9]. In the first phase of the selection process, sets of possible resources which meet all the *hard* requirements

(such as location) are identified. In the second phase, performance and cost heuristics are used to filter resources from the initial set.

In this example, hardware constraints, such as CPU and memory, is applied as *soft* requirements to the transcoding domain. This allows the system to compare the performance and cost of either using GPU resource or standard resource. For example, the transcoding tasks took 45 minutes to complete if an *AWS Standard Extra-Large* resource is used, however, it took 30 minutes to complete if an *AWS Cluster GPU Quadruple-Extra-Large* resource is used. Although there is a significant performance improvement when GPU resource is used, the standard resource costs \$0.92 per hour but GPU resource costs \$2.6 per hour (any partial hour consumed will be billed as a full hour). Using the standard resource is a cheaper solution while delivery deadline constraint is still enforced.

Two groups of compute resources are selected for the UK distribution domain and Singapore distribution domain. The cost of the whole infrastructure is calculated by taking into the consideration of the compute cost, storage cost, and data transfer cost as defined in Equation (1). **Figure 7** illustrates how the media transcoding and distribution requirements are mapped onto low-level cloud resources.

5. Related Work

The Mosaic project [5] uses ontologies for annotating cloud resources with a set of functional and non-functional properties. Bernstein *et al.* [6] propose an ontology-based catalogue which describes features and capabilities

