Enabling a Marketplace of Clouds: VMware's vCloud Director

Orran Krieger
VMware
Cambridge, MA
okrieger@vmware.com

Phil McGachey
VMware
Cambridge, MA
pmcgachey@vmware.com

Arkady Kanevsky
VMware
Cambridge, MA
arkady@vmware.com

The VCD Team*
VMware

ABSTRACT

Cloud computing promises to bring about a fundamental shift in the computer industry where consumers of IT enjoy on-demand access to massive compute capacity and producers of IT benefit from economies of scale and automation. We believe that the advantages of cloud computing will be best realized if there is a highly competitive marketplace. We describe our vision of a marketplace of clouds, discuss what is needed to make this vision a reality, and then describe what VMware is doing to help enable this marketplace model of cloud computing.

1. INTRODUCTION

Cloud computing has the potential to bring about a paradigm shift in the way that we think about computing infrastructure. Consumers of IT will no longer have to purchase and maintain private data-centers, with the associated administrative and physical overheads that they entail. Rather, computing will be provided as a utility, with dedicated providers benefitting from large economies of scale, and consumers enjoying on-demand access to massive compute capacity.

Cloud computing means very different things to different people. There are three major categories that differ in the level at which services are provided:

Software as a Service (*SaaS*). Customers of an SaaS solution pay to use an application that is hosted on a remote provider. The service provider manages the software and underlying infrastructure for all tenants, who can simply pay for their usage rather than purchasing the software outright. SaaS covers an enormous and growing set of applications, including connectivity tools (such as Yahoo or Google mail, social networking sites like Facebook, and sites for matching buyers and sellers like Amazon, Ebay and Craigslist), productivity tools (such as Google Docs and Microsoft Office Web Apps) and special services tools like CRM applications, online banking, or travel reservation.

Platform as a Service (*PaaS*). PaaS clouds offer a common environment for application developers to deploy their code. The provider supplies a scalable managed execution environment while the developer writes the software that exploits it. Examples of PaaS solutions include Salesforce's Force.com, Google's App Engine, and parts of the Microsoft Azure and Amazon AWS services.

Infrastructure as a Service (*IaaS*). In an IaaS cloud, the provider sells access to computers upon which the customer can run any software or platform. These resources, which are frequently virtual, are expressed in terms of processing power, memory, storage and network access much as a physical computer would be, but their purchase and maintenance are performed centrally by the service provider. The dominant provider of IaaS today is Amazon [1], although Google, IBM, Microsoft and others have services in this area.

In this paper, we will focus on IaaS. We believe that this space has great potential to revolutionize the way in which compute resources are provisioned and consumed; IaaS allows the consumer to avoid much of the massive expense in maintaining a data center, and allows for rapid expansion to account for peak usage. It also benefits the producer, who can run a large economical data center that caters to many clients. Finally, unlike either SaaS or PaaS, an IaaS solution has the flexibility to support the full gamut of applications, from productivity tools and web services to resource-intensive HPC or enterprise workloads.

IaaS cloud computing has the potential to bring enterprise-class performance and features to a wide range of customers to whom they were previously unavailable. Small businesses will have access to geographic distribution and disaster recovery in a way that was previously practical only to large companies. Ready, self-service access to on-demand bursts of computing capacity will enable researchers to perform high-performance computation when their work requires it, bringing such facilities within the reach of even small projects. And organizations of all sizes can dabble with cloud computing before committing fully, scaling their workloads in the cloud as their confidence grows.

We focus our efforts on *general purpose* cloud computing — making IaaS a feasible prospect for all possible workloads, including academia, small and medium businesses (SMB), scientific workloads and academic research. Historically, IaaS providers have focused only on a subset of this range. For example, Amazon's cloud offerings have emerged from their experience in providing highly-scalable, elastic web services. This has led to *special purpose* cloud solutions that are designed mainly with the web application domain in mind, and with other use cases added as an afterthought. For cloud computing to be a viable option within the enterprise or High Performance Computing (HPC) spaces, the cloud infrastructure must be designed from the bottom up with the performance, connectivity, and reliability that such customers demand.

