Toward a Unified Ontology of Cloud Computing

Lamia Youseff
University of California, Santa Barbara
Santa Barbara, CA 93106
lyouseff@cs.ucsb.edu

Maria Butrico, Dilma Da Silva
IBM T.J. Watson Research Center
Yorktown, New York 10598
{butrico, dilmasilva}@us.ibm.com

***Abstract*—Progress of research efforts in a novel technology is contingent on having a rigorous organization of its knowledge domain and a comprehensive understanding of all the relevant components of this technology and their relationships. Cloud Computing is one contemporary technology in which the research community has recently embarked. Manifesting itself as the descendant of several other computing research areas such as Service-Oriented Architecture, distributed and grid computing, and virtualization, cloud computing inherits their advancements and limitations.**

Towards the end-goal of a thorough comprehension of the field of cloud computing, and a more rapid adoption from the scientific community, we propose in this paper an ontology of this area which demonstrates a dissection of the cloud into five main layers, and illustrates their inter-relations as well as their inter-dependency on preceding technologies. The contribution of this paper lies in being one of the first attempts to establish a detailed ontology of the cloud. Better comprehension of the technology would enable the community to design more efficient portals and gateways for the cloud, and facilitate the adoption of this novel computing approach in scientific environments. In turn, this will assist the scientific community to expedite its contributions and insights into this evolving computing field.

I. Introduction

Although the term "Cloud Computing" is based on a collection of many old and few new concepts in several research fields like Service-Oriented Architectures (SOA), distributed and grid computing as well as virtualization, it has created much interest in the last few years. This was a result of its huge potential for substantiating other technological advances while presenting a superior utilitarian advantage over the currently under-utilized resources deployed at data centers. In this sense, cloud computing can be considered a new computing paradigm that allows users to temporary utilize computing infrastructure over the network, supplied as a service by the cloud-provider at possibly one or more levels of abstraction. Consequently, several business models rapidly evolved to harness this technology by providing software applications, programming platforms, data-storage, computing infrastructure and hardware as services. While they refer to the core cloud computing services, their inter-relations have been ambiguous and the feasibility of enabling their inter-operability has been debatable. Furthermore, each cloud computing service has a distinct interface and employ a different access protocol. A unified interface to provide integrated access to cloud computing services is nonexistent, although portals and gateways can provide this unified web-based user interface.

Equally important, the current state of the art in cloud computing research lacks the understanding of the classification of the cloud systems, their correlation and inter-dependency. In turn, this obscurity is hindering the advancement of this research field. As a result, the introduction of an organization of the cloud computing knowledge domain, its components and their relations – i.e. an ontology – is necessary to help the research community achieve a better understanding of this novel technology.

In this paper, we present a summary of our assimilation of cloud computing, with a classification of its components, and their relationships as well as their dependency on some of the prior concepts from other fields in computing. In addition, we highlight some of the technical challenges involved in building cloud components in each layer of our proposed ontology, and how these challenges were addressed in the predecessor areas of computing research. We also underline the constraints associated with specifying a user interface for each layer.

The rest of this paper is organized as follows. Section II discusses the motivation for this study in light of the limitations and imperfections of current cloud systems. Section III addresses the classification methodology we used to design our proposed ontology of the field, while section IV examines each layer of the ontology, elaborating on its composition, limitations, and correlation with other layers. We conclude our paper with a discussion on several contemporary challenges for cloud computing research, and the various hinders delaying its wide adoption in scientific computing in section V and recapitulate our work in section VI.

II. Motivation

The recent evolution of cloud computing has borrowed its basics from several other computing areas and systems engineering concepts. Cluster and Grid Computing on one hand, and virtualization on the other hand are perhaps the most obvious predecessor technologies that enabled the inception of cloud computing. However, several other computing concepts have indirectly shaped today's cloud computing technology, including peer-to-peer (P2P) computing [1], SOA [2] and autonomic computing [3]. MapReduce [4] is an example of programming models that demonstrated a simpler way to develop data-intensive applications for large distributed systems, which can be leveraged to utilize resources available through the cloud. In this respect, a fundamental understanding of the extent that cloud computing inherits its concepts from these various computing areas and models is essential to understanding the landscape of this novel computing field and defining its potentials and limitations. Such comprehension will facilitate further maturation of the area while enabling innovation of novel systems to allow more creative uses of the cloud. In order to comprehend the impact of those various concepts on cloud computing, our approach is to determine the different layers and components that make the cloud, and study their characteristics in light of their dependency on the other computing fields and models. In this vein, it was important to define the constituents of cloud computing in order to define its strengths and imperfections with respect of its adoption of previous technological concepts.

