Ontology Integration for Advanced Manufacturing Collaboration in Cloud Platforms

Shravya Ramisetty[1], Prasad Calyam[1], J. Cecil[2], Amit Rama Akula[1], Ronny Bazan Antequera[1], Ray Leto[3]
[1]University of Missouri-Columbia, [2]Oklahoma State University, [3]TotalSim USA
Email: *calyamp@missouri.edu*; {*srd6c*, *ar442*, *rcb553*}*@mail.missouri.edu*; *j.cecil@okstate.edu*; *rleto@totalsim.us*

***Abstract*—Advances in the field of cloud computing and networking have led to rapid development and market growth in areas such as online retail, gaming and healthcare. In the field of advanced manufacturing however, the impact has been significantly lesser than expected due to limitations in cloud platforms for fostering community engagement. To address this problem, we study a new cloud-based architecture that provides Platform-as-a-Service (PaaS) management capabilities to the manufacturing community for delivering Software-as-a-Service (SaaS) "Apps" to their customers. Our architecture aims at supporting an "App Marketplace" that thrives on agile development, organic collaboration and scalable sales of next generation manufacturing Apps requiring high-performance simulation and modeling.**

Towards realizing the vision of the above architecture, our paper involves investigation and implementation of an Ontology Service that interoperates with other common web services related to resource brokering and accounting. Our Ontology Service uses principles of mapping and merging to translate a manufacturing App's collaboration requirements to suitable resource specifications on public cloud platforms. Integrated resultant ontology can be queried to provision the required resource parameters such as amount of memory/storage, number of processing units, and network protocol configurations needed for deployment of an App. We validate the effectiveness of our Ontology Service using the Protégé framework in a pilot testbed of a real-world "WheelSim" App in the NSF GENI Cloud platform. Our ontology integration results show benefits to an App developer in terms of: optimal user experience, lower design time and lower cost/simulation.

***Keywords*-Integrated Cloud Management, Advanced Manufacturing, Ontology Service, App Marketplace**

I. Introduction

Advances in the field of cloud computing and networking have led to rapid development and market growth in areas such as online retail, gaming and health care. These advancements have helped in creating economic benefits by transforming investment in infrastructure. Enterprises can now rent infrastructure on-demand without the hassle of frequent maintenance or upgrades. They can also access high performance computing and elastic resources to collaborate with their peers and improve service delivery to customers [1]. An exemplar use case of cloud services adoption can be seen in a health care industry study [2], where a secure cloud environment was leveraged to manage information and foster collaboration between emergency health care professionals.

In the field of advanced manufacturing however, the adoption and benefits of cloud computing have been significantly lesser than expected. This is due to limitations in cloud platforms that are not capable of fostering community engagement for advanced manufacturing design (e.g., fluid/thermal analyses), which typically requires collaborative work among multi-site engineering experts. To leverage cloud services and minimize cost, there is also a need to transform traditional workflows that feature data-intensive computation and networking during design and development of new domain applications.

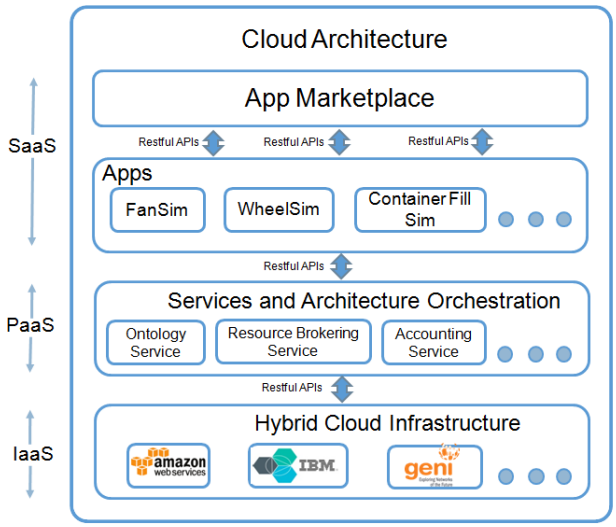

Fig. 1. App Marketplace Cloud architecture

To address these problems, the advanced manufacturing community should adopt a new cloud-based architecture that provides Platform-as-a-Service (PaaS) management capabilities, for delivering Software-as-a-Service (SaaS) "Apps" to their customers. Such an architecture should aim at supporting an "App Marketplace" that thrives on agile development, organic collaboration and scalable sales of new manufacturing Apps requiring easy access to simulation and modeling resources.

