# Two's Company, Three's A Cloud: Challenges To Implementing Service Models

Yvette Ghormley, Ph.D., Walden University, USA

**ABSTRACT**

*Although three models are currently being used in cloud computing (Software as a Service, Platform as a Service, and infrastructure as a service, there remain many challenges before most business accept cloud computing as a reality. Virtualization in cloud computing has many advantages but carries a penalty because of state configurations, kernel drivers, and user interface environments. In addition, many non-standard architectures exist to power cloud models that are often incompatible. Another issue is adequately provisioning the resources required for a multi-tier cloud-based application in such a way that on-demand elasticity is present at vastly different scales yet is carried out efficiently. For networks that have large geographical footprints another problem arises from bottlenecks between elements supporting virtual machines and their control. While many solutions have been proposed to alleviate these problems, some of which are already commercial, much remains to be done to see whether these solutions will be practicable at scale up and address business concerns.*

**Keywords:** Cloud Computing; Cloud-Based Applications; Service Science; Virtualization; Resource Provisioning; Dynamic Allocation

## INTRODUCTION

Cloud computing has become of considerable interest in the business community in the last few years due to the potential it offers through its various service models. Cloud computing can be defined as applications delivered over the Internet as services as well as the hardware and systems software in data centers required to provide those services (Armbrust et al., 2010). It is essentially a shared pool of virtual computing resources, including networks, servers, storage, and applications, packaged from the consumer's point of view on a pay-as-you-go basis into one of three models: Software as a Service (SaaS); Platform as a Service (PaaS); and infrastructure as a service (IaaS) (Goodburn & Hill, 2010).

The concept of delivering software applications over a network is certainly not a new concept, and the Internet has facilitated delivery of a wide variety of simple packaged software applications since the 1990s. However, when companies began creating open-integration platforms in which other companies could utilize features to develop their products to be delivered a new paradigm was born (Cusumano, 2010); AppExchange (Saleforce.com), Elastic Compute Cloud (EC2; Amazon), and Google App Engine (Google) are all examples of SaaS platforms.

While model definitions do vary, and some researchers suggest that such definitions are too woolly to justify separate model definitions, NIST (National Institute of Standards and Technology) defines three cloud computing models (Mell & Grance, 2010). The essential feature of cloud SaaS is the ability to provide one or more applications to the consumer that are hosted by the provider on a cloud infrastructure via a suitable client interface and which the consumer has no control over in terms of execution except perhaps for user-specific configuration settings. Cloud IaaS, on the other hand, is the provision to the consumer of basic computing resources, such as networks with associated storage and processing devices, which the customer can utilize to install software,

including operating systems and applications. While the consumer has domain over operating systems, storage, deployed applications, and perhaps some networking components, management or control of the cloud infrastructure is still within the purview of the provider. Lying in the middle of the continuum between these two models PaaS positions itself as the capability to deploy applications developed by the consumer using tools and programming languages supplied by the provider without control over the cloud infrastructure or any of its associated operating mechanisms. The consumer does have control, however over how the applications are deployed and often the environmental configurations under which the applications run.

The real advantage of cloud computing lies in the capability to harness massive computing resources to meet shifting loads on demand without the upfront needs to buy hardware and network-associated systems to accommodate some possible maximum demand. In fact, for many business start-ups and small-to-medium businesses, this is an ideal situation in that not only is capital investment drastically reduced but payment for cloud computing models depends on moment-by-moment utilization, rather than premium or reservation fees for future surges (Glaser, 2011; Oakes, 2011; Payton, 2010). This is possible when economies of scale are married to agile reallocation of cloud resources for maximum utilization (Armbrust et al., 2010). This allows businesses to focus more on core competencies and IT functions on innovation and value to business processes (Giniat, 2011).

Users of cloud computing can be grouped into several categories: private, public, community, and hybrid. A private cloud represents a closed environment in which a single organization utilizes resources hosted by a third party (Goodburn & Hill, 2010). In some circumstances this may mean that the consumer and service host are the same entity, although geographically separated elements of the organization may be present (e.g., data center and central offices) (Yoo, 2011). Academic organizations can also host private clouds, such as the e-Learning application Servlet Container Platform developed by the Hochschule Furtwangen University in Germany (Doelitzscher, Sulistio, Reich, Kuijs, & Wolf, 2011).