Besides, an ontology of cloud computing will allow better understanding of the inter-relations between the different cloud components, enabling the composition of new systems from existing components and further re-composition of current systems from other cloud components for desirable features like extensibility, flexibility, availability or merely optimization and better cost-efficiency. We as well postulate that understanding the different components of the cloud will allow system engineers and researchers to deal with other hard technological challenges. For example, comprehending the relationship between different cloud systems can accentuate opportunities to design interoperable systems between different cloud offerings that provide higher-availability guarantees. Although high availability is one of the fundamental design features of every cloud offering, failures are not uncommon. Therefore, highly-available cloud applications can be constructed, for example by deploying them on two competitive cloud offerings; e.g. Google App engine [5] and Amazon's EC2 [6]. Even in the case that one of the two clouds fails, the other cloud will continue to support the availability of the applications. In brief, understanding the cloud components may enable creative solutions to common cloud computing problems like availability, application migration between cloud offerings, and system resilience. The broad objective of this classification is to attain a better understanding of cloud computing, and define key issues in current systems as well as accentuate some of the research topics that need to be addressed in such systems.

Not only can an ontology impact the research community, but it also can simplify the educational efforts in teaching cloud computing concepts for students and new cloud applications' developers. Understanding the implications of developing cloud applications against one cloud layer versus another will equip developers with the knowledge to make informed decisions about their applications' expected time-to-market, programming productivity, scaling flexibility as well as performance bottlenecks. In this regard, an ontology can facilitate the adoption of cloud computing.

III. Classification Methodology

In order to define an ontology of cloud computing, we had to choose a methodology to classify the different cloud systems. We opt to use composability as our methodology, as it enables our proposed ontology to capture the inter-relations between the different cloud components. Although we borrow this method from the principle of composibility in SOA, we use it here in a limited fashion to refer to the ability to compose one cloud service from one or more other cloud services.

For simplicity, we define our proposed ontology to be envisioned as a stack of layers. Each layer encompasses one or more cloud services. Cloud services belong to the same layer if they have equivalent levels of abstraction, as evident by their targeted users. For example, all cloud software environments (a.k.a. cloud platforms) target programmers, while cloud applications target end users. Therefore, cloud software environments would be classified in a different layer than cloud applications.

By composability, we classify one cloud layer to be higher in the cloud stack, if its services can be composed from the services of the underlying layer. Consider the cloud application layer as an example. Since we can develop cloud applications using cloud software environments, we say that cloud applications are composable from cloud software environments, and that the cloud application layer is higher in the cloud stack. Using this rationale, we formulate our proposed cloud ontology as shown in figure 1.

Capturing cloud computing systems as composable services allows researchers in cloud computing to define a more robust interaction model between the different cloud entities, on both the functional and semantic levels. Further, it facilitates recognizing the inter-dependency between the different cloud systems, which in turn, accent opportunities for collaboration between different services and enhance QoS based analysis techniques that can result in better guarantees for the service levels of the different cloud systems.

IV. The Cloud Ontology

Cloud computing systems fall into one of five layers: applications, software environments, software infrastructure, software kernel, and hardware. Obviously, at the bottom of the cloud stack is the hardware layer which is the actual physical components of the system. Some cloud computing offerings have built their system on subleasing the hardware in this layer as a service, as we discuss in subsection IV-E. At the top of the stack is the cloud application layer, which is the interface of the cloud to the common computer users through web browsers and thin computing terminals. We closely examine the characteristics and limitations of each of the layers in the next five subsections.

A. Cloud Application Layer

The cloud application layer is the most visible layer to the end-users of the cloud. Normally, the users access the services provided by this layer through web-portals, and are sometimes required to pay fees to use them. This model has recently proven to be attractive to many users, as it alleviates the burden of software maintenance and the ongoing operation and support costs. Furthermore, it exports the computational work from the users' terminal to data centers where the cloud applications are deployed. This in turn lessens the restrictions on the hardware requirements needed at the users' end, and allows them to obtain superb performance to some of their cpu-intensive and memory-intensive workloads without necessitating huge capital investments in their local machines.

As for the providers of the cloud applications, this model even simplifies their work with respect to upgrading and testing the code, while protecting their intellectual property. Since a cloud application is deployed at the provider's computing infrastructure (rather than at the users' desktop machines), the developers of the application are able to roll smaller patches to the system and add new features without disturbing the users with requests to install major updates or service packs. Configuration and testing of the application in this model is arguably less-complicated, since the deployment environment becomes restricted, i.e., the provider's data center. Even with respect to the provider's margin of profit, this model supplies the software provider with a continuous flow of revenue, which might be even more profitable on the long run. This model conveys several favorable benefits for the users and providers of cloud applications, and is normally referred to as *Software as a Service (SaaS)*. Salesforce Customer Relationships Management (CRM) system [7] and Google Apps [8] are two examples of *SaaS*. As such, the body of research on SOA has numerous studies on composable IT services which have direct application to providing and composing *SaaS*.