Our envisioned cloud architecture is broadly illustrated in Figure 1 that enables client requests to be provisioned from any one of the cloud providers such as Amazon, IBM, Google or GENI [3]. To deploy an App development environment, we need to be able to gather requirements, allocate resources and provide a means for the clients to keep track of resource usage. We identify three core platform services that are essential to perform these tasks, namely: (i) Ontology Service, (ii) Resource Brokering Service, and (iii) Accounting Service. These services can deploy a development environment for

This material is based upon work supported by the City of Dublin, Mozilla Foundation and National Science Foundation under award numbers CNS-1347889, CNS-0714770. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the City of Dublin, Mozilla Foundation or National Science Foundation.

engineers to collaborate with their clients, and allow them to use this environment for agile development of manufacturing Apps. When multiple Apps that work together are developed (e.g., WheelSim that needs to work with TruckSim), they can be integrated within an App Marketplace.

Towards realizing the vision of the above architecture, this paper focuses mainly on the investigation and implementation of an Ontology Service that interoperates with other common App marketplace services. Our Ontology Service uses principles of mapping and merging [4] to *translate manufacturing App collaboration requirements to suitable resource specifications on public cloud platforms*. We construct ontologies based on the domain and related abstractions that are: (a) manufacturing domain specific, and (b) span generic App resource requirements.

One of the challenges that need to be addressed in our approach is the problem of constant ontology evolvement; new understanding of an existing concept or a new concept itself might be required to be included into the ontology through a versioning method. Hence, we incorporate a 'feedback loop' in our workflow steps to trace and maintain the ontologies as "Spec Templates" in a consistent and coherent manner to cope with the dynamic nature of related information.

For the study purposes of this paper, we show how a custom App viz., "WheelSim" can be realized with our ontology integration approach that uses the Protégé framework [4]. We describe how a WheelSim App was successfully built on the development and deployment environment created within the NSF-supported Global Environment for Network Innovations (GENI) public cloud using our approach. In our Ontology Service effectiveness validation, we use a publicly available GENI ontology and App platform requirements of WheelSim to create ontologies that could be integrated and queried. Lastly, we validate the effectiveness of our Ontology Service through our ontology integration results that show benefits to an App developer in terms of: optimal user experience, lower design time and lower cost/simulation. Overall, our work in this paper helps bridge the collaboration gap between manufacturing App developers and the cloud platform engineers.

The remainder of this paper is organized as follows: Section II details related work. Section III discusses the research problem and our proposed architecture solution. Section IV discusses the Ontology Service. Section V discusses the experiments conducted and the validation results obtained. Section VI concludes the paper.

II. Related Work

There are several cloud-based manufacturing works whose primary focus often is on specific areas of manufacturing such as Machine Tools [7] [11], Micro Assembly [12], and Assembly Modeling systems [13]. Domain ontology which provides the ontology specification for machine tools was used in [7]. However, the domain ontology approach is specific to App requirements, and does not take into account resource brokering aspects. Our approach is to develop a generic method for capturing ontologies for integration in diverse advanced manufacturing App use cases, and select a pertinent cloud service composition through resource brokering.

Authors in [9] study service-oriented ontology learning which is based on text understanding and automatic processing. In this approach, the limitation is that there is lack of interaction in ontology learning process between users. Prior work on cloud-based resource brokering is abundant [14] - [17]. However, only few works focus on ontology-based resource brokering [18] [19] through the use of ontology specification, and most of them use single ontology specifications. In our ontology-based resource brokering, we cover multiple aspects relating to Apps and related resources, and handle merging and mapping of multiple ontologies [4].

There have been earlier successful trials to use cloud infrastructure for meeting advanced manufacturing design needs. In [20], we setup a testbed to explore remote collaboration and access of advanced manufacturing resources using the GENI infrastructure. Other cloud infrastructures have been leveraged as development environments in [21] [22] for building software applications. There are also several cloud-based marketplaces that exist [23] for different domains, and our work builds on similar concepts in the context of advanced manufacturing App collaborations to satisfy customer needs.