An additional issue with the current IaaS market is that it has

*See Acknowledgements section for details.

formed an oligopoly in which a small number of providers dominate. These vendors offer tightly-integrated vertical stacks with rich, highly-differentiated interfaces. Consumers of these cloud services are locked into a particular vendor's product and, as a consequence, cannot switch providers when their needs change. While these vertically-integrated clouds have proved successful in the early days of cloud computing, we believe that a wider marketplace of cloud providers will ultimately lead to a richer ecosystem, with greater consumer choice and opportunity for many providers to innovate.

We envisage a standardized set of interfaces and abstractions supported by IaaS providers that will allow applications that interact programatically with clouds to work on multiple clouds, and enable computers hosted on a cloud to be moved from one IaaS provider to another. Enabling cross-cloud compatibility and mobility opens up new options that are not currently available — consumers can choose the ideal provider based on the services that are offered, and will have the ability to migrate between clouds to address changes in their requirements or in the providers' prices or services.

A common cloud interface will also benefit providers, who can tailor their products to the market: a small provider whose scale prohibits competition on price alone can offer a higher service guarantee or additional features such as high availability or regulatory compliance. Independent software vendors (ISVs) can build tools that work with the standard cloud model, encouraging research and innovation that is not hampered by a single provider's business model, as well as driving costs down and supporting competition. Finally, a broad landscape with many implementations of a common standard will avoid the dangers of a mono-culture, where a security bug or misconfiguration can affect the world's IT infrastructure.

In this paper we argue that a highly-competitive marketplace for clouds is possible, is preferable, and has major advantages for both producers and consumers. We first motivate the reasoning behind these assertions, and discuss what is required to make such a marketplace a reality. We then describe the fundamental abstractions and the open-source API that we have proposed as a common interface for cloud computing. Finally, we introduce the recently-released VMware vCloud Director (VCD) product that allows producers to easily deploy general purpose clouds adhering to this interface.

VCD builds upon VMware's vSphere virtualization stack, leveraging its powerful capabilities in configuring, managing and running virtual machines. It expands on vSphere's functionality to apply to the cloud space in several ways. It implements our proposed cloud entity model and API allowing standardized access to cloud resources. It permits multi-tenancy for providers, allowing many customers to be hosted within a single cloud with strong isolation between them. It add self-service access to cloud resources, enabling individual users within customer organizations to provision and manage virtual machines. And, finally, VCD can work with many instances of the vSphere platform, making it possible for service providers to scale beyond the capacity offered by a single vCenter server.

2. THE IAAS MARKETPLACE

We believe that a wide marketplace of compatible clouds would offer significant advantages over the current market, where a small number of vertically-integrated clouds dominate. In this section we will discuss why this approach would benefit both providers and consumers, and detail some of the challenges associated with moving to this model.

2.1 Why a Marketplace?

Cloud computing will thrive if many providers, including internal corporate providers such as IT departments, can instantiate compatible clouds with diverse implementations. There are a number of basic motivations for such a marketplace:

Avoiding vendor lock-in: Customers will embrace cloud computing only if they are not tied to a particular service provider and can switch between vendors as their requirements, or a provider's offerings, change. Today's cloud vendors provide highly differentiated products, making the prospect of transitioning any non-trivial workload between clouds a daunting one, with large costs incurred when migrating data and reworking applications to use a new API.

Enabling internal clouds: Not all workloads are immediately applicable to the cloud computing model; systems may access data that must remain internal for business or regulatory reasons, some processes may take advantage of custom hardware or internal resources, and so forth. However there is great value in presenting a single interface to a company's internal computing infrastructure and their externally-contracted cloud resources. A standard cloud implementation means that an organization can use those abstractions within a secure, internal cloud, and migrate workloads to the external cloud when appropriate. Indeed, in the case of very large enterprises, a cloud maintained by the company's internal IT department may obviate the need for an external provider entirely.

