A Substrate Description Framework and Semantic Repository for Publication and Discovery in Cloud-Based Conferencing

Jerry George[#1], Fatna Belqasmi[#2], Roch Glitho[#3], Nadjia Kara[*4]

[#]*Concordia University, Canada*
[#1]jerry.george@concordia.ca
[#2]fbelqasmi@alumni.concordia.ca
[#3]glitho@ece.concordia.ca

[*]*ETS, University of Quebec, Canada*
[*4]nadjia.kara@etsmtl.ca

***Abstract* – Cloud computing is an emerging paradigm with three main facets: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Several benefits are expected from cloud-based conferencing (e.g. efficiency in resource usage, easy introduction of new conferencing applications). This paper proposes a publication and discovery architecture for the interactions between the substrate providers, the infrastructure providers, and the broker of a cloud based conferencing business model.**

***Keywords*—cloud-based conferencing, publication, discovery, semantic repository, cloud conference ontology.**

I. INTRODUCTION

Conferencing is the conversational exchange of media between several parties. A business model has been recently proposed for cloud-based conferencing [1]. There are five roles in the proposed business model: connectivity provider, broker, conferencing substrate provider, conferencing infrastructure provider, and conferencing service provider. Conference substrates are elementary building blocks that can be virtualized and shared between conferencing applications for resource efficiency purposes. This paper proposes an architecture for realizing the interactions between the substrate provider, the infrastructure provider, and the broker. The substrates need to be described in a non-ambiguous manner for publication and discovery purposes from both technical and business perspectives. Furthermore, a repository is also required to enable the actual publication and discovery of the substrates.

Our proposed architecture is made up of a semantic-oriented description framework for substrates and a repository for publication and discovery of the substrates. The description framework is made up of a substrate description language and cloud-based conference ontology, both of which should meet ten key requirements.

First, the substrate description framework should be standards-based. Second, it should enable machine-readable substrate description. Third, the substrate description framework should hide the heterogeneity of the substrates and provide the service interfaces in a uniform manner. Fourth, the substrate description language and cloud conference ontology should accommodate both the technical and business aspects of the conference substrates. Fifth, the substrate description language should be flexible by supporting a wide range of data formats.

Sixth, the repository interface for publication and discovery should be independent of the stored substrates. Seventh, the interface should be based on existing standard protocols/APIs. Eighth, to support easy interoperability, the interface should be flexible in terms of the supported serialization formats for substrate description. Ninth, the interface should enable the specification of both technical and business aspects using standard technologies, while publishing or discovering the substrates. Tenth, the substrate repository should provide either an extensible architecture for adding support for new languages or explicit support for a chosen description language.

Work has been done on both the substrate description language [2], [3] and cloud-based conference ontology [4], [5]. However, none of this related work meets all the requirements. The next section presents the proposed architecture, followed by the implementation architecture and prototype. The final section concludes this paper.

II. PROPOSED ARCHITECTURE

In this section, the overall architecture is presented first, followed by the substrate description framework, and then the substrate repository.

A. OVERALL ARCHITECTURE

Figure 1 depicts the overall proposed architecture. The substrate and infrastructure providers communicate with the repository via a REST interface. The discovery requests are described using SPARQL, and are transferred as REST request content.

The substrate repository uses a semantic data store to save the substrate descriptions and the cloud conference ontology, which serves as a reference ontology for the validation of substrate description documents during publication. The repository includes a set of supporting components to access, validate, and manage the substrate description documents and cloud conference ontology. These components can be classified into three categories. The first category supports the validation and the management of the substrate descriptions

and it includes the substrate document validator and substrate classifier.

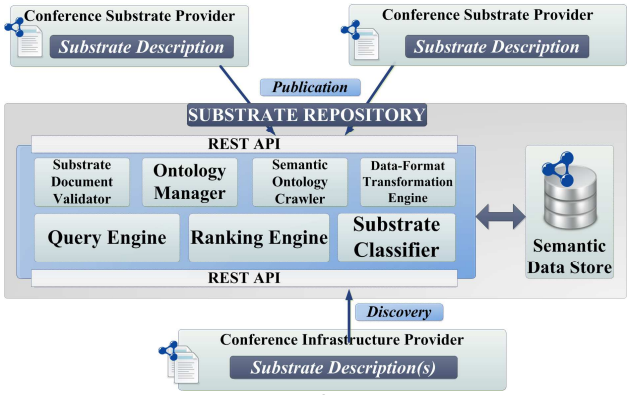

Fig. 1: Proposed architecture for publication and discovery

The second category is used for the management of the cloud conference ontology and it consists of the ontology manager and semantic ontology crawler. The last category enables efficient discovery of substrates and it contains the query and ranking engines. The data-format transformation engine is a supporting component used for both management and discovery of substrates.