III. Problem description

In today's dynamic and rapidly evolving cloud services market, cloud platform providers employ proprietary technologies for common service related tasks and operations. This trend makes it challenging for an App developer to easily request for a cost-effective and suitable resource configuration when choosing between multiple providers. It is therefore critical to develop cloud services with improved data processing and resource customization with inclusion of semantics and relationships for better knowledge exchange of user experience and other performance requirements. Moreover, the 'development' of the App may need a different service configuration than the 'consumption' of the App by customers. For e.g., multi-expert collaboration for manufacturing design will need higher remote access requirements to cloud-hosted software/-data, versus the functional App will need to be consumed in a scalable manner by customers with different end-devices (e.g., PC, tablet) accessing from distributed locations with mobility.

For addressing the above problems, an ontology-based approach would be suitable to foster common understanding of the platform requirements and enable mapping of these requirements to the desired App's underlying cloud resource capabilities. Ontology-based approaches have been successfully used in many application domains to bridge the gap between heterogeneous concepts, and achieve the desired level of semantic interoperability. However, one of the critical requirements for successfully adopting an ontology approach is to have commonly accepted and openly accessible ontology trees within the application community. Unfortunately, cloud service providers such as Amazon or even GENI do not have fully developed ontologies, and their latest ontologies are constantly evolving. New ontologies are being proposed and are being developed such as WSMO (Web Service Modeling Ontology) in the W3C community that aim to address use cases in Amazon; GENI community also has published a compute resources ontology [3] that is being refined in efforts related to Resource Specification (RSpec) for Future Internet experiments. Nevertheless, the challenge is to consistently use both App as well as cloud resource ontologies such that resource configuration can be made easier, and also users can take advantage of market competition in inter-cloud scenarios for more cost-effective resource access.

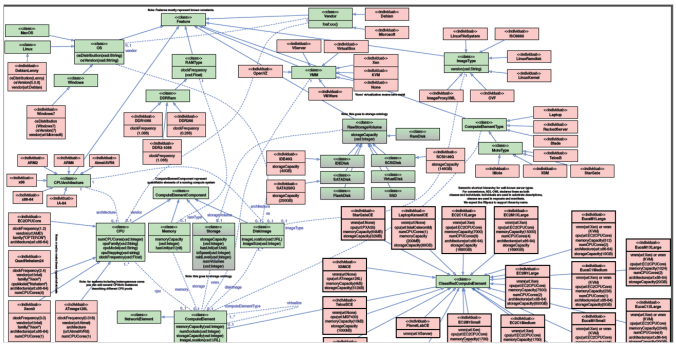

Fig. 2. GENI Resource Schema diagram from [3]

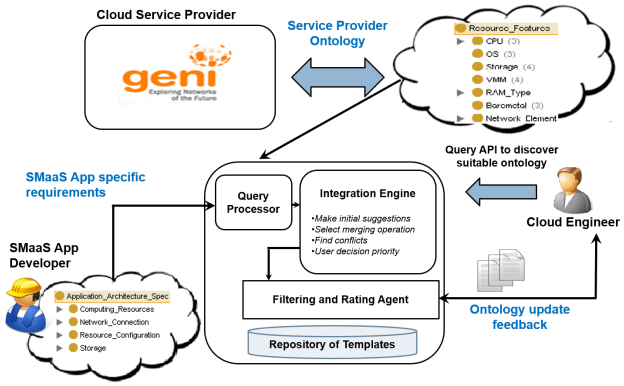

Fig. 3. Ontology Integration Workflow Components

A. *Ontology Integration*

In our ontology creation phase, we assume that a manufacturing user is not necessarily an expert in cloud resource configurations and will take assistance from a cloud engineer for the actual service implementation. There are two main Ontologies that are created to address the semantic interoperability issue. The 'App Spec Ontology' describes the requirements for meeting the development and hosting needs of a given manufacturing App. For our study purposes, we conceive the 'App Spec Ontology' in Section V-A based on the use case of TotalSim, a small business advanced manufacturing company with experts in the USA and UK. Next, we consider a separate 'Platform Capabilities Ontology' that corresponds to the publicly available ontology from a cloud provider. More specifically, Figure 3 shows the schema diagram of the resources provisioned by GENI that we use to create the 'Platform Capabilities Ontology'. Hence these two Ontologies will have to be integrated to generate a merged Ontology which can be queried to extract specifications. These specifications form the "Spec Template" which is useful to deploy the App environment.