Empowering small producers: A marketplace of clouds will enable many small service providers to participate. This is important to target smaller customers with specialized needs who are best addressed in a retail rather than a wholesale fashion. A multitude of small providers will, in aggregate, provide a cloud with a greater geographical distribution than even the largest single provider could support. This will address local markets that require low-latency access to interacting customers or devices. We have seen this model succeed already: Akamai, for example, does not build its own data centers to serve each territory. Rather, it hosts its content distribution system by purchasing computation capacity in ISPs around the world [21].

Avoiding a mono-culture: Housing the world's IT resources on a small number of providers increases the risk that a single product bug, a software misconfiguration, or a security breach will have a huge impact on the operations of many customers [24, 14, 32]. A competitive landscape of independent implementations greatly reduces this danger, both by containing the exposure of such an incident to the customers of a single provider, and by encouraging more widespread development of security measures.

Enabling infrastructure innovation: One of the strongest reasons for a marketplace model is that it will harness the innovation of a large community of computer vendors selling to many different service providers. Today's vertical clouds are difficult for infrastructure vendors to target with differentiated products, obscuring the value of their products and limiting innovation to a small number of cloud providers.

Enabling application and platform innovation: IaaS clouds have great potential to serve as the underlying infrastructure for

scalable PaaS and SaaS products. Having a standard interface across all clouds will allow these services to be written once and then deployed on multiple clouds. In today's highly-differentiated market, such solutions must be rewritten to the unique interface of each cloud offering.

In addition, current IaaS cloud providers compete with the very application developers that could be enriching their platforms. For example, Amazon provides a message bus that competes with Platform Computing and a database service that competes with Oracle's Platform for SaaS [22]. The provider has huge advantages in this competition, stifling innovation. Adding a standard interface common to all clouds may shift the advantage to application developers whose products could be available on all clouds, opening the market to a much larger community of software developers.

One critical source of innovation is the academic community who, until now, have faced major challenges when researching the design and implementation of IaaS clouds. The current model of large clouds with sophisticated interfaces is not conducive to academic research. A marketplace model with simple interfaces will enable researchers to instantiate their own clouds and investigate tough problems that are immediately relevant to this fast-changing field. We can easily imagine a scenario where researchers can innovate on internal clouds, while evaluating the impact of their work on real and even production workloads.

2.2 Challenges in Creating a Marketplace

While we believe that the marketplace model is essential if cloud computing is to have a fundamental impact, there are enormous challenges in this approach. Developing software that can be consumed by many different providers is much more difficult than the task faced by large cloud vendors today who develop and deploy their solutions in-house. Software developed for multiple providers must:

- Scale up to massive deployments, but also scale down to be manageable for small providers or research groups with only a few machines. Today's cloud solutions are designed only to scale up; developing an environment that can handle both extremes poses additional challenges.
- Have a simple installation model that allows thousands of providers to be able to set up the software. There are very few examples in the industry of shrink wrapped distributed systems, and we have found providing a simple and manageable installation experience to be a major part of the effort in developing our cloud offering.
- Support a broad hardware compatibility list (HCL) that will enable a diverse community of hardware partners to sell to internal and external providers. For example, our IaaS software must support both high performance storage accessed over a local SAN, and lower performance NFS storage that can be accessed over more scalable networks.

It is impossible for software that is delivered to many different vendors to have the same innate agility as that which is developed, deployed and refined within a single organization. To compensate, the software must be sufficiently extensible to allow partners, application vendors and service providers to build upon the platform.