Our proposed ontology illustrates that cloud applications can be developed on the cloud software environments or infrastructure components (as discussed in the next two subsections). In addition, cloud applications can be composed as a service from other cloud services offered by other cloud systems, using the concepts of SOA. For example, a payroll application might use another accounting SaaS to calculate the tax deductibles for each employee in its system without having to implement this service within the payroll software. In this respect, the cloud applications targeted for higher layers in the stack are simpler to develop and have a shorter time-to-market. Furthermore, they become less error-prone since all their interactions with the cloud are through pre-tested APIs. Developed for a higher cloud-stack layer, the flexibility of the applications is however limited and this may restrict the developers' ability to optimize their applications' performance.

Despite all the advantageous benefits of this model, several deployment issues hinder its wide adoption. Specifically, the security and availability of the cloud

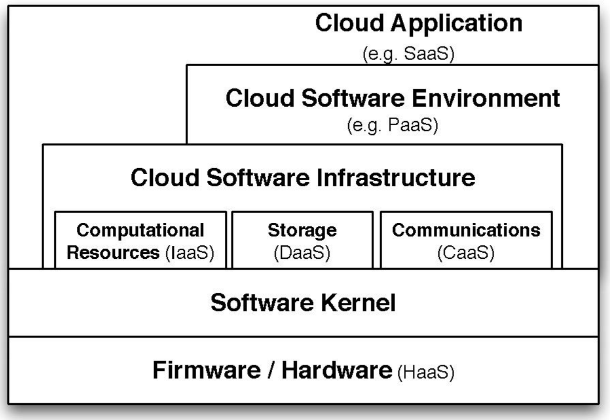
Cloud Application
(e.g. SaaS)
Cloud Software Environment
(e.g. PaaS)
Cloud Software Infrastructure
Computational Resources (IaaS)
Storage (DaaS)
Communications (CaaS)
Software Kernel
Firmware / Hardware (HaaS)

Fig. 1. Our Proposed Cloud Computing Ontology: depicted as five layers, with three constituents to the cloud infrastructure layer. The layered figure represents the inter-dependency and composability between the different layers in the cloud.

applications are two of the major issues in this model, and they are currently avoided by the use of lenient service level agreements (SLA). Furthermore, coping with outages is a realm that users and providers of *SaaS* have to tackle, especially with possible network outage and system failures. Additionally, the integration of legacy applications and the migration of the users' data to the cloud is another matter that is also slowing the adoption of *SaaS*. Before they can persuade users to migrate from desktop applications to cloud applications, cloud applications' providers need to address end-users' concerns about security and safety of storing confidential data on the cloud, users authentication and authorization, up-time and performance, as well as data backup and disaster recovery and provide reliable SLAs for their cloud applications.

B. Cloud Software Environment Layer

The second layer in our proposed cloud ontology is the cloud software environment layer (also dubbed the software platform layer). The users of this layer are cloud applications' developers, implementing their applications for and deploying them on the cloud. The providers of the cloud software environments supply the developers with a programming-language-level environment with a set of well-defined APIs to facilitate the interaction between the environments and the cloud applications, as well as to accelerate the deployment and support the scalability needed of those cloud applications. The service provided by cloud systems in this layer is commonly referred to as *Platform as a Service (PaaS)*. One example of systems in this category is Google's App Engine [5], which provides a python runtime environment and APIs for applications to interact with Google's cloud runtime environment. Another example is SalesForce Apex language [9] that allows the developers of the cloud applications to design, along with their applications' logic, their page layout, workflow, and customer reports.

Developers reap several benefits from developing their cloud application for a cloud programming environment, including automatic scaling and load balancing, as well as integration with other services (e.g. authentication services, email services, user interface) provided to them through the *PaaS*-provider. In such a way, much of the overhead of developing cloud applications is alleviated and is handled at the environment level. Furthermore, developers have the ability to integrate other services to their applications on-demand. This in turn makes the cloud application development a less complicated task, accelerates the deployment time and minimizes the logic faults in the application. In this respect, a Hadoop [10] deployment on the cloud would be considered a cloud software environment, as it provides its applications' developers with a programming environment, i.e. map reduce framework for the cloud. Similarly, Yahoo's Pig [11], a high-level language to enable processing of very large files on the hadoop environment may be viewed as an open-source implementation of the cloud platform layer. As such, cloud software environments facilitate the process of the development of cloud applications.

C. Cloud Software Infrastructure Layer

The cloud software infrastructure layer provides fundamental resources to other higher-level layers, which in turn can be used to construct new cloud software environments or cloud applications. Our proposed ontology reflects the fact that the two highest levels in the cloud stack can bypass the cloud infrastructure layer in building their system. Although this bypass can enhance the efficiency of the system, it comes at the cost of simplicity and development efforts.

Cloud services offered in this layer can be categorized into: computational resources, data storage, and communications.