B. Substrate Description Framework

The description framework defines a new cloud-based conference ontology and reuses OWL as the description language. The cloud conference ontology consists of three key constituent ontologies – cloud infrastructure, substrate description and conference ontologies. The reasoning support for these ontologies can be supported by OWL-DL reasoners. It reuses existing ontology (e.g. Linked-USDL) concepts, which extends to meet the conferencing specifics.

The cloud infrastructure ontology describes the business aspects of the cloud conferencing infrastructure, such as the substrate and the infrastructure providers' information, and the subscription information (i.e. which infrastructure provider is subscribed to which substrate). Figure 2 presents the main concepts and properties that constitute the cloud infrastructure ontology.

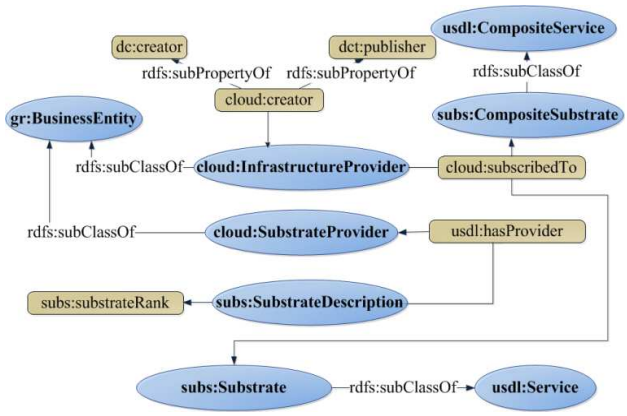

Fig. 2: Cloud Infrastructure Ontology

The conferencing substrates are modelled as Linked-USDL services, allowing the reuse of the Linked-USDL models for expressing the pricing (e.g. per user, per month, etc.) and the constraints information. Linked-USDL allows constraints specification for both atomic substrates (e.g. signalling substrate) and composite substrates (e.g. dial-out audio conference substrate).

The substrate description ontology (Figure 3) describes the technical aspects of the substrates, such as the interfaces and the substrate features. The substrate interfaces are described through the set of operations they encompass, along with the inputs and outputs of each operation. The operations are described as per the SA-REST service model, which we extend in order to support asynchronous substrate operations. We added a collection of seven properties to define an asynchronous callback end-point. The substrate features indicate the other functional features of the substrate (i.e. other than the interface ones), such as the substrate type (e.g. audio mixing, signalling). Composite substrates may have multiple features or capabilities, which are described using an RDF list. The substrate description ontology provides a classification for the common conferencing substrate features,

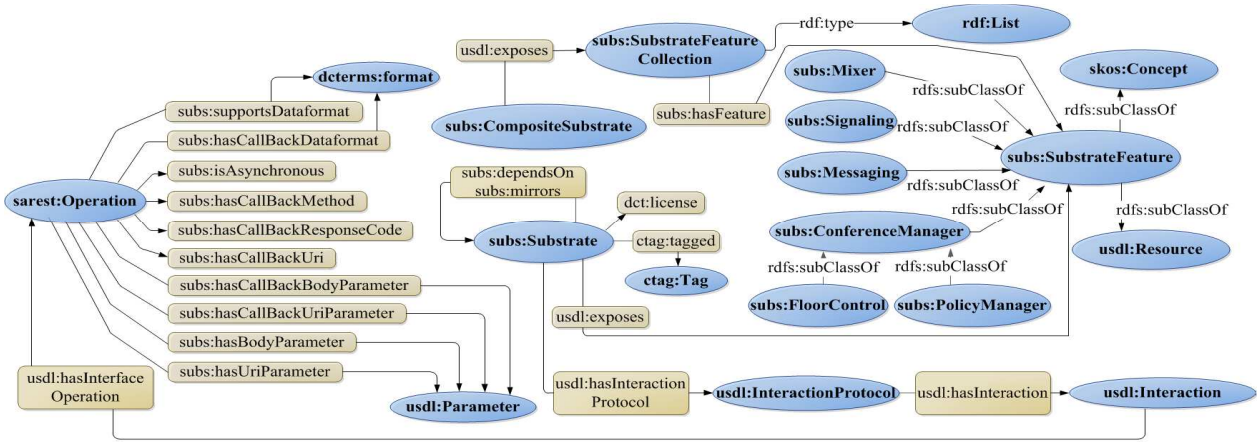

Fig. 3: Substrate Description Ontology

including signalling, mixing, and advanced conference control features such as floor control and policy management.

The conference ontology gives in-depth information about the conference and its participants (Figure 4). A conference is depicted as a composition of a set of substrates. A conference is also defined as a Linked-USDL resource, to capture the fact that it is the concrete object that implements the conferencing service. The participants are described using three important descriptors – signalling, media and preference descriptors.