Public clouds, by contrast, are by definition potentially accessible by all entities and are hosted by a wide variety of providers. Pricing varies with fees according to application and resource consumption although some applications may be free albeit tied to fee-for-service for "premium" variants. Quality of service also ranges from "as is" (i.e., no guarantees) to specific reliability and accessibility guarantees defined in service level agreements.

Hybrid clouds are an extension of private cloud models via addition of public cloud computing resources to handle flash overloads in a cost-effective and seamless manner—what is often termed cloudbursting (Heckel, 2010). However, hybrid cloud computing is often heavily criticized on cost, interoperability, and uptime grounds. More importantly as Linthicum (2011) has pointed out: "The problems are the lack of architectural forethought around the business requirements to the technology requirements, as well as the absence of a clear understanding of the advantages and limitations of the technology available. Workers involved with these projects get caught up in the hype, feel pressured to get something cloudlike up and running, and hit walls they don't see until it's too late." Thus, while hybrid clouds may be appropriate for some applications, they are not a universal panacea.

Perhaps the most interesting application of cloud computing may lie in community clouds, the architects of which seek to eliminate vendor control, common in public cloud computing, and address lack of environmental sustainability. Thus community clouds are amalgamations of distributed resource provision (grid computing), distributed control (digital ecosystems), and sustainability (green computing), with greater use of self-management advances (autonomic computing) using case models from cloud computing (Briscoe & Marinos, 2009). One of the aspirations for the community cloud computing variant may lie in high performance scientific computing applications currently utilizing grid computing (Schwiegelshohn et al., 2010) in which resources may be limited at any given time and for which there may be a larger number of potential users than the system currently can accommodate.

Much is being promised regarding cloud computing and although there may be as many detractors as enthusiasts, what are the technical obstacles to make cloud computing more of a reality in business? In this review, some of these issues are explored together with potential solutions.

## VIRTUALIZATION

Virtualization has been around the computing world since the advent of mainframes. Originally it was conceived as a way to "partition" a single machine into smaller virtual machines by running multiple operating systems or multiple sessions of the same operating system thus giving the impression to an end user that his or her job was running on a separate, dedicated machine (Yoo, 2011). In the cloud computing environment, this process can be extended through configuration on demand, maintained, and replicated (Marston, Li, Bandyopadhyay, Zhang, & Ghalsasi, 2011), so that if more resources are required, the virtual machine (VM) can be expanded.

There are many advantages to virtualization. For example, some legacy applications are very brittle and require very specific hardware to successfully execute, including many forms of scientific modeling in which old validated codes are used. Virtualization can be employed in these cases to utilize a specific operating system, libraries, software packages, and a directory structure in which the application can be anchored so that it does not interfere with other users of the system (Deelman, 2010). Another example is that created by the Hochschule Furtwangen University in Germany in which a cloud computing environment was employed using virtual machines created by students of e-Learning applications that increased utilization of PC labs and servers (Doelitzscher et al., 2011; Dong, Zheng, Yang, Li, & Qiao, 2009). Each student can create a maximum of 3 VMs using an IaaS web front-end wizard that permits 1 GB RAM for each VM, and 100 h per semester of CPU time (Doelitzscher et al., 2011). After VM parameters are set, the user then chooses which software packages to install. In an IaaS setting, VM parameters would not be set by the user but are preconfigured depending on the application and remain transparent to the user.

However, virtualization comes at a steep price in terms of memory because in addition to specific applications, accessory pieces of state, such as auxiliary libraries, locales, general purpose software, configuration structures that support all-encompassing distribution, a multitude of kernel drivers for extensive hardware, and user interface environments (Lagar-Cavilla et al., 2011). In essence, the massive state predicated by a VM slows its creation and scalability, which is in opposition to the inherent cloud computing concept of being "on demand" or elastic.

## MIGRATING GRIDS INTO CLOUDS

Grid computing originally evolved as a means to provide users who do not have the computing power to solve engineering and scientific problems access to larger resources. Some examples of computing grids developed in the last decade include the German D-Grid (Gentzsch & Reinefeld, 2009), the French Grid'5000 (Bolze et al., 2006), the Dutch DAS (Bal et al., 2000), and the TeraGrid and Open Science Grid in the USA (Pordes et al., 2007; Stewart, Ahalt, & Pritchard, 2008). However, the materialization of computing grids has lagged behind forecasts and so far is limited to a relatively small number of users. Common issues found in grid computing are how to: (1) acquire timely and accurate information about available distributed resources; (2) efficiently scale workflow mapping and execution processes; (3) overcome hardware and middleware limitations; (4) efficiently mesh workflow task scheduling on to batch systems; (5) manage data transfer and storage; and (6) cope with resource and application limitations (Deelman, 2010). Schwiegelshohn et al. (2010) define a computing grid as "a distributed system that supports a virtual research environment across different institutions" (p. 1105). Consequently, computing grid research has looked increasingly toward cloud systems as a means of surmounting these problems.