Within a broad ecosystem of providers, we expect different providers to sell their capacity in very different ways. For example, some may want to charge per-VM, while others may provide pools of capacity, and others direct consumption of resources. This means that we must enable providers to define their own offerings and pricing structure within the framework that we provide. Since we do not aim to be prescriptive on either the HCL or on the offerings it becomes very difficult to automate the provision of resources within the software. This is a major challenge since the efficiency of a large cloud depends on end-user self-service and massive automation from the cloud provider.

Perhaps the greatest challenge in creating a marketplace of clouds is that they must be compatible for a consumer to be able to move from one provider to another. This means that the clouds must have the same basic abstractions and interfaces. Enabling an interface that can be standardized across many different providers with different types of offerings and with software implemented by different vendors is a huge challenge.

3. THE VCLOUD MODEL

In this section we discuss the requirements, abstractions and interfaces that we propose as a foundation for the marketplace of clouds. We will begin by outlining the fundamental abstractions that we believe are needed by an IaaS offering to support the enterprise computing market. We then describe in more detail the specifics of the entities that make up our vCloud model. Finally, we discuss the open-source API that we have proposed for standardization and that will enable programmatic access to the cloud.

3.1 Fundamental Abstractions

We have found the following five characteristics to be fundamental for any general-purpose IaaS offering that supports enterprise, SMB, academic and HPC workloads, as well as the web applications traditionally supported by cloud vendors:

- **Virtual machines (VMs)** with persistent storage and connection to networks.
- **Layer 2 (Ethernet) networks** connecting multiple virtual machines that can be bridged or routed to other networks.
- A container that defines **locality and resource mapping** for a group of VMs. We refer to this abstraction as a *vCloud Data-Center (vDC)*.
- A model of an **individual tenant** that contains multiple users and role-based access control. We call this abstraction an *Organization*.
- A vendor-independent way to author, transport, and easily control **complex multi-VM applications**. We refer to these as *vApps* when instantiated, and *vApp Templates* when uninstantiated.

The first three abstractions represent direct analogies to the physical world; compute provisioning is expressed in computers with attached disks, ethernet networks and data centers. We therefore have strong evidence from the physical world that all applications can be supporting on top of those three primitives.

The fourth abstraction is necessary for cloud providers in order to maintain a business that supports multiple customers, but is also vital if organizations are to manage their own users and policies. To support enterprise users the cloud provider must cede some degree of control over user management and access control within the resources allocated to that client. Existing clouds that support only a single user per tenant make it difficult to, for example, revoke the privileges of users that have left the organization.

The final abstraction (a logical grouping of complex applications) is arguably non-essential for standing up a single cloud, since

it has no direct analogy in the physical world. However, grouping logically-related VMs together affords us powerful functionality when managing and operating over large workloads. For example we can deploy all the virtual machines that form an application together, and can manage and reason about them as a unit. More importantly, this abstraction is critical to enable a marketplace of clouds, where we be able to move applications from one cloud to another. In this context there must be a means to serialize complicated and interconnected applications, maintaining the relationships between component parts.

3.1.1 General-Purpose Cloud Computing

While we have argued that our first three primitives are sufficient to support many applications, other IaaS clouds differ in their primitives, so it seems clear that they are not necessary. However, we believe that these primitives are necessary for a general-purpose cloud; i.e., a cloud that efficiently supports enterprise, SMB and HPC use cases.

Machines with persistent state are necessary for enterprise applications that use dedicated machines to perform specific roles. While stateless web applications can tolerate ephemeral storage that disappears when a VM is powered off, most applications have been designed with the assumption that storage is persistent and reliable.

Layer 2 networking is required to support broadcast or other Layer 3 protocols. This is essential for many enterprise workloads. Moreover, as HPC workloads become more important to clouds, we expect that capabilities like RDMA, that depend on Layer 2 connectivity, will be critical.

The container construct allows a provider and consumer to agree on a contract for the connectivity between machines. While such a contract is of less importance for web applications (which are primarily external facing) it is critical for many enterprise and HPC applications to have some guaranteed bandwidth and latency between frequently communicating machines. In fact, Amazon have recently introduced a clustering concept to support HPC users in EC2.