B. *Need for Ontology Update*

The ontology integration methodolgy described earlier results in a merged Ontology which is then queried to generate a "Spec Template" that is used to deploy the App environment. This may lead to an assumption that the ontology is static. However as discussed earlier, ontologies need to handle dynamic information and need to constantly evolve to remain useful. Specifically, at any stage of this process, a change in App's requirements or in the Cloud Service Provider's resource capabilities would render the merged ontology useless because querying it would present unreliable and inaccurate results.

To overcome this issue, we incorporate a 'feedback loop' in our final workflow step to update the templates. This feedback would also enable reusing or recycling successfully used prior ontologies for either integration or provisioning of new Apps. As the ontologies are updated, the Spec Templates are versioned and stored for future use of the App or for adaptation to provision new Apps.

IV. Proposed Framework For Ontology Service

In this section, we describe the Ontology Integration Service and discuss its various components.

A. *Ontology Integration Service Components*

This Ontology Integration Service essentially converts the requirements of the App developers into a usable App-Resource ontology. It accomplishes this using two primary functions: (i) combining the App's ontology and Infrastructure Service Provider's ontology to a App-Resource ontology which consists of a taxonomy of concepts of different cloud services to comply with the App-specific requirements, and (ii) finding the best PaaS service options. Figure 5 illustrates how the requirements are gathered from App developers and recorded in the App-specific requirement file which is forwarded to the service components that work together to generate the App ontology. This is then merged with the Service Provider's ontology and ultimately the unique App-Resource ontology is output. The cloud platform engineer can enter queries through a web portal containing the service name and App requirements. The ontology may then be updated if necessary by the cloud engineer based on performance engineering experiments in co-ordination with the App developer. The components of the Ontology Integration Service are shown in Figure 5, which we now describe in detail.

1) *Query Processor (QP)*: The Query Processor parses the query to read the App-specific requirements. It analyzes the requirements and creates the suitable App ontology from existing Spec Templates or Catalog of ontologies specific to the particular manufacturing domain. The resultant App ontology output is forwarded to the Integration Engine.
2) *Integration Engine (IE)*: The Integration Engine analyzes App and Service Provider ontologies and makes suggestions for merging and mapping. It then aligns the App ontology with the Service Provider's ontology using one of the many ontology Mapping & Merging Tools available such as the Protégé framework [4]. If conflicts arise as a result of merging and mapping operations, the service can fix conflicts automatically or prompt the user for suggestion. The resultant App-Resource ontology is next sent to the Filtering and Rating Agent.
3) *Filtering and Rating Agent*: The Filtering Agent sorts through the results to eliminate any duplicate or non-trivial configurations obtained from the result. It stores the newly established ontologies into the database for the Rating Agent to use it as part of a knowledge base. The Rating Agent analyzes the merged ontology

and calculates a rating called "popularity score" based on: (a) comparison of concept labels matching in the given query, (b) success of the development environment testbed, and (c) frequency of use of templates. Depending on the popularity score, it displays the ontology templates in an order of priority. Some templates may be less re-used, which does not necessarily reflect in the quality of the ontologies but may indicate a less-common App specification. The Rating Agent could also consider user satisfaction with any resultant ontology, which can be obtained through feedback obtained from a cloud engineer or App developer.

With the help of the resultant integrated App-Resource ontology, the cloud engineer can build the environment with a Resource Brokering Service. Ontologies can be very time-consuming and expensive to construct. As the use of ontologies for the representation of domain knowledge increases, so will the need for an effective set of tools to aid the discovery and re-use of existing knowledge representations. This is because a major advantage of using ontologies is their ability to be re-used as well as easily adapted to work with new knowledge bases of Apps which may have unique testbed requirements (compute, network, proprietary software etc.).

V. Performance Evaluation

In this section we discuss the requirements for the development and deployment of an actual manufacturing App. Next, we discuss the generation of Spec Template using the Ontology Service and query mechanisms. Following this, we discuss the testbed configuration of our manufacturing App. Finally, we describe the experiments conducted to validate the testbed of the App in the GENI public cloud and discuss the results.

A. WheelSim App Requirements

For our use case, the WheelSim App being developed by TotalSim is a simulation application used to learn about the different lift and drag forces acting on a wheel in motion. The workflow of the App is illustrated in Figure 7, where the customers access the modeling and simulation output of TotalSim experts by presenting familiar sets of inputs in a web-portal. The web-portal uses the App customer input to specify parameters for simulation, schedule and monitor jobs at a cloud-based supercomputer and retrieve simulation results. The requirements for developing and hosting this application for users is to have on-demand access to pre-loaded virtual machines (VMs) with the relevant compute, network and storage environment.