To understand how computing grids or other systems might be transformed into computing clouds, it is useful to examine a generic structure in terms of the architectural layers and the components involved in an IaaS application (Figure 1). The user defines the model to be deployed either via a template or by programming it through a suitable interface; these functions constitute the *design layer*. The *service orchestrator* coordinates all actions, passing the model parameters to the *infrastructure manager*, which selects the appropriate cloud provider (if more than one is available) and requests the virtual infrastructure be created with appropriate connections registered with the VM connection manager (Kirschnick, Calero, Wilcock, & Edwards, 2010). The *software installation manager* takes the installation image and selects the required component-specific package specifications from the library. The library holds package and configuration descriptions for all the software components that the service orchestrator needs to manage. The defined configuration image is then handled by the *configuration manager* for implementation

of the configuration, which can initially involve configuration of the software prior to running, after running has started, or reconfiguration during the VM life cycle should topological changes be necessary. The function of the *application manager* is then to start the individual software programs in the newly created VM(s). Once deployment has been achieved the service orchestrator uses the monitoring manager to track the VM and its associated software components in real time until the life cycle of the VM is complete.

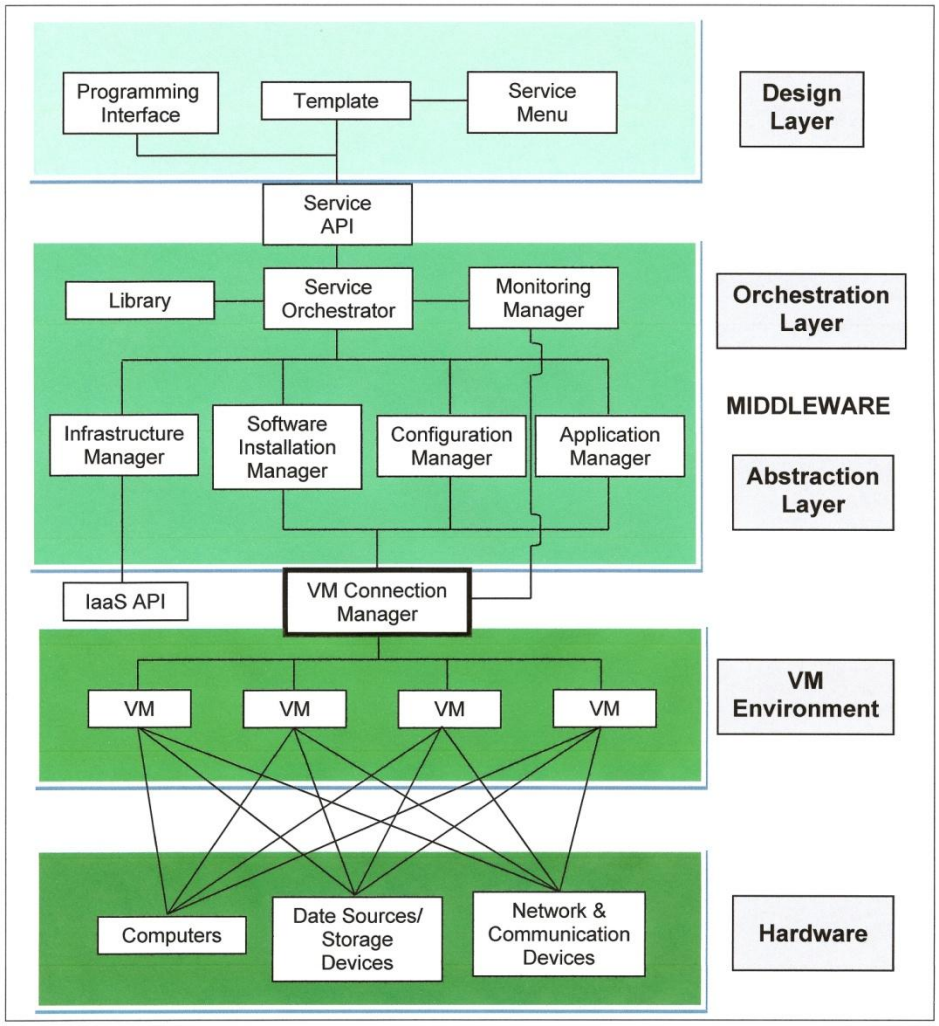


**Figure 1. Generic architecture required for an IaaS application (adapted from Kirschnick et al., 2010)**

Efficiently managing VMs in the cloud environment can be a challenge, and although all aspects cannot be covered here, some of the most common issues are detailed, along with various proposed solutions.