C. *Substrate Repository*

The substrate provider may choose to publish the substrate description document in any supported RDF serialization format. Prior to storing a published document, the substrate repository converts the document into XML format using the data transformation engine. The substrate repository then checks the document validity against the cloud conference ontology and set of inference rules. This function is handled by the substrate document validator, which seeks the help of the ontology manager to retrieve the latest version of the ontology from the semantic data store. Once the validation is completed, the substrate description document is stored in the semantic data store. At regular intervals of time, the substrate classifier indexes the published documents based on the substrates' type (e.g. signalling, mixing, etc.). Indexing periodically instead of after each publication optimizes the repository resource usage and up time. For instance, the indexing may be scheduled for periods when traffic is low, and use the full capacity of the repository to answer the users' requests during the busiest period. The indexing reduces the response time for simple discovery requests (e,g. those based on substrate type), and it is performed only when needed. The infrastructure provider can look for a substrate by providing the criteria required as part of the request content. The criteria are specified using the SPARQL specification. Upon receiving the request, the substrate repository uses the query engine to parse the SPARQL query and ensures the request is coherent with the described ontologies. The query engine is then used to optimize the query using SPARQL re-writing rules for basic graph pattern (BGP) based on the index generated by the substrate type classifier. The infrastructure provider may limit the number of substrates to get in the response, in which case the ranking engine is used to prioritize the results. The ranking engine utilizes the multi-criteria decision making scheme proposed in [6] to rank the substrates based on some of their characteristics (e.g. latency, availability, and cost). The description documents of the selected substrates are then reformatted (if needed) according to the data format (e.g. XML, JSON, N3) supported by the infrastructure provider. Such a transformation is performed by the data transformation engine.

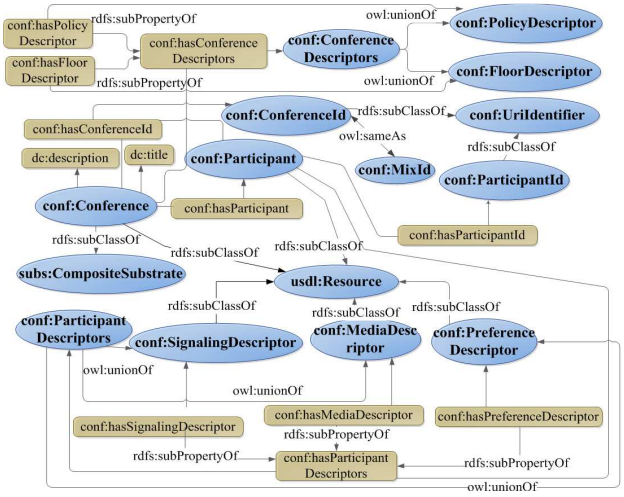

Fig. 4: Conference Ontology

III. Implementation

We first present the implemented prototype, followed by the performance measurements.

A. *Prototype*

The prototype consists of a substrate repository with both publication and discovery interfaces, and a set of infrastructure and conference substrate providers. The semantic data store component of the repository is based on Sesame and the other components are implemented using Sesame and RDF2Go libraries. Sesame is an open-source framework for storing and querying RDF data, and RDF2Go provides an abstraction layer for easier communication with the Sesame data store. The built-in SPARQL query optimizer of Sesame is extended to support optimizations based on BGPs related to substrate types. To support inference and validation, the Sesame framework's parser module is used along with OWLIM[1], a family of semantic-based database management systems. The data transformation engine uses Apache Any23 libraries for transformation between the RDF serialization formats. The REST interfaces are implemented using Jersey, a reference implementation of JSR 311.

To have a near-realistic view of the system execution, we needed a test bed setup with several dozens of substrates belonging to different providers, as well as random and varied constraints. We implemented a benchmarking tool including a substrate test data generator and a query generator, representing a set of substrate and infrastructure providers respectively. Both generators are implemented using Java concurrency API and can issue varying numbers of parallel requests to the substrate repository. Some of the existing benchmarking tools for RDF-based repositories such as Berlin SPARQL Benchmark[2] allow only the benchmarking of pre-defined use cases with specific sets of product templates.

Two laptops were used to run the prototype. The first one was used to run the substrate repository, while the second was used to run the benchmarking tool for publication and discovery.

[1] OWLIM - http://www.ontotext.com/owlim

[2] Berlin SPARQL Benchmark (BSBM) - http:// bit.ly/17RxHHZ

B. *Performance Metrics*

The performance of our prototype is assessed in terms of time delays for both publication and discovery. The publication delay measurements were taken for different numbers of substrate providers, different number of simultaneous requests, and for the cases where different numbers of substrates were published prior to the time of measurements. The discovery delays were measured for two types of queries: simple and complex. Simple queries are, for instance, those based only on the substrate type. Complex queries may include multiple relational criteria (e.g. capacity>=100 and latency<=1000ms), textual operations (e.g. textual search for a specific provider or substrate within a specific region), or ranking criteria (e.g. get an ordered list of the first 10 recommended audio mixers in Canada). We also compared the discovery delays of simple queries with and without optimization to show the added value of the optimization algorithm.