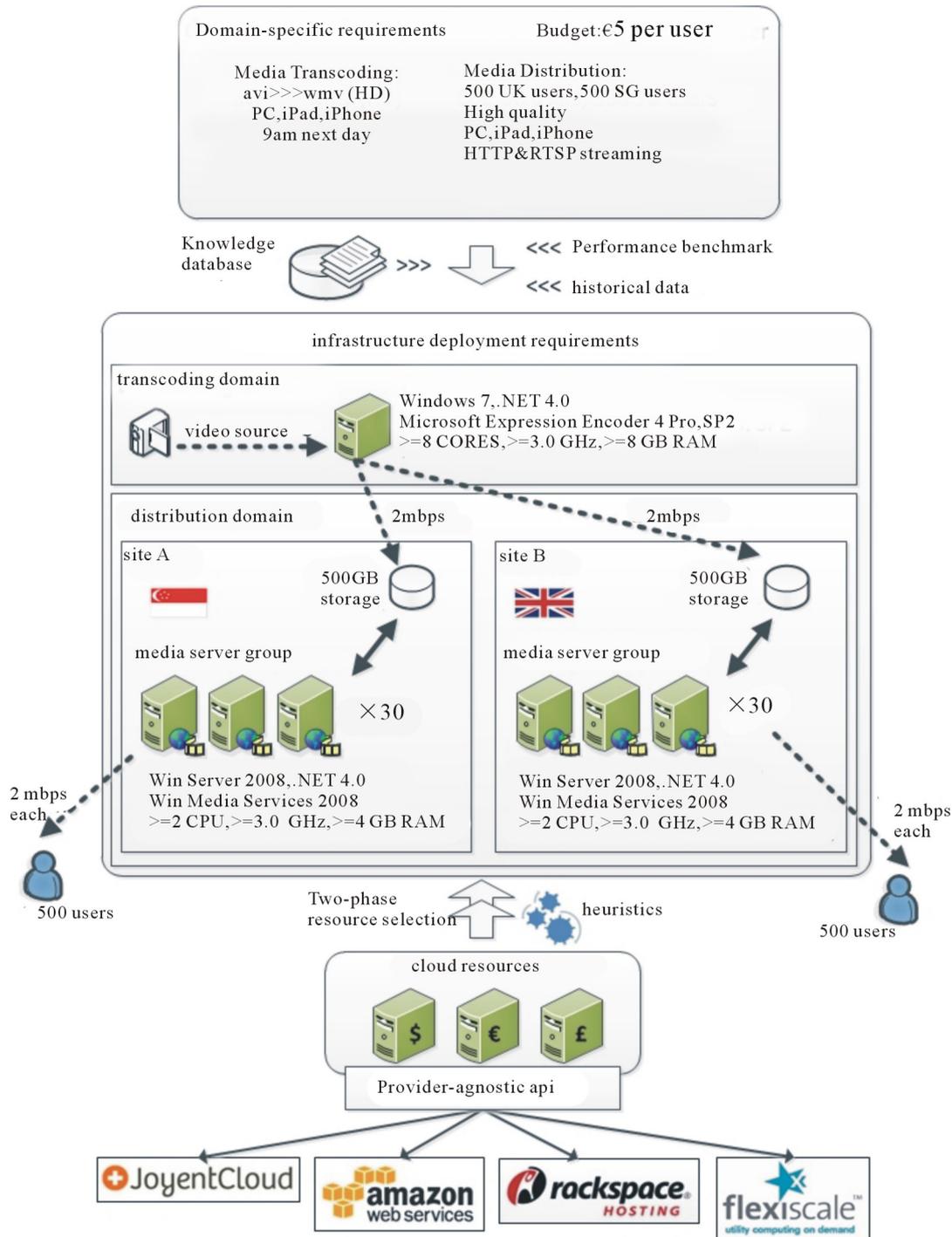


Figure 7. Example-media transcoding and distribution.

of resources offered by cloud providers, such as CPU, storage, security, and compliance capabilities. However, cost factors are not discussed in [5,6]. The RESERVOIR project proposes a service specification mechanism [4], by extending DMTF’s OVF standard, which includes VM details, application settings and deployment settings. LoM2HiS [14] proposes a framework for mapping low-level resource metrics to high-level SLA parameters.

Although the SLA parameters are high-level entities, LoM2HiS focuses on hardware and network attributes. Truong and Dustdar [15] propose various cost models which can be composed to determine the costs for executing application. In the resource ontology layer, we combine the approach in [6,15] and enhance the model by including cost details. Most of the frameworks are resource-centric and investigated from cloud provide’s

perspective. This paper proposes application-centric multi-layer ontologies that focus on the requirements of applications rather than just the cloud resources available in the market.

6. Conclusions and Future Work

Cloud computing provides highly flexible and cost-effective platform for deploying business applications. The cloud marketplace comprises a dynamic environment of providers and products. Searching for suitable resources in such a dynamic environment within a given budget is challenging. Little attention has been paid to describe a cloud application's requirements at an appropriate level of abstraction. In this paper, an application-centric, multi-layer ontology for describing cloud application requirements is proposed. This ontology provides a semantic mechanism for capturing application needs in a language familiar from users' application domains. A cost ontology for specifying the costs of consuming cloud resources is introduced. A two-phase resource mapping mechanism is also considered. A media application is used to illustrate how application requirements can be formulated and mapped onto low-level cloud resources.

We hope to extend the requirement ontologies by investigating other application domains. We believe that the proposed ontological approach provides an effective mechanism for specifying cloud requirements semantically, which can be utilized in the resource selection process in a multi-provider cloud environment.