1) Computational Resources:

Virtual machines (VMs) are the most common form for providing computational resources to cloud users at this layer, where the users get finer-granularity flexibility since they normally get super-user access to their VMs, and can use it to customize the software stack on their VM for performance and efficiency. Often, such services are dubbed *Infrastructure as a Service (IaaS)*. Virtualization is the enabler technology for this cloud component, which allows the users unprecedented flexibility in configuring their settings while protecting the physical infrastructure of the provider's data center. Recent advances in OS Virtualization have made the concept of *IaaS* plausible. This was specifically enabled by two virtualization technologies: paravirtualization and hardware-assisted virtualization. Although both virtualization technologies have addressed performance isolation between virtual machines contending on common resources, performance interference between VMs sharing the same cache and TLB hierarchy cannot yet be avoided [12]. Further, the emergence of multicore machines into mainstream servers exacerbate this performance interference problem. In turn, the lack of a strict performance isolation between VMs sharing the same physical node has resulted in the inability of cloud providers to give strong guarantees for performance to their clients. Instead, they offer them unsatisfactory SLAs in order to provide a competitive pricing for the service. Such weak guarantees, unfortunately, can inject themselves up the layers of the cloud stack, and affect the SLAs of the cloud systems built above the IaaS's SLAs. Amazon's Elastic Compute Cloud (EC2[6]), and Enomalism elastic computing infrastructure [13] are arguably the two most popular examples of commercial systems available in this cloud category. In this space, there are also several academic open-source cloud projects [14], [15].

2) Data Storage:

The second infrastructure resource is data storage, which allows users to store their data at remote disks and access them anytime from any place. This service is commonly known as *D*ata-Storage as a *S*ervice (*DaaS*), and it facilitates cloud applications to scale beyond their limited servers.
Data storage systems are expected to meet several rigorous requirements for maintaining users' data and information, including high availability, reliability, performance, replication and data consistency; but because of the conflicting nature of these requirements, no one system implements all of them together. For example, availability, scalability and data consistency can be regarded as three conflicting goals. While those features are hard to be met with general data storage systems, *DaaS*-providers have taken the liberty of implementing their system to favor one feature over the others, while indicating their choice through their SLA. These implementations have borrowed their fundamental ideas from proceeding research and production systems. Some examples of data storage systems are: distributed file systems (e.g., GFS [16]), replicated relational databases *(RDBMS)* (e.g., Bayou [17]) and key-value stores (e.g., Dynamo [18]). RDBMS, for example opt to present a stricter consistency model at the cost of the availability of the data, while key-value stores have placed more importance on the availability of the data while relaxing the consistency model for the storage. In this respect, the cloud *DaaS* has inherited the different characteristics of today's data storage systems. Example of commercial *DaaS*-systems are Amazon's S3 [19] and EMC Storage Managed Service [20].

3) Communication:

As the need for a guaranteed quality of service (QoS) for network communication grows for cloud systems, communication becomes a vital component of the cloud infrastructure. Consequently, cloud systems are obliged to provide some communication capability that is service-oriented, configurable, schedulable, predictable, and reliable. Towards this goal, the concept of Communication as a Service (CaaS) emerged to support such requirements, as well as network security, dynamic provisioning of virtual overlays for traffic isolation or dedicated bandwidth, guaranteed message delay, communication encryption, and network monitoring. Although this model is the least discussed and adopted cloud service in the commercial cloud systems, several research papers and articles [21], [22], [23] have investigated the various architectural design decisions, protocols and solutions needed to provide QoS communications as a service. One recent example of systems that belong to CaaS is Microsoft Connected Service Framework (CSF) [24]. VoIP telephone systems, audio and video conferencing as well as instant messaging are candidate cloud applications that can be composed of CaaS and can in turn provide composable cloud solutions to other common applications.

Several common design features are shared between the three infrastructure components. To name a few, security of the services, their availability and quality are among the most commonly addressed concerns for these cloud infrastructure components. Encryption through X509 certificates is customary for these systems. However, providing other security mechanisms for service-oriented architectures is a rich area of research with little focus so far from the SOA and security communities.

User-interface to the cloud infrastructure components varies substantially from one system to another. SOAP and REST are examples of interface protocols used with some cloud computational resources. However, an encompassing characteristic of all cloud software infrastructure constituents is their service-oriented implementation. Therefore, designing a unified interface, and interacting with it through a web portal that communicates with the web services can be a plausible approach to providing a standardized interface to the cloud services at this level of the cloud.

D. Software Kernel

This cloud layer provides the basic software management for the physical servers that compose the cloud. Software kernels at this level can be implemented as an OS kernel, hypervisor, virtual machine monitor and/or clustering middleware. Customarily, grid computing applications were deployed and run on this layer on several interconnected clusters of machines. However, due to the absence of a virtualization abstraction in grid computing, jobs were closely tied to the actual hardware infrastructure and providing migration, checkpointing and load balancing to the applications at this level was always a complicated task.