A more subtle requirement for these abstractions is that they enable overprovisioning, both at the provider and the consumer level. Overprovisioning is fundamental to supporting enterprise applications efficiently. Existing cloud providers such as Amazon offer elasticity at the machine level, allowing the user to add or remove EC2 instances from their application. This works well for stateless web applications where an additional server can be easily integrated into an existing application. However, it is less appropriate for complex enterprise applications that may change in resource use over time, but cannot easily adjust to new machines being added or removed

The model of fully allocating physical machines without overprovisioning is essentially the same model that enterprises faced prior to virtualization. Traditional enterprise data centers when deploying a new application would provision a computer large enough to meet the performance demands under the worst case load. Of course, it is very rare that an application actually runs at peak load; the average utilization in an enterprise data center used to be on the order of 10%. With virtualization and the ability to overprovision resources, enterprises have moved to 80% or even 90% average utilization while still meeting individual applications' service level agreements.

To support overprovisioning in the cloud, the notion of a container (such as a vDC) is critical. A container allows the service provider to sell the customer a fixed capacity and then the consumer can overprovision VMs in that capacity. Without a container, the service provider could overprovision resources, but could not overprovision aggressively. Only the user has an understanding of the relative priority of different work, and so only the user can specify what resources must be reserved for particular applications in order for them to meet their service level requirements.

To overprovision the resources of a data center, we must be able to move VMs from one physical machine to another as load on the different machines change [30]. Layer 2 networks are critical for this, allowing us to move VMs between hosts without updating individual routing tables. Persistent storage is also a natural outcome of this. Other clouds introduced ephemeral storage so that they can exploit locally attached disks. Since storage must be shared for overprovisioning and VM mobility, it makes sense to make all storage persistent in our model.

3.1.2 Scalability and Elasticity

As we will see in Section 4, we have built a highly-scalable IaaS cloud infrastructure using these basic abstractions; a single cloud installation can support many thousands of vDCs, VMs, and concurrent users. However, by introducing abstractions that better support enterprise and HPC workloads, we encounter different scalability challenges than those faced by clouds that focus on web applications. In particular, the notion of a vDC imposes scalability limitations that must be addressed.

Our vDC concept allows related VMs to be collocated with the appropriate resources in order to exploit fast local networking and shared storage. While this clustering is essential for high performance or closely-connected applications, it introduces a containment boundary where the scalability of a vDC is limited by the network and network-accessible storage capacity. This results in a bin-packing problem, where the contents of an individual vDC is limited to those VMs that can be supported by the local resources. Note that the cloud installation as a whole can still scale by adding vDCs, but this issue does eliminate the possibility of a single massive and flat vDC.

We believe that this scalability limitation will decrease and eventually disappear with the innovations that are currently emerging from the storage and networking industries. For example, companies such as LeftHand Network (recently acquired by HP) provide scale-out NAS solutions that are competitive in price to locally-attached storage. Similarly, there are huge ongoing innovations in L2 network implementations; technologies such as TRILL [27] and ECMP [25] will allow thousands of hosts (with hundreds of thousands of VMs) to co-exist within a flat L2 network bin without the need for expensive aggregation layers.

3.1.3 Simplicity

Finally, a model for general purpose cloud computing must be sufficiently simple as to allow multiple implementations. This way, different providers can specialize in their underlying infrastructure, taking advantage of different hardware and software stacks to produce very different implementations of the standard. It will allow producers to compete on the range and quality of their services, while allowing consumers to move between clouds based on their needs.

To this end, the abstractions that we present are significantly simpler than those offered by other IaaS vendors. All we provide is a model for virtual hardware, while existing external IaaS clouds provide much richer services such as blob storage, content distribution, messaging or databases. Ultimately, however, all such services are implemented on top of physical hardware. Since we have strong evidence that virtual hardware can be as efficient as physical hardware, we see no reason why these other services can't be built on

top of the basic virtual hardware services we offer.