Network connectivity with high-bandwidth is a requirement among the collaboration sites of the WheelSim App that include a remote office in California, Supercomputer Center for high-performance computing (HPC) and customers in North Carolina. The HPC is required for simulation, modeling phases and for storage of model data and results. Computer-Aided Design (CAD) and other software licenses have to be acquired from the TotalSim server at their remote office in Dublin, Ohio. Keeping these requirements as the baseline, we designed an ontology (refer to Figure 8) for the App specification.

Developing an ontology for resource specification of cloud service providers is a very complex and laborious task. It involves tasks such as information collection, resource abstraction, and categorization of data. Fortunately, we were

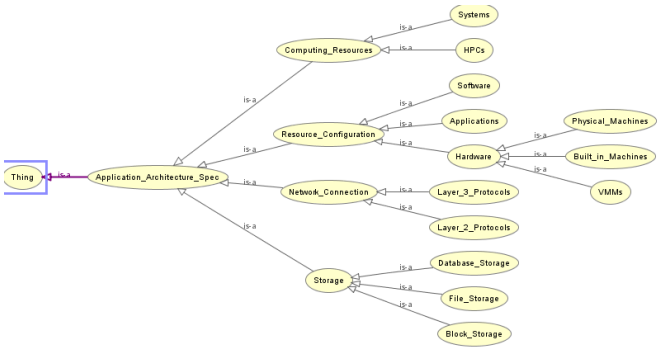

Fig. 5. Class hierarchy for Manufacturing Application Requirements Ontology

Fig. 6. Service Provider Ontology-GENI

able to discover and use the GENI Compute ontology schema available at [3]. For our study, we have designed a minimal version of the basic ontology shown in Figure 10 and use it for the resource specification.

B. Ontology Service and Query

For the use case with TotalSim, we have selected the PROMPT tool (plugin for Protégé framework) in the Integration Engine shown in Figure 5. During the integration process, the Integration Engine analyzes all the classes, their properties, the relationships between them and the restrictions on individual classes. Note that the resultant ontology needs to be kept rational and consistent during the process. The Ontology Service generates a single ontology in one subject from the two existing and different ontologies of different subjects. Since the subjects of these ontologies are related, they can be merged into a single App-Resource ontology.

A cloud engineer can draw inferences from the knowledge in the merged ontology and obtain recommendations of the available options suitable for the App. The cloud engineer should query in a way such that all the possible options for the environment setup are displayed. From the obtained query results, the optimal resources considering performance