The two most successful grid middleware that harness the physical resources to provide a successful deployment environment for Grid applications were arguably Globus [25] and Condor [26]. The body of research in grid computing is enormous, and several grid-developed concepts are realized today in cloud computing. However, additional grid computing research can potentially be integrated to the research area of the cloud. For example, Grid computing micro-economics models [27] are possible initial models to study the issues of pricing, metering and supply-demand equilibrium of the computing resources in the realm of cloud computing. Those grid economic models could further assist in the transition from the current offering of cloud computing to a national infrastructure for utility computing.

The scientific community has also addressed the quest of building grid portals and gateways for grid environments through several approaches [28], [29], [30], [31], [32], [33]. Such approaches and portal design experiences are further essential for the development of usable portals and interfaces for the cloud at different software layers. In this respect, cloud computing can benefit from the different research directions that the grid community has embarked for almost a decade of grid computing research.

E. Hardware and Firmware

The bottom layer of the cloud stack in our proposed ontology is the actual physical hardware and switches that form the backbone of the cloud. In this regard, users of this layer of the cloud are normally big enterprises with huge IT requirements in need of subleasing *H*ardware as a *S*ervice (*HaaS*). For that, the *HaaS* provider operates, manages and upgrades the hardware on behalf of its consumers, for the life-time of the sublease. This model is advantageous to the enterprise users, since they do not need to invest in building and managing data centers. Meanwhile, *HaaS* providers have the technical expertise as well as the cost-effective infrastructure to host the systems. One of the early examples *HaaS* is Morgan Stanley's sublease contract with IBM in 2004 [34]. SLAs in this model are more strict, since enterprise users have predefined business workloads whose characteristics impose strict performance requirements. The margin benefit for *HaaS* providers materialize from the economy of scale of building huge data centers infrastructures with gigantic floor space, power, cooling costs as well as operation and management expertise.

HaaS providers have to address a number of technical challenges in operating and managing their services. Efficiency, ease and speed of provisioning such large scale systems, for example is a major challenge. Remote scriptable boot-loaders is one solution to remotely boot and deploy complete software stacks on the data centers. PXE [35] and UBoot [36] are examples of remote bootstrap execution environments that allow the system administrator to stream a binary image to multiple remote machines at boot-time. One example of such systems is IBM Kittyhawk [37], a research project that uses UBoot to script the boot sequence of thousands of remote Bluegene/P nodes over the network. Other examples of challenges that arise at this cloud layer include data center management, scheduling, and power-consumption optimizations.

V. Discussion

In the last section, we outlined our proposed cloud ontology and described the different layers of our classification and the relationships between them. We also identified the users of each cloud layer. In addition, we explicated the challenges and trade-offs encountered by the providers of the cloud services at each level of the stack. Relevant to the discussion is the business model which was the incentive for the inauguration of cloud computing.

Huge coperation, like Amazon and Google generally build their computing infrastructure to support the peak demand of their business. As the average demand of the system is however, several times smaller than the peak demand [18], their computing infrastructure is usually under-utilized and the cost of its operation constitutes an additional expense. In order to offset this extra expense, they have offered utilizing this surplus computing power as a service when it is not used by their business systems. Their offerings came at very attractive prices since they have deployed their systems at a large scale, and thus benefit from the economy-of-scale. Table I exemplify some of the commercial cloud systems discussed earlier in our proposed ontology.

In addition, advances in virtualization in the last few years have introduced novel system techniques that gives the cloud-provider the ability to transparently satisfy its cloud users requests without impacting its own utilization of the system. In this regard, cloud computing is different from grid computing in that the cloud workloads are capable of running alongside the original business workloads. Furthermore, novel virtualization capabilities such as live-migration and pause-resume [38], [39] allow rapid and transparent solutions, should interference occur between the original business and the cloud workloads. On the other side, cloud users benefit from the reduced cost and the rapid system provisioning, in addition to their ability to expand or reduce their computing infrastructure efficiently on-demand.

While grid computing research has investigated several economical models for computational grid infrastructure over the last decade, cloud computing has a more applied approach to business models, empirically evaluationg their effectiveness on the market place.

Pricing models for the different cloud services have taken one of three forms: tiered pricing, per-unit pricing and subscription-based pricing. Tiered pricing is the model adopted by Amazon's cloud systems, where the cloud services are offered in several tiers; each tier offers fixed computing specifications (i.e. memory allocation, CPU type and speed, etc), and SLA at a specific price per unit time. Per-unit pricing is normally applied to data transfers or memory usage. Main-memory allocation, for example is used by GoGrid Cloud offering [40], where they denoted "RAM/hour" as the usage unit for their system. This model is, arguably more flexible than the tiered pricing, as it allows the users to customize the main memory allocation of their system based on their specific applications' needs. Finally, the subscription-based model is the most-widely used pricing model for *SaaS*. This model allows the users to predict their periodic expenses of using the cloud applications. However, it lacks the accuracy of charging the users for what they actually have used.