## RESOURCE PROVISIONING (DYNAMIC ALLOCATION)

Besides the performance penalty associated with virtualization, sometimes called overhead, how does one provision the resources required for a multi-tier cloud-based application so that resources can be expanded upon demand, yet minimize the time associated with this "elasticity?"

 

One software-based mechanism rearranges VMs so that the number of allocated physical machines is minimized, but is based on off-line application performance as a function of machine utilization (Khanna, Beaty, Kar, & Kochut, 2006), and thus cannot be used for on-the-fly adjustment. Commercially available software is already available to allot the amount of physical resources required by VMs automatically based on predefined policies and priorities at the VM level using a live-migration automation mechanism (VMware, 2011). Xen, another popular VM manager, uses a layer of software (the hypervisor) to replace the operating system, thus permitting computer hardware to run multiple guest operating systems for a very small performance penalty—typically a few percent (Barham et al., 2003). Currently, Xen powers most public cloud services and many hosting services, such as Amazon Web Services, Rackspace Hosting, and Linode. However, while scalability of disk, memory, and processor capacity may be an existing virtue of existing cloud providers, instantiation latencies of over a minute for Amazon's EC2 cloud were common when spawning VMs according to Lagar-Cavilla et al. (2011). The solution to these long latencies, according to these researchers is to use a cloud-programming paradigm (SnowFlock; a Xen-based implementation) that mimics the semantics of UNIX process forking. Preliminary results suggest that by copying only the critical state and fetching the VM's memory image on efficiently on demand, it is possible to significantly reduce time taken to clone a VM. Moreover, simple modifications to the guest kernel can substantially reduce network traffic by eliminating transfer of pages that will be overwritten. Thus the cost for "forking" a VM can be lowered to only 40 MB for application footprint of up to one GB. The logical outcome of this research points to "just in time" VM creation and disassembly so that "idle" VMs do not exist.

Much research in this area focuses on the infrastructure layer, including resource provisioning and placement of VMs. For example, the 1000 Islands architecture (Zhu et al., 2009) harnesses three different types of controllers: (1) *node controllers*, which operate on the shortest time scale (seconds) and dynamically adjust resource allocations for the VMs running on each server (node); (2) *pod controllers*, which operate on a time scale of minutes and manage pods (workload migration domains consisting of multiple nodes) through adjustment of workload placement on nodes within a pod as pod conditions change; and (3) *pod set controllers*, which operate on time scales of hours to days and analyze past resource consumption history of many workloads, determining whether the data center as whole has sufficient resource capacity to meet workload demands.

Iqbal et al. (2011) utilized a heuristic approach to test a prototype for reactive scale-up of a two-tier Web application and a predictive model for scale-down in which scaling means the amount of resources devoted to job as it progresses so that time response requirements are met and temporary over-provisioned resources are released when no longer needed. Although their experiments were limited in scope, they noted that insufficient static resource allocation polices led to system failure but that maximal static resource allocation policies led to over-provisioning; examples static resources included HTML, JPG, PNG, and TXT pregenerated flat files.

Schemes for managing clusters of VMs within grid settings, termed virtual organization clusters (VOCs) (Murphy & Goasguen, 2010), are also proliferating, as a means of improving the issue of resource underutilization and better bridging the gap between available resources and applications, such as the Open Science Grid (Foster, Freeman, Keahy, Scheftner, Sotomayer, & Zhang, 2006).