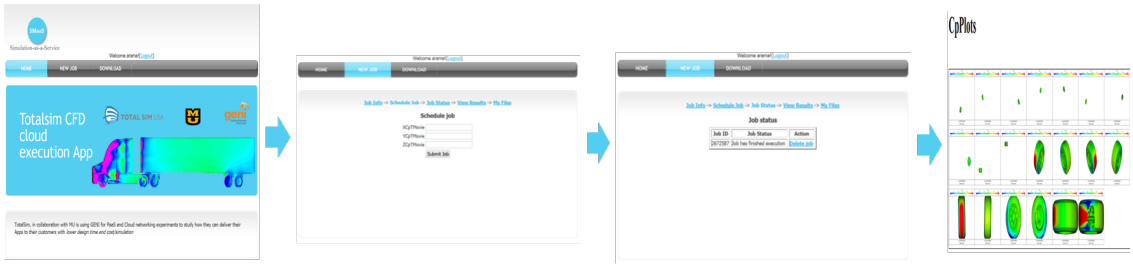

Fig. 4. Workflow of WheelSim App

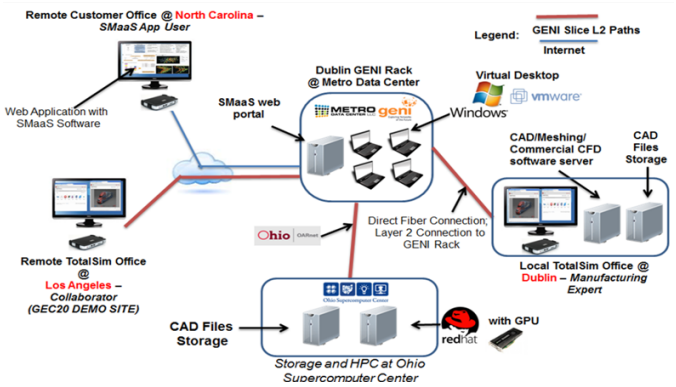

Fig. 7. Hybrid Cloud Testbed setup for TotalSim

versus cost trade-offs are chosen based on the priority. If the resource brokering service is not able to procure the resources for setting up the environment, the secondary options obtained from the query results can be viewed instead. Thus, querying the ontology for compute power, memory, storage and network connectivity will generate the Spec Template for the testbed. Thus, ontology by itself can provide individual resource options and their details with the help of class definitions and object properties and can also be queried to output the various resource configurations. We remark that this approach is better than finding the resource information manually, and can save design time and cost for manufacturing App developers.

C. Testbed Configuration

Assuming role of the cloud engineer, we set up the testbed shown in Figure 14 for the WheelSim App based on the results from Ontology Service. The testbed includes the following components:

1) Public Cloud: A GENI Rack at Metro Data Center (MDC) which acts as the resource broker and provides a network overlay infrastructure
2) Elastic HPC backend at the Ohio Supercomputer Center (OSC) to run jobs charged on a run-by-run basis only making it a cost effective compute resource
3) Layer-2 connectivity established between the GENI Rack, OSC, TotalSim's office and remote collaborator at Los Angeles (LA)
4) A web-portal accessible through Internet is hosted on the GENI Rack to enable customers to run WheelSim App that schedules batch jobs at OSC

The GENI Rack at MDC provides the network overlay infrastructure through a GENI slice. The slice reserved contains a HP bare metal computer with 1 Intel Xeon X5650@2.67GHz processor, 50 GB of RAM and 1 TB of disk capacity. The slice along with a private network and VLAN have been initially reserved for long-standing experiments over several months timeframe. ESXi hypervisor is installed with VMware Horizon as the provisioning management middleware. The installation and administration are done remotely with SSH, RDP, or through VMware tools such as VMware vSphere web-client.

D. Experiments and Results Discussion

1) Quantitative Results: The testbed for the WheelSim App shown in Figure 14 was successfully deployed on GENI. Post deployment of the testbed, remote collaborators used VMs deployed on the testbed to access tools and software that they use as a part of their daily work routines. However, they faced performance issues when software such as Paraview were run on the VMs. To assess and improve the performance of the VMs, we conducted tests for three scenarios: System, virtualization and model location. In the system scenario, we altered the compute configurations (RAM & vCPU) of the VMs. In the virtualization scenario, we measured the performance of a physical machine versus a VM. In the data location scenario, we measured the performance by placing data at a remote location.

To quantify the performance results, we now define the following terms we use:

1) Open Time: The time taken by the Paraview application to open and load the model
2) Render Time: The time taken by the Paraview application to render the model and display it on the screen
3) Running Time: This is the overall time taken by the Paraview application to load and render the model

$$T_{running} = T_{open} + T_{render} \quad (1)$$

a) RAM and vCPU Effects: In this set of experiments, we observed the options available for vCPU and RAM from the query results of the resultant ontology. We varied the RAM and vCPU of the VM through VMware Horizon environment and then measured the Open Time followed by the Render Time and calculated the Running Time.

It can be observed from the Figure 15 that as the number of vCPUs increase, the Running Time is reduced by a few seconds. However, an increase in RAM reduces the Running Time considerably. The maximum variation in Running Time is about 20 seconds from 1 vCPU and 2 GB of RAM to 8 vCPUs and 32 GB of RAM. This experiment allows us to carefully tune the configuration of the VM to enhance user experience.

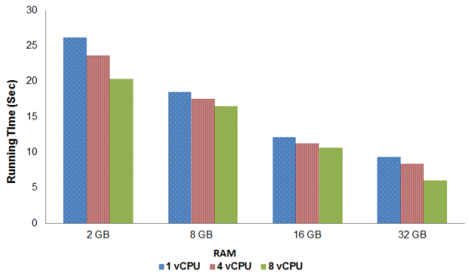

Fig. 8. Performance results with varying vCPUs and RAM

b) Virtualization Effects: In this set of experiments, we compared the performance of a physical machine versus a VM. We compared the Open Time and Render Time between physical machine and a VM for the same model. In this case for machines with similar configuration, the Open and Render Time for a VM were very close to the physical machine thereby ensuring that user experience remains same whether they use a VM or physical machine as shown in Table I.