3.2 Abstractions

Figure 1 presents the entity model that we have designed to fulfill these requirements. It shows the components and relationships that define a consumer's view of its cloud computing resources. We have taken care when defining the model to avoid dependencies on any particular hypervisor or technology; the entities combine to produce a *pure virtual* view which divorces the use of resources from the physical hardware and software stack on which they are deployed.

The central abstraction in Figure 1 is the *organization (Org)*, which represents a cloud customer. The organization contains *catalogs* (in which *media* and *vApp templates* are stored), *virtual data centers (vDCs)* and *networks*. An organization can have zero or more catalogs and networks, but must have at least one virtual data center upon which its virtual machines run. VMs themselves are grouped into *vApps* which contain (possibly recursive) definitions of interacting components of a single logical application. VMs within a vApp are connected using *vApp networks* that can themselves connect to the organization's main networks. In the remainder of this section, we will discuss these abstractions in greater detail.

3.2.1 Organization

The organization represents a single customer in the cloud, and is that client's interface to their cloud resources. It serves as a container for user management, access control, VM quotas, storage limits and so forth while sandboxing the customer within the provider's infrastructure. A provider may host many organizations, so long as they are strictly isolated from one another.

It is expected that customers will allow individual users access to cloud resources; as is currently the case, an organization will have an administrator who mediates access to IT for the rest of the company. We therefore provide a role-based access control model within an organization that allows elevated privileges for those in an administration role while restricting the activities of regular users. Further, we control access to virtual machines based on ownership and sharing settings, enabling isolation between individual users. The model allows user management to be performed directly through the organization entity (by adding and removing user entities) or through a company-specific directory mechanism such as LDAP.

The organization administrator can also define policies that control the resource usage of virtual machines within the organization. Storage limits prevent users from provisioning too many VMs, while leases allows old VMs to be cleaned up when they are no longer required. Resource limits prevent runaway processes from exhausting the available CPU and memory which could affect the performance of other VMs as well as running up huge service bills. Finally, flexibility in setting storage and resource policies allow an organization to overprovision the resources allocated to it by a provider, supporting a large population of users at a lower cost.

3.2.2 VMs

Virtual machines with associated storage are the basic unit of provisioning within the cloud. VMs can be manipulated either across the network or through console access. Customers can specify complex VM configurations by uploading disk images that have an operating system and any necessary software pre-installed. Such images can be stored as templates, allowing rapid deployment of new resources and providing the ability to grow a virtual cluster based on demand. Similarly, a user can undeploy or delete VMs in order to shrink the cluster when demand has subsided. Our VM representation includes the processor speed and memory capacity, as well as the available hard disk size. These capabilities are tunable when creating a VM, up to a maximum allowed by the service provider.

Using the VM as our basic abstraction supports the IaaS model; we place no restrictions on the software or PaaS solutions that can run on a given VM. We are also intentionally abstract when defining the characteristics of a VM. Beyond the most basic processor, memory and disk capacities we leave the specification of a VM's capabilities to be determined by the service provider. This allows flexibility in the cloud offerings that can be supported, but does not insist on a particular hypervisor or technology stack.

3.2.3 vApp

A single VM is often insufficient for large enterprise applications, which require multiple interconnected machines that combine to form a single system. We use the abstraction of a vApp to encapsulate the VMs and the networks that make up a single logically-connected *virtual application*. The VMs in such a configuration can be deployed en-masse with a single operation, and can be managed as a unit. Cloning vApps and instantiating vApp templates gives a powerful mechanism by which complex applications can be rapidly deployed.

vApps provide an isolated workspace in which virtual machines can be deployed, modified or removed without affecting the configuration of other vApps. They can be created and deployed by non-administrative users, allowing the members of an organization to consume virtual resources in a self-service manner, conforming to the organization's policies but without direct intervention from an administrator. Access controls can be assigned on a per-vApp basis, isolating the workspace of one user from another, while allowing sharing when configured by the vApp owner.