To understand how a VOC is created, it is important to first define a cluster, which consists of a set of *nodes*, also known as processing elements (Figure 2). The core of a cluster is composed of *compute nodes*, which are dedicated servers that form the cluster's processing power, and which use a dedicated *interconnect fabric* to communicate data and control messages between the compute nodes (Kant, 2005). The core cluster is supported by *head nodes*, *storage nodes*, *visualization nodes*, and *management nodes*. Head nodes—typically dedicated servers—constitute the communications conduit of the cluster to the outside world and allow the user to both solve and check the status of submitted problems. While storage nodes are optional, if an application requires significant memory, then a dedicated server is used in this capacity. For applications that have visualization components or when a user might want to inspect ongoing computations or analyze postprocessed data, a workstation is employed. Finally, clusters generally need a management node—usually a workstation with a serial multiplexer device—to control other nodes within the cluster, including such modes as monitoring software and hardware, and access to the system console.

A VOC is equivalent to a set of compute nodes in which scheduling of jobs can be accommodated by a shared local scheduler on the same physical site as the VOC, by a dedicated virtual head node also at the same physical site as the VOC, or via a central scheduler using an overlay network. In their modeling of VOCs Murphy and Goasguen (2010) found that although automatic self-provisioning responded nicely to changing workload conditions, VM boot times of 1 minute were common. Based on studies of virtual workspace cluster testbeds using the Chiba City cluster at the Argonne National Laboratory, Zhang, Keahey, Foster, and Freeman (2005) also observed that staging times for different sizes of requested virtual workspace clusters were dominated by transfer times with a time-dependent penalty with increasing cluster size, although the increase in time was minimal with clusters >8. More advanced systems address the need for creating custom computational clouds by virtual organizations of grid users. For example, a model has been developed that creates a virtual cluster environment, which is homogeneous across grid sites and automatically self-provisioned, meaning that resources are acquired on an as-needed basis. The model can be implemented in stages and transparency from the user's point of view achieved by a design that obviates the requirement for submission endpoint middleware (Murphy & Goasguen, 2010).

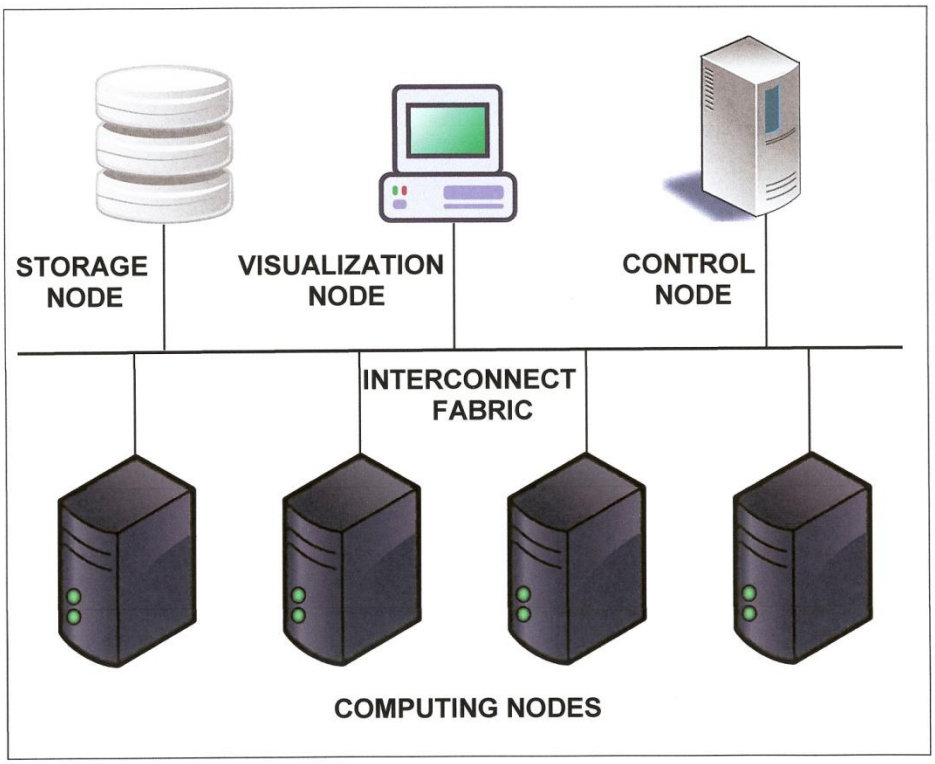


**Figure 2. Typical cluster architecture (adapted from Kant [2005]).**

Interoperability—the ability to add a given resource to a cloud or transformed grid regardless of interfaces in play—is another facet of dynamic provisioning that needs to be addressed, and appears to be partly the result of a failure to adopt accepted standards. Vásquez, Huedo, Montero, and Llorente (2011) have proposed a metascheduler architecture in which a portal is created that utilizes different components to submit jobs, gather information, and transfer files, and which interfaces between sites. These components are in essence adaptors designed to operate with specific middleware stacks or even versions. Thus, when the service manager detects an overload of the system through heuristic algorithms as it monitors the meta-scheduler, it can grow the current grid infrastructure by using specific adaptors to access other cloud providers. Either single hosts can be recruited or a full virtualized cluster can be deployed.