TABLE I
PERFORMANCE COMPARISON OF PHYSICAL AND VIRTUAL MACHINES

Machine type	Open Time	Render Time
Native Physical machine	9.43s	3.61s
Virtual machine	10.13s	3.78s

c) Data Location Effects: The above results measure the response when the model is at the same location as the machine. There are cases where in a model is in a remote location and will be accessed over the network. To test this scenario, we used the GENI Rack as the remote storage and a user accessed a very large model with file size of 3.8 GB from both a physical system and a VM. The access time from the physical system was high with 285 seconds in comparison to the VM which only took 26 seconds. This could be attributed to the fact that the VMs are located at the GENI Rack thereby reducing the access time. Since the VMs provide the same utility as the physical systems but provide better performance in this context, they are suitable to be used for collaborating remotely. Hence, our study helped us refine the specifications of the ontology with the Ontology Service.

d) Discussion: We can observe from the results that by provisioning a VM with an appropriate compute configuration and by placing data adjacent to the VMs, the performance is very similar and sometimes better than an actual workplace desktop. These results not only enhance the user experience and collaboration but also encourage manufacturing enterprises to migrate to the cloud. Additionally, the performance study helped us to identify the optimal configuration for specific scenarios to improve the user experience. The results are hence used to update the ontology with more detailed rules and modified axioms. The additional information updated into the ontology makes it more precise, and this reverse feedback helps in constructing robust Spec Templates.

2) Qualitative Results: If the entire process was done following the traditional development life cycle of advanced manufacturing, there would be several delays in the design cycle, increased cost and overhead for productivity. In the traditional enterprise environment, collaboration is asynchronous as screen capture/e-mail programs are used to communicate information. Data duplicity and version control management also are arduous tasks. Connecting to external HPC resources and transferring data over public Internet would be a slow process due to network congestion and last-mile network bottlenecks. Further, local cluster configuration is time consuming and may lead to over provisioning, thereby resulting in inefficient resource management.

The advantages of using our new proposed architecture would enable an agile development lifecycle that ensures quicker delivery, reduced cost and increased productivity. This would be facilitated through synchronous collaboration and smooth cross platform operation. Utilization of cloud storage at OSC, Box.net prevents duplication of data common when using e-mail programs for data exchange, and also ensures data security and integrity. The elasticity of cloud platforms could dynamically handle on-demand bursts for compute resources when needs of App developers or customers changes. Lastly, bulky data obtained from simulation results can be transferred over extended VLANs with high-speed network connections, which in turn saves time.

VI. CONCLUSION

A cloud architecture for advanced collaborative manufacturing has been presented in this paper that allows agile methods to be adopted in this domain. This architecture leverages public cloud infrastructure to provide scalable and on-demand resources in a cost-effective manner. It encompasses three services at the platform level: Ontology Service, Resource Brokering Service and Accounting Service.

In our study, we used ontology concepts to bridge the semantic gap between manufacturing App developers and cloud service providers. We developed an Ontology Service that translates the App requirements of the manufacturing domain to the development and hosting environment specifications. The service functioned by merging the ontology of the App requirements with the ontology of the cloud provider resulting in the App-Resource ontology.

We successfully used the resultant ontology to generate a testbed environment for a WheelSim App. We also discussed a performance study that measures the performance of VMs instantiated by the testbed. We identified core parameters that improved the performance of the VM i.e. RAM, vCPUs and data location when software such as ParaView are used in the App design. We used results obtained from this performance study to update the ontology, establishing a reverse feedback loop which enhances the ontology resulting in construction of robust templates for future re-use or adaptation in similar App contexts.

As part of the future work, we plan to investigate how multiple Apps in an App marketplace in the manufacturing domain work can together in the form of plug-and-play modules by forwarding results of execution from one App to another. This would create a coherent environment for an App Marketplace that uses our novel services suitable for development and scalable sales of advanced manufacturing related Apps. Ultimately, such App Marketplaces can allow rapid innovations in manufacturing products in important sectors such as automobiles and even pipes.

REFERENCES

[1] J. Yu, J. Ni, "Development Strategies for SME E-Commerce Based on Cloud Computing", *International Conference on Internet Computing for Engineering and Science (ICICSE)*, 2013.