Among the substantial challenges that hinders the wide adoption of cloud computing are the security and privacy concerns of the data in the cloud. Although the cloud providers have deployed their *DaaS*-systems near the computational infrastructure in order to decrease the latency in transferring the data from its storage to its processing site, many users still fear data leakage. Current security approaches include using Public Key

Cloud Layer	Examples of Commercial Cloud Systems
Cloud Application Layer	Google Apps and Salesforce Customer Relation Management (CRM) system
Cloud Software Environment	Google App Engine and Salesforce Apex System
Cloud Software Infrastructure	*Computational Resources*: Amazon's EC2, Enomalism Elastic Cloud.
	Storage: Amazon's S3, EMC Storage Managed Service.
	Communication: Microsoft Connected Service Framework (CSF).
Software Kernel	Grid and Cluster Computing Systems like Globus and Condor.
Firmware / Hardware	IBM-Morgan Stanley's Computing Sublease, and IBM's Kittyhawk Project.

TABLE I
SUMMARY OF EXAMPLES OF SOME CONTEMPORARY CLOUD COMPUTING SYSTEMS, AND THEIR PROSPECTIVE CLASSIFICATION INTO OUR PROPOSED CLOUD ONTOLOGY LAYERS

Infrastructure (PKI) and X.509 SSL certificates as a methodology for authentication and authorization in the cloud. In addition, data ownership are among the significant obstacles to cloud computing. Due to the absence of cloud computing standards that addresses these issues, cloud security, data privacy and ownership policies are approached differently by each cloud provider.

Monitoring for cloud systems is another active research topic in cloud computing. With the enormous size of the cloud data centers and the large number of nodes supporting any cloud offering, hardware and software failures become an unavoidable reality, for which a robust monitoring system must be in place to allow the cloud services to actively react to failures. For that, two common approaches in designing monitoring and management systems for the cloud are repetitively implemented: tree-based systems or Gossip-based protocols. Spanning-tree approaches have been used in grid and distributed systems [41], [42] for managing, monitoring and co-scheduling jobs to different nodes, as they have the advantage of their fast scalability and hierarchical characteristics. However, they lack the resilience to nodes failures, since failure of non-leaf nodes would require partial re-organization of the tree structure. On the other hand, Gossip-based protocols offer a better fault-tolerance model for monitoring nodes in large systems. However, they present a trade-off between monitoring estimation accuracy and the protocols' overhead. For example, Amazon *S3* [43] and *Dynamo* [18] deploy a gossip-based protocol to quickly spread node state-information throughout the system, in order to avoid scheduling jobs to failed nodes. On the other hand, IBM has maintained Tivoli [44] as its solution for cloud monitoring and management, which utilizes a CORBA-based architecture to manage the large number of nodes in its clouds. Despite its centralized management, Tivoli is capable of collecting and storing sustainable quantity of data in addition to its data visualization capabilities.

Research in monitoring large distributed systems and sensor networks has comprehensibly outlined the trade-offs between tree-based approaches and gossip-based protocols. For example, Wulib et al. in [45] presented a comparison of tree-based and gossip-based aggregation protocols for distributed real-time monitoring, while Babaoglu et al. in [46] presented a case study for a general-purpose P2P framework that enables self-organization and monitoring in large distributed systems. Nevertheless, a challenge in cloud monitoring lies in exporting the system monitoring information to the cloud users, whom might consequently utilize the information to optimize and fine-tune their own cloud applications.

In addition to its commercial use, cloud computing is offering a very attractive deployment infrastructure for HPC and scientific applications. Several researchers have studied the feasibility of deploying their scientific codes on cloud computing systems. In our previous work [47], [48], we have characterized the performance ramifications of deploying high performance computing benchmarks – including NAS parallel benchmarks and Linpack – on clusters of virtual machines similar to those systems deployed in the *IaaS* layer. Furthermore, our recent results [49] have shown that memory-intensive linear algebra applications have not encountered statistically significant performance impact on these systems. Memory-intensive applications could further benefit from customized application-specific operating systems [50], [51], [52] and dynamic adaptation of the virtual machines [53]. Additionally, studies like [54], [55] have evaluated the deployment and performance benefits of porting specific scientific parallel applications to Amazon's EC2 cloud. After some fine-tuning, they reported that their applications did not observe any notable performance ramifications. The main hinder in adopting scientific application in the cloud is, however the high latency in communication between the nodes. since most parallel scientific applications were designed to run on tightly-coupled dedicated nodes, they need to be modified to tolerate the high communication latency in the cloud. Ideally, scientific applications in the cloud should have the capability to dynamically adjust their communication-to-computation ratio in order to transparently endure the high variability in the cloud performance.