Application level interoperability is also of importance and researchers have made the case that an application ought to be able to utilize any programming model and that any programming model ought to be executable on any underlying infrastructure (Sehgal, Erdelyi, Merzky, & Jha, 2011). To prove their point, these authors successfully developed an interoperable implementation of MapReduce (developed by Google), using SAGA—an API to support distributed programming—and a canonical word count application that scaled out across clouds, clusters, and high-performance computing resources. However, other researchers suggest that MapReduce is not the optimal paradigm, because the overall system must be aware of which task's output is required as another task's input (Warneke & Kao, 2011). In a VM environment, if this workflow is not properly mapped, the result will be idle VMs because they cannot be deallocated. Warneke and Kao (2011) suggests that new processing frameworks, such as Nephele, have an advantage in terms of cost reduction and resource utilization, because users have to employ a job manager before VM instantiation. The job manager receives client's jobs, schedules them, coordinates their execution, and communicates with the interface that the user employs to start VMS, also know as the cloud controller.

Response time is often tied to bottlenecks regardless of the geographical footprint of the network, and is a common problem in a data center network that might support a cloud service. Greenberg et al. (2011) argue that conventional architectures rely on treelike configurations in which capacity between different levels of the tree is oversubscribed by factors of 1:5 or more due to the high cost of equipment. Paths near the root are oversubscribed even more by an order of magnitude. This situation emerges because traffic flooding from one service affects all others sharing the same network subtree. Furthermore, the typical routing design to accommodate scale forces servers to be assigned topologically significant IP addresses and dividing servers among the VLANs (virtual local area networks), which causes a major problem when servers need to be reassigned to other services as is common in a virtual environment. Their answer is to connect all the servers for a particular service via a noninterfering Ethernet switch, which they term a *virtual layer 2*, regardless of how many servers are dedicated to a given service at any time. Thus, traffic flow is limited only by hardware limitations—i.e., available capacity of network-interface cards of the sending/receiving servers—and one service never interferes with another, with the servers "belonging" to each service experiencing the network as an Ethernet LAN.

Finally, the "vagueness" of resource availability requirements and uncertain load workload in the context of the scheduling strategy of a virtual data center can be tackled using a fuzzy logic system. In this model, an interval type II fuzzy logic system (FLM) (Liang & Mendel, 2000) is harnessed using the Multi-classes Single Queue to Multiple Servers with Local Queues Model for task scheduling in virtual data center (Kong, Lin, Jiang, Ya, & Chu, 2011). While the FLM performed well, the authors also noted that as more VMs were consolidated into each server (known as the consolidation ratio), availability, and responsiveness improved up to a ratio of 4, after which no substantial increase in these parameters was observed.

Finally, another pressing issue is the ability to monitor so that clouds can function efficiently (Meng, Liu, & Wang, 2011). These include (1) events generated by applications, operating systems, servers, and network devices; the limited ability of servers to monitor due to lack of resources; and failures of servers and networks links, which leads to inconsistent monitoring results; (2) the need to process large amounts of monitoring data in a distributed manner while summarizing the information appropriately and avoid running similar tasks in isolation, which causes unnecessary resource consumption. Several solutions have been proposed, including window-based state monitoring (WISE) (Meng et al., 2011), SNPdisk (Yu, Weng, Li, & Luo, 2011), and Lattice (Clayman et al., 2010).

## CONCLUSIONS

Cloud computing is a reality. However, many businesses have not yet made a commitment for some of the technical reasons discussed in this paper, as well as nontechnical reasons, including issues of cost, service-level agreements, and security. On the technical side, there remains much to be done to make cloud more computing efficient in terms of resource utilization, time to create VMs, ensuring that VMS are disassembled when no longer needed, and perfecting the architecture and programming required to make heterogeneous elements work cooperatively. While many potential solutions exist, it is hoped that standardization and customer demands will eventually drive the process so that better models will attract more and different types of clients for private, public, and hybrid cloud computing.