REFERENCES

- [1] Amazon, "Summary of AWS Service Event in the US East Region." <http://aws.amazon.com/message/67457/>
- [2] D. Nurmi, R. Wolski, C. Grzegorzczak, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*, Shanghai, 18-21 May 2009, pp. 124-131. [doi:10.1109/CCGRID.2009.93](https://doi.org/10.1109/CCGRID.2009.93)
- [3] R. S. Montero, "Open NEBula: The Open Source Virtual Machine Manager for Cluster Computing," *Open Source Grid and Cluster Conference*, Oakland, 13-15 May 2008.
- [4] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmerich and F. Galán, "The Reservoir Model and Architecture for Open Federated Cloud Computing," *IBM Journal of Research and Development*, Vol. 53, No. 4, 2009, pp. 535-545.
- [5] F. Moscato, R. Aversa, B. D. Martino, T. Fortis, V. Munteanu, "An Analysis of mOSAIC Ontology for Cloud Resources Annotation," *Proceedings of 2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Szczecin, 18-21 September 2011, pp. 973-980.
- [6] D. Bernstein and D. Vij, "Intercloud Directory and Exchange Protocol Detail Using XMPP and RDF," *Proceedings of the 6th World Congress on Services (SERVICES-1)*, Miami, 5-10 July 2010, pp. 431-438. [doi:10.1109/SERVICES.2010.131](https://doi.org/10.1109/SERVICES.2010.131)
- [7] T. Harmer, P. Wright, C. Cunningham, J. Hawkins and R. Perrott, "An Application-Centric Model for Cloud Management," *Proceedings of the 6th World Congress on Services (SERVICES-1)*, Miami, 5-10 July 2010, pp. 439-446. [doi:10.1109/SERVICES.2010.132](https://doi.org/10.1109/SERVICES.2010.132)
- [8] Y. L. Sun, R. Perrott, T. Harmer, C. Cunningham and P. Wright, "An SLA Focused Financial Services Infrastructure," *Proceedings of the 1st International Conference on Cloud Computing Virtualization (CCV 2010)*, Singapore, 2010. [doi:10.5176/978-981-08-5837-7_189](https://doi.org/10.5176/978-981-08-5837-7_189)
- [9] P. Wright, Y. L. Sun, T. Harmer, A. Keenan, A. Stewart and R. Perrott, "A Constraints-Based Resource Discovery Model for Multi-Provider Cloud Environments," *Journal of Cloud Computing: Advances, Systems and Applications*, Vol. 1, 2012, p. 6. [doi:10.1186/2192-113X-1-6](https://doi.org/10.1186/2192-113X-1-6)
- [10] T. Harmer, P. Wright, C. Cunningham and R. Perrott, "Provider-Independent Use of the Cloud," *Proceedings of the 15th International Euro-Par Conference on Parallel Processing (Euro-Par 2009)*, Delft, 25-28 August 2009, pp. 454-465. [doi:10.1007/978-3-642-03869-3_44](https://doi.org/10.1007/978-3-642-03869-3_44)
- [11] A. Edmonds, T. Metsch and A. Papaspyrou, "Open Cloud Computing Interface in Data Management-Related Setups," In: S. Fiore and G. Aloisio, Eds., *Grid and Cloud Database Management*, Springer, Berlin, 2011, pp. 23-48. [doi:10.1007/978-3-642-20045-8_2](https://doi.org/10.1007/978-3-642-20045-8_2)
- [12] W3C, "The OWL 2 Web Ontology Language." <http://www.w3.org/TR/owl2-overview>
- [13] Microsoft, "Calculating Required Server Capacity." [http://technet.microsoft.com/en-us/library/cc772121\(v=w s.10\)](http://technet.microsoft.com/en-us/library/cc772121(v=w s.10))
- [14] V. C. Emeakaroha, I. Brandic, M. Maurer and S. Dustdar, "Low Level Metrics to High Level SLAs-LoM2HiS Framework: Bridging the Gap between Monitored Metrics and SLA Parameters in Cloud Environments," *Proceedings of 2010 International Conference on High Performance Computing and Simulation (HPCS)*, Caen, 28 June-2 July 2010, pp. 48-54. [doi:10.1109/HPCS.2010.5547150](https://doi.org/10.1109/HPCS.2010.5547150)
- [15] H. L. Truong and S. Dustdar, "Composable Cost Estimation and Monitoring for Computational Applications in Cloud Computing Environments," *Procedia Computer Science*, Vol. 1, No. 1, 2010, pp. 2175-2184. [doi:10.1016/j.procs.2010.04.243](https://doi.org/10.1016/j.procs.2010.04.243)