VI. Conclusions

In this paper, we proposed a detailed ontology for the cloud in an attempt to establish the knowledge domain of the area of cloud computing and its relevant components. We used composability as our methodology in constructing our cloud ontology which allowed us to capture the inter-relations between the different cloud components. We presented our proposed ontology as a stack of cloud layers, and discussed each layer's strengths, limitations as well as dependence on preceding computing concepts.

Few recent online articles attempt to establish a similar structure for cloud computing and its components [56], [57], [58]. Although they attain some valuable understanding of several cloud services and components, they tend to be more general classifications. They neither went to the level of detail in the analysis as we did, nor they included all the cloud layers we captured in our model. Their main end-goal is to classify the commercial cloud offerings in order to analyze the cloud computing market opportunities. As such, they do not address the specific potentials or limitations of the several cloud layers, nor the research opportunities associated with each cloud layer. We believe our proposed cloud ontology is more comprehensive, and encompass more detailed analysis of the cloud computing knowledge domain.

Acknowledgments: The authors would like to acknowledge Dan Tsafrir and Jonathan Appavoo for the useful discussions on this topic and their inputs and feedback while writing this paper.

References

[1] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, 2004.

[2] M. P. Papazoglou and W.-J. Heuvel, "Service oriented architectures: approaches, technologies and research issues," *The VLDB Journal*, vol. 16, no. 3, pp. 389–415, 2007.

[3] X. Jin and J. Liu, "From individual based modeling to autonomy oriented computation," in *Agents and Computational Autonomy*, ser. Lecture Notes in Computer Science, M. N. et al., Ed., vol. 2969. Springer, 2003, pp. 151–169.

[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI'04*, 2004. [Online]. Available: http://labs.google.com/papers/mapreduce.html

[5] "GOOGLE App Engine," http://code.google.com/appengine.

[6] "Amazon elastic compute cloud," http://aws.amazon.com/ec2/.

[7] "Salesforce Customer Relationships Management (CRM) system," http://www.salesforce.com/.

[8] "GOOGLE Apps," http://www.google.com/apps/business/index.html.

[9] "Apex: Salesforce on-demand programming language and framework," http://developer.force.com/.

[10] "Hadoop," http://hadoop.apache.org/.

[11] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2008, pp. 1099–1110. [Online]. Available: http://dx.doi.org/10.1145/1376616.1376726

[12] Y. Koh, R. C. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *ISPASS*. IEEE Computer Society, 2007, pp. 200–209.

[13] "Enomalism elastic computing infrastructure," http://www.enomaly.com.

[14] "Eucalyptus home page," http://eucalyptus.cs.ucsb.edu/.

[15] "Virtual workspaces science clouds," http://workspace.globus.org/clouds/.

[16] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[17] K. Petersen, M. Spreitzer, D. Terry, and M. Theimer, "Bayou: replicated database services for world-wide applications," in *EW 7: Proceedings of the 7th workshop on ACM SIGOPS European workshop*. New York, NY, USA: ACM, 1996, pp. 275–280.

[18] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *SOSP '07*. New York, NY, USA: ACM, 2007, pp. 205–220.

[19] "Amazon simple storage service," http://aws.amazon.com/s3/.

[20] "EMC Managed Storage Service," http://www.emc.com/.

[21] W. J. et al., "Network Communication as a Service-Oriented Capability," *High Performance Computing and Grids in Action, Advances in Parallel Computing*, vol. 16, March 2008.

[22] A. H. et al., "Perfsonar: A service oriented architecture for multi-domain network monitoring," in *ICSOC*, ser. Lecture Notes in Computer Science, B. B. et al., Ed., vol. 3826. Springer, 2005, pp. 241–254.

[23] J. Hofstader, "Communications as a Service," http://msdn.microsoft.com/en-us/library/bb896003.aspx.

[24] "Microsoft Connected Service Framework," http://www.microsoft.com/serviceproviders/solutions/connectedservicesframework.mspx.

[25] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, 1997.

[26] T. Tannenbaum and M. Litzkow, "The condor distributed processing system," *Dr. Dobbs Journal*, February 1995.

[27] R. W. et al., "Grid resource allocation and control using computational economies," in *Grid Computing: Making the*

Global Infrastructure a Reality. John Wiley & Sons, 2003, pp. 747–772.

[28] T. Severiens, "Physics portals basing on distributed databases," in *IuK*, 2001.

[29] P. Smr and V. Novek, "Ontology acquisition for automatic building of scientific portals," in *SOFSEM 2006: Theory and Practice of Computer Science: 32nd Conference on Current Trends in Theory and Practice of Computer Science*. Springer Verlag, 2006, pp. 493–500. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=8004

[30] M. Chau, Z. Huang, J. Qin, Y. Zhou, and H. Chen, "Building a scientific knowledge web portal: the nanoport experience," *Decis. Support Syst.*, vol. 42, no. 2, pp. 1216–1238, 2006.