## AUTHOR INFORMATION

**Dr. Yvette Ghormley** holds several degrees to include a PhD in Organization and Management with a concentration in E-Commerce, MA in Education and Human Development with a concentration in Education, Technology and Leadership and a MA in Information Technology, Information Security. In addition, she holds graduate certificates in Acquisition and Contract Management and Project Management. Dr. Ghormley has presented at numerous conferences and has published in International journals, texts and handbooks. Dr. Ghormley is well known for using technological initiatives in business and entrepreneurship for non-profit and for-profit enterprises. E-mail: yvette.ghormley@cox.net.

## REFERENCES

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., et al. (2010). *Communications of the ACM*, *53*(4), 50-58.
2. Bal, H., Bhoedjang, R., Hofman, R., Jacobs, C., Kielmann, T., Maassen, J., et al. (2000). The distributed ASCI supercomputer project. *The International Journal of Supercomputing Applications and High Performance Computing*, *34*(4), 76-96.
3. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., et al. (2003). Xen and the art of virtualization. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (pp. 165-177). New York, NY: Association for Computing Machinery.
4. Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., et al. (2006). Grid'5000: A large scale and highly configurable experimental grid testbed. *International Journal of High Performance Computing Applications*, *20*(4), 481-494.
5. Briscoe, G., & Marinos, M. (2009). Community cloud computing. Paper presented at the First International Conference on Cloud Computing, December 1-4, 2009, Beijing, China. Retrieved October 31, 2011, from http://eprints.lse.ac.uk/26516/1/community_cloud_computing_(LSERO_version).pdf
6. Clayman, S., Galis, A., Chapman, C., Toffetti, G., Rodero-Merino, L., Vaquero, L. M., et al. (2010). Monitoring Service Clouds in the Future Internet. In G. Tselentis et al. (Eds.), *Towards the Future Internet* (pp. 115-125).Amsterdam: IOS Press.
7. Cusamano, M. Technology strategy and management cloud computing and SaaS as new computing platforms (2010). *Communications of the ACM*, *53*(4), 27-29.
8. Deelman, E. (2010). Grids and clouds: making workflow applications work in heterogeneous distributed environments. *The International Journal of High Performance Computing Applications*, *24*(3), 284-298.
9. Doelitzscher, F., Sulistio, A., Reich, C., Kuijs, H., & Wolf, D. (2011). Private cloud for collaboration and e-Learning services: from IaaS to SaaS. *Computing*, *91*, 23-42.
10. Dong, B., Zheng, Q., Yang, J., Li, H., & Qiao, M. (2009). *An E-learning system based on cloud computing infrastructure*. Paper presented at the Ninth IEEE International Conference on Advanced Learning Technologies, ICALT, 2009, 125-137.
11. Foster, I., Freeman, T., Keahey, K., Scheftner, D., Sotomayer, B., & Zhang, X. (2006). Virtual clusters for grid communities. In the Sixth IEEE International Symposium on Cluster Computing and the Grid (pp. 513-520). Washington, DC: IEEE Computer Society.
12. Gentzsch, W., & Reinefeld, A. (2009). Special section D-grid. *Future Generation Computer Systems*, *25*(3), 266-267.
13. Giniat, E. J. (May, 2011). Cloud computing: innovating the business of health care. *Healthcare Financial Management*, 130-131.
14. Glaser, J. (August, 2011). Cloud computing can simplify HIT infrastructure management. *Healthcare Financial Management*, 52-55.
15. Goodburn, M. A., & Hill, S. (2010, December). The cloud transforms business. *Financial Executive*, 35-39.
16. Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., et al. (2011). VL2: A scalable and flexible data center network. *Communications of the ACM*, *54*(3), 95-104.
17. Heckel, P. C. (2010). Hybrid clouds: Comparing cloud toolkits. Retrieved October 31, 2011, from http://www.philippheckel.com/files/hybrid-cloud-paper.pdf
18. Iqbal, W., Dailey, M. N., Carrera, D., & Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications. *Future Generation Computer Systems*, *27*(6), 871-879.