[2] V. Koufi, F. Malamateniou, G. Vassilacopoulos, A. Prentza, "An Android-enabled Mobile Framework for Ubiquitous Access to Cloud Emergency Medical Services",*Symposium on Network Cloud Computing and Applications (NCCA)*, 2012.

[3] GENI Wiki - http://groups.geni.net

[4] H. Sun, W. Fan, W. Shen, T. Xiao, "Ontology Fusion in High-Level-Architecture-Based Collaborative Engineering Environments", *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, Vol. 43, No. 1, pp. 2-13, 2012.

[5] M. Saeki, "Ontology-based Software Development Techniques", *ERCIM News*, No. 58, 2004.

[6] NDL-OWL Models in ORCA - https://geni-orca.renci.org/trac/wiki/NDL-OWL

[7] M. Hung, Y. Lin, H. Huang, M. Hsieh, H. Yang, F. Cheng, "Development of an Advanced Manufacturing Cloud for Machine Tool Industry based on AVM Technology", *Proc. of IEEE CASE*, 2013.

[8] H. Lan, "A Web-based Rapid Prototyping Manufacturing System for Rapid Product Development", *Collaborative Design and Planning for Digital Manufacturing*, 2009.

[9] T. Zhang, S. Ying, S. Cao, X. Jia, "A Modeling Framework for Service-Oriented Architecture, Quality Software", *Proc. of QSIC*, 2006.

[10] N. Gobinath, J. Cecil, T. Son, "A Collaborative System to realize Virtual Enterprises using 3APL, Declarative Agent Languages and Technologies IV", *Springer Lecture Notes in Artificial Intelligence*, 2006.

[11] M. Hung, Y. Lin, T. Huy, H. Yang, F. Cheng, "Development of a Cloud-Computing-based Equipment Monitoring System for Machine Tool Industry", *Proc. of IEEE CASE*, 2012.

[12] J. Cecil, R. Gunda, P. Calyam, S. Seetharam, "A Next Generation Collaborative Framework for Advanced Manufacturing", *Proc. of IEEE CASE*, 2013.

[13] C. Wang, Z. Bi, L. Xu, "IoT and Cloud Computing in Automation of Assembly Modeling Systems", *IEEE Trans. on Industrial Informatics*, Vol. 10, No. 2, pp.1426 - 1434, 2014.

[14] A. Prasad, S. Rao, "A Mechanism Design Approach to Resource Procurement in Cloud Computing", *IEEE Trans. on Computers*, Vol. 63, No. 1, 2014.

[15] Y. Laili, F. Tao, L. Zhang, L. Ren, "The Optimal Allocation Model of Computing Resources in Cloud Manufacturing Systems", *Proc. of Intl. Conference on Natural Computation*, 2011.

[16] S. Sotiriadis, N. Bessis, N. Antonpoulos, "Decentralized meta-brokers for inter-cloud: Modeling brokering coordinators for interoperable resource management", *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2012.

[17] B. Ray, S. Khatua, S. Roy, "Negotiation based service brokering using game theory", *Applications and Innovations in Mobile Computing (AIMoC)*, 2014.

[18] T. Han, K. Sim, "An Ontology-enhanced Cloud Service Discovery System", *Intl. MultiConference of Engineers and Computer Scientists (IMECS)*, 2010.

[19] A. Amato, G. Cretella, B. Martino, S. Venticinque, "Semantic and Agent Technologies for Cloud Vendor Agnostic Resource Brokering", *Intl. Conference on Advanced Information Networking and Applications Workshops*, 2013.

[20] A. Berryman, P. Calyam, J. Cecil, G. Adams, D.Comer, "Advanced Manufacturing use Cases and Early Results in GENI infrastructure", *Proc. of GENI Research and Educational Experiment Workshop (GREE)*, 2013.

[21] D. Nasui, V. Sgarciu, A. Cernian, "Cloud-Based Application Development Platform for Secure, Intelligent, Interlinked and Interactive Infrastructure", *IEEE International Symposium on Applied Computational Intelligence and Informatics*, 2013.

[22] L. Cocco, K. Mannaro, G. Concas, "A Model for Global Software Development with Cloud Platforms", *Euromicro Conference on Software Engineering and Advanced Applications*, 2012.

[23] G. Modica, G. Petralia, O. Tomarchio, "Procurement Auctions to Trade Computing Capacity in the Cloud", *Intl. Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2013.