C. *Performance Results*

Figure 5 shows the results. Each measurement is calculated as an average of 15 experiments. Figure 5.a displays the measurements for publishing up to 32 substrates simultaneously by varying the number of existing substrates in the semantic data store. As expected, the delays increase with the number of simultaneous publications as well as the number of substrates already in the registry. Nevertheless, the delays remain acceptable considering that the publication is a one-time operation performed by the substrate providers.

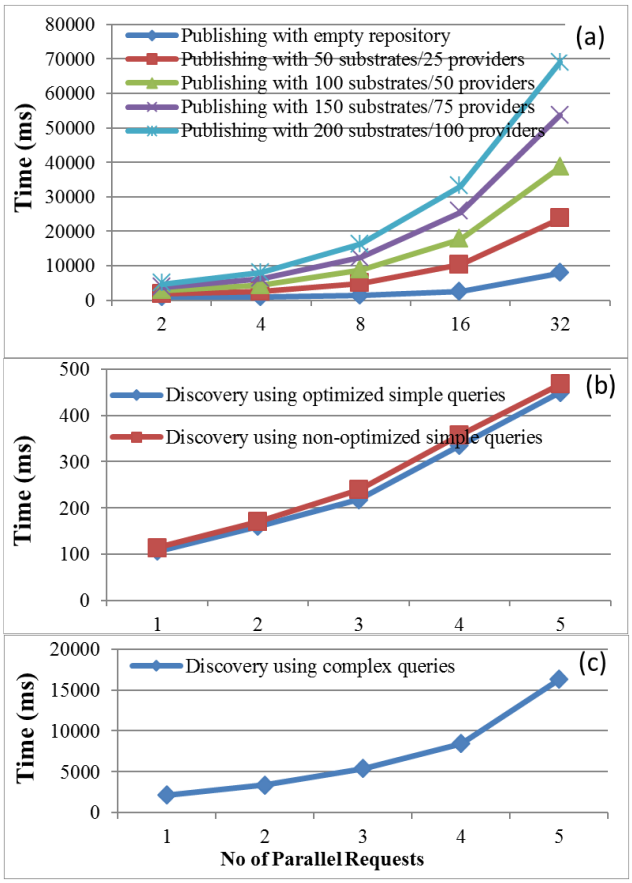

Fig. 5: Performance measurements for substrate repository: a) publication delays; b) discovery delays for simple queries; c) discovery delays for complex queries.

The discovery delay measurements were performed on a substrate repository containing 100 substrates. The discovery requests are randomly generated by the benchmarking tool, according to the chosen request complexity (i.e. simple or complex). Figure 5.b compares the discovery delays for optimised and non-optimised simple queries. The results show that optimization reduces delays by about 7%; this percentage can be further increased by creating indexes for frequently-used BGPs, such as substrate provider region. Complex discovery queries require more processing time and induce much larger delays compared to simple queries (Figure 5.c). An optimization solution for such queries is therefore worth investigation.

IV. Conclusion

We proposed a substrate description framework and semantic repository architecture for cloud-based conferencing substrates. A proof-of-concept prototype was implemented, deployed, and successfully tested. The performance results for the proposed architecture delivers satisfactory results for publication and discovery of conference substrates. However, methods for further optimization need to be investigated for complex queries. Our future work is also directed toward extending the already-implemented repository architecture to other providers of the cloud-based conferencing business model.

References

[1] R. H. Glitho, 'Cloud-based Multimedia Conferencing: Business Model, Research Agenda, State-of-the-Art', in *2011 IEEE 13th Conference on Commerce and Enterprise Computing (CEC)*, 2011, pp. 226 –230.

[2] R. Kanagasabai, 'OWL-S Based Semantic Cloud Service Broker', in *Web Services (ICWS), 2012 IEEE 19th International Conference on*, 2012, pp. 560–567.

[3] Jos de Bruijn and Dieter Fensel, 'Web Service Modeling Language (WSML) - W3C Submission'. [Online]. Available: http://www.w3.org/Submission/WSML/. [Accessed: 18-Jan-2013].

[4] J. Li and F. Yang, 'Resource-Oriented converged network service modeling', in *Communications Technology and Applications, 2009. ICCTA'09. IEEE International Conference on*, 2009, pp. 895–899.

[5] N. Loutas, E. Kamateri, and K. Tarabanis, 'A Semantic Interoperability Framework for Cloud Platform as a Service', in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 280–287.

[6] Y. Cui, C. Chen, and Z. Zhao, 'Web Service Selection Based on Credible User Recommended and QoS', in *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, 2012, pp. 637–642.