19. Kant, C. (2005). Introduction to clusters. Retrieved November 9, 2011, from http://linuxclusters.com/books/clusters/chapter1/index.html
20. Khanna, G., Beaty, K., Kar, G., & Kochut, A. Application performance management in virtualized server environments (2006). In *Network Operations and Management* Symposium '06 (pp. 373-381). Washington, DC: IEEE Computer Society.
21. Kirschnick, J., Calero, J. M., Wilcock, L., & Edwards, N. (2010). Toward an architecture for the automated provisioning of cloud services. *IEEE Communications Magazine*, *48*(12), 124-131.
22. Kong, X., Lin, C., Jiang, Y., Yan, W., & Chu, X. (2011). Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. *Journal of Network and Computer Applications*, *34*(4), 1068-1077.
23. Lagar-Cavilla H. A., Whitney JA, Bryant, R., Patchin, P., Brudno, M., de Lara, E., et al. (2011). SnowFlock: Virtual machine cloning as a first-class cloud primitive. *ACM Trans Computer Systems*, *29*(1), 1-45.
24. Liang, Q., & Mendel, J. (2000). Interval type-2 fuzzy logic systems: theory and design. *IEEE Transactions on Fuzzy Systems*, 8(5), 535-550.
25. Linthicum, D. (2011). The hybrid cloud is not a silver bullet. *Infoworld*. Retrieved October 31, 2011, from http://www.infoworld.com/d/cloud-computing/the-hybrid-cloud-not-silver-bullet-175327?source=footer
26. Marston, M., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision Support Systems*, *51*(1), 176-189.
27. Mell, P., & Grance, T. (2010). The NIST definition of cloud computing. *Communications of the ACM*, *53*(6), 50.
28. Meng, S., Liu, L., & Wang, T. (2011). State monitoring in cloud datacenters. *IEEE Transactions on Knowledge and Data Engineering*, *23*(9), 1328-1344.
29. Murphy, M. A., & Goasguen, S. (2010). Virtual organization clusters: self-provisioned clouds on the grid. *Future Generation Computer Systems*, *26*(8), 1271-1281.
30. Oakes, G. (Summer, 2011). The cloud: hit or mist? *Manager*, 32-33.
31. Payton, S. (November, 2010). Fluffy logic. *Financial Management*, 22-25.
32. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., et al. (2007). The Open Science Grid status and architecture. Retrieved November 4, 2011, from http://lss.fnal.gov/archive/2007/conf/fermilab-conf-07-451-cd.pdf
33. Schwiegelshohn, U., Badia, R., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., et al. (2010). Perspectives on grid computing. *Future Generation Computer Systems*, *26*(8), 1104-1115.
34. Sehgal, S., Erdelyi, M., Merzky, A., & Jha, S. (2011). Understanding application-level interoperability: Scaling out MapReduce over high-performance grids and clouds. *Future Generation Computer Systems*, *27*(5), 590-599.
35. Stewart, C. A., Ahalt, S. C., & Pritchard, R. H. (2008). The future of the TeraGrid and the national advanced cyberinfrastructure: a perspective from the Coalition for Academic Scientific Computation. Retrieved November 4, 2011, from http://casc.org/papers/CASC_TeraGrid_Futures_White_Paper.pdf
36. Vásquez, C., Huedo, E., Montero, R. S., & Llorente, I. M. (2011). On the use of clouds for grid provisioning. *Future Generation Computer Systems*, *27*(5), 600-605.
37. VMware vSphere™. Distributed Resource Scheduler (DRS)—Dynamic Resource Balancing. Retrieved November 8, 2011, from http://www.vmware.com/products/drs/overview.html
38. Warneke, D., & Kao, O. (2011). Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, *22*(6), 985-997.
39. Yoo, C (2011). Cloud computing: Architectural and policy implications. *Review of Industry Organization*, *38*(4), 405-421.
40. Yu, L., Weng, C., Li, M., & Luo, Y (2011). SNPdisk: An efficient para-virtualization snapshot mechanisms for virtual disks in private clouds. *IEEE Network*, *25*(4), 20-26.
41. Zhang, X., Keahey, K., Foster, I., & Freeman, T. (2005). Virtual cluster workspaces for grid applications. ANL Technical Report ANL/MCS-P1246-0405.Retrieved November 9, 2011, from http://www.nimbusproject.org/files/VWCluster_TR_ANL_MCS-P1246-0405.pdf
42. Zhu, X., Young, D., Watson, B. J., Wang, Z., Rolia, J., Singhal, S., et al. (2009). 1000 islands: an integrated approach to resource management for virtualized data centers. *Cluster Computing*, 12(1), 45-47.

**NOTES**

 