[31] M. Christie and S. Marru, "The lead portal: a teragrid gateway and application service architecture: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 19, no. 6, pp. 767–781, 2007.

[32] D. G. et al., "Building Grid Portals for e-Science: A Service Oriented Architecture," *High Performance Computing and Grids in Action, IOS Press, Amsterdam*, 2007.

[33] D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Dukle, L. Fang, M. Farellee, G. Fox, S. Hampton, G. Kandaswamy, D. Kodeboyina, C. Moad, M. Pierce, B. Plale, A. Rossi, Y. Simmhan, A. Sarangi, A. Slominski, S. Shirasauna, and T. Thomas, "Building grid portal applications from a web-service component architecture," *Proceedings of the IEEE (Special issue on Grid Computing)*, vol. 93, no. 3, pp. 551–563, March 2005.

[34] "Morgan Stanley, IBM ink utility computing deal," http://news.cnet.com/2100-7339-5200970.html.

[35] "Preboot Execution Environment (PXE) Specifications, Intel Technical Report, September 1999."

[36] "Das U-Boot: The Universal Boot Loader," http://www.denx.de/wiki/U-Boot/WebHome.

[37] J. Appavoo, V. Uhlig, and A. Waterland, "Project Kittyhawk: building a global-scale computer: Blue Gene/P as a generic computing platform," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 1, pp. 77–84, 2008.

[38] C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, 2005, pp. 273–286.

[39] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the man/wan," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 901–907, October 2006. [Online]. Available: http://dx.doi.org/10.1016/j.future.2006.03.007

[40] "GoGrid," http://www.gogrid.com.

[41] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical in-network data aggregation with quality guarantees," in *In EDBT*, 2004.

[42] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 757–768, 1999. [Online]. Available: citeseer.ist.psu.edu/wolski98network.html

[43] "Amazon S3 Availability," http://status.aws.amazon.com/s3-20080720.html.

[44] "IBM Tivoli Monitoring," http://www.ibm.com/software/tivoli/products/monitor/.

[45] F. Wuhib, M. Dam, R. Stadler, and A. Clemm, "Robust monitoring of network-wide aggregates through gossiping," *10th IFIP/IEEE International Symposium on Integrated Network Management, 2007*, pp. 226–235, 21 2007-Yearly 25 2007.

[46] O. Babaoglu and A. Montresor, "Managing clouds: a case for a fresh look at large unreliable dynamic networks," *SIGOPS Oper. Syst. Rev*, vol. 40, p. 2006.

[47] L. Youseff, R. Wolski, B. Gorda, and C. Krintz, "Evaluating the performance impact of xen on mpi and process execution for hpc systems," in *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, 2006.

[48] ——, "Paravirtualization for HPC Systems," in *ISPA Workshops*, ser. Lecture Notes in Computer Science, G. Min, B. D. Martino, L. T. Yang, M. Guo, and G. Rünger, Eds., vol. 4331. Springer, 2006, pp. 474–486.

[49] L. Youseff, K. Seymour, H. You, J. Dongarra, and R. Wolski, "The impact of paravirtualized memory hierarchy on linear algebra computational kernels and software." in *HPDC*. ACM, 2008, pp. 141–152.

[50] G. Back and D. S. Nikolopoulos, "Application-Specific Customization on Many-Core Platforms: The VT-ASOS Framework," in *Proceedings of the Second Workshop on Software and Tools for Multi-Core Systems*, March 2007.

[51] C. Krintz and R. Wolski, "Using Phase Behavior in Scientific Application to Guide Linux Operating System Customization," in *Workshop on Next Generation Software at IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, April 2005.

[52] L. Youseff, R. Wolski, and C. Krintz, "Linux Kernel Specialization for Scientific Application Performance," Univ. of California, Santa Barbara, Tech. Rep. UCSB Technical Report 2005-29, Nov 2005.

[53] T. Naughton, G. Vallee, and S. Scott, "Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure," in *First Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2007)*, Mar 2007.

[54] C. H. Constantinos Evangelinos, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2," *The First Workshop on Cloud Computing and its Applications (CCA'08)*, October 2008.

[55] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–12.

[56] "Defogging Cloud Computing: A Taxonomy, June 16, 2008," http://refresh.gigaom.com/2008/06/16/defogging-cloud-computing-a-taxonomy/.

[57] "Cloud Services Continuum, July 3, 2008 ," http://et.cairene.net/2008/07/03/cloud-services-continuum/.

[58] "The Cloud Services Stack and Infrastructure, July 28, 2008," http://et.cairene.net/2008/07/28/the-cloud-services-stack-infrastructure/.