Networking in a vApp is provided by internal L2 networks which are assigned from a pool of underlying network resources. A vApp network can be connected to the organization network either directly or through an edge router (which can also provide firewalling or DHCP services). By isolating the network behind a DHCP-enabled router, a vApp configuration can connect to the outside world while referring only to internal addresses. As well as the obvious benefit when migrating a vApp between data centers, this isolation allows vApps to be cloned without reconfiguring the internal network for each copy.

Much of the desirable functionality of a vApp relies on a consistent means of serializing the VMs and network configuration that make up its state. We represent serialized vApps using the OVF format, an open standard that represents a collection of virtual machines. Using a widely-implemented standard ensures interoperability between our cloud model and VMs created using other applications.

3.2.4 Catalogs

Fast, self-service vApp deployment requires that pre-configured VM images be made available to users, eliminating the need to install an operating system and applications on every VM in a vApp. Further, uploading the media images (floppy disks and CD ISO files) that may be required by an application can be a slow process, and should be performed only once. We store these common resources in an organization's catalogs.

Each catalog has its own access control, allowing individual catalogs to be shared with some users and not with others. Catalogs can also be published, making them accessible outside the organization. This allows providers to supply tailored vApps that take

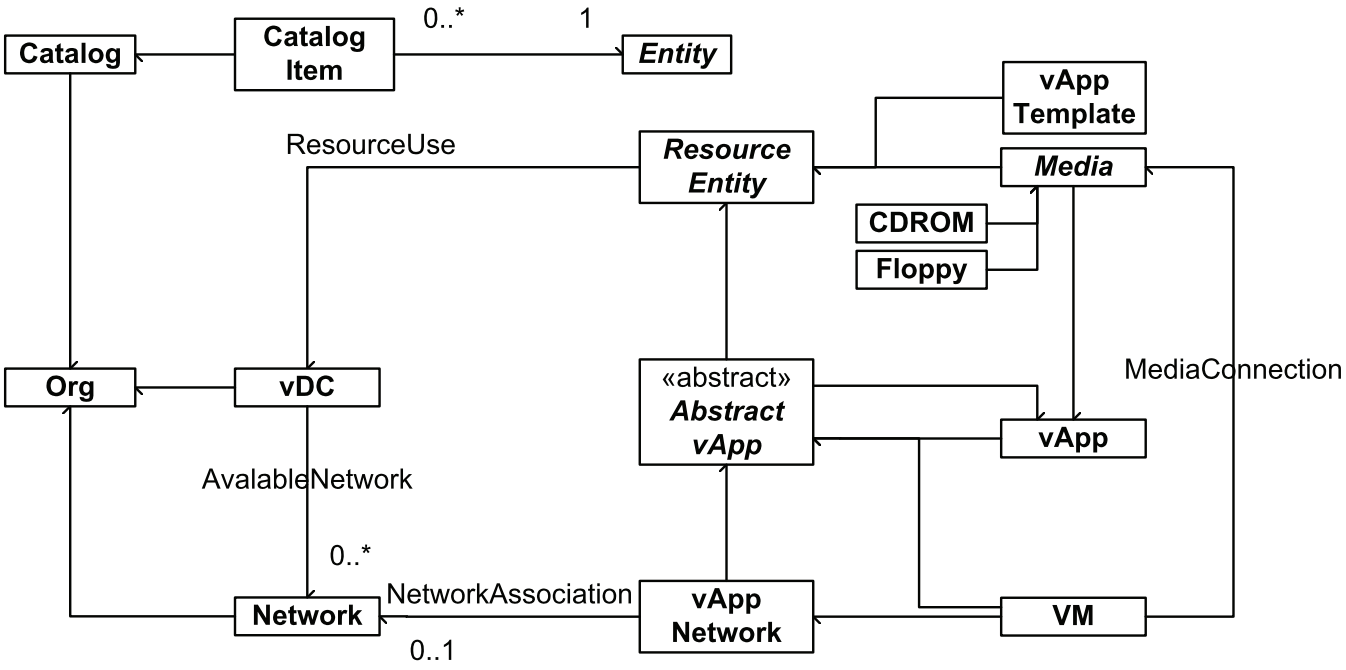

Figure 1: Cloud model entities

advantage of their hardware to all organizations, or for third-party organizations to sell access to useful templates.

3.2.5 vDC

Cloud computing customers have non-functional requirements that can make a significant impact on their applications. For example, a high-performance computing infrastructure may require a high degree of locality and fast network connections in the underlying hardware. A business-critical application may demand uptime guarantees and the option for off-site disaster recovery should a cluster fail, while a university contracting to a service provider for student projects may value cost savings over guaranteed availability. Modeling all possible use cases would necessarily be incomplete, and would limit the flexibility for customers and providers to determine the best service agreements. Instead, we provide the concept of a Virtual Data Center (vDC) that can encapsulate the rich variety of possible service offerings.

A vDC is assigned to an organization by a provider, and is backed by a set of physical resources. The vDC serves to group together vApps that require a given set of features, allowing the provider to provision the appropriate hardware and software configuration to support their needs. vDCs can be created that offer different underlying hardware, enabling specialized applications. An organization can contain multiple vDCs, each of which can come with a different service level. Thus the organization may allocate its critical vApps in an expensive "Gold level" vDC that guarantees availability, while placing its non-essential functions in a separate "Bronze level" vDC that does not have the same features, but costs less per VM.

Organizations can also use vDCs as a means of resource management. Different entities within a company can be assigned separate vDCs, preventing any one from exceeding their share of resources to the detriment of others. Additionally, an organization administrator can perform overprovisioning at the vDC level, allocating resources based on anticipated need on the assumption that some percentage of VMs will be idle at any given time.

3.2.6 Networks

In keeping with the purely-virtual nature of our model, we assign logical networks to organizations in order to connect their VMs. A logical network can be thought of as an isolated L2 network that can connected to another network, either through bridging (forming a direct connection) or routing. Logical networks can have three different scopes: a vApp network connects the VMs in a single vApp; an organization network connects the vApps within a single organization; and an external network connects an organization with the rest of the world.

A vApp network can be bridged to an existing organization network, or can be isolated by fencing. A fenced network is purely internal to the vApp, and is backed by an isolated network resource. Such fenced networks are useful, for example, to allow a vApp to be cloned without updating the individual VMs IP addresses.

We generally expect that service providers will create isolated organization networks for each organization that they host, thus guaranteeing isolation in multi-tenant clouds. The provider can also bridge or route an organization network to an external network in order to provide Internet connectivity or to route back to an enterprise via VPN.

There are a limited number of networking services built into the vCloud model today:

L3 subnet configuration. We associate IP subnet information with a network, which is used when statically configuring a VM's connection.

IP address management. We track the static IP addresses allocated on a network, and can configure a pool of IP addresses to be used.

Edge router configuration. A logical network can be configured with an edge manager that can provide DHCP service, network address translation and firewall functionality.

We expect that providers will expand beyond these services when configuring organization networks, making network features an area in which providers can differentiate their services. For example, we would expect service providers to allocate a small number of IP addresses from the public address space to an organization. The organization administrator can then set NAT rules for packets going in and out of the organization network.

3.3 vCloud API

The second major aspect of our cloud model is the vCloud API through which the abstractions can be manipulated programatically. The API enables third party software developers to produce applications that operate against any cloud provider that supports it. End users are able to create new tools and interfaces to manipulate the cloud objects available to them, while organization administrators can script the deployment and control functionality to simplify their routine operations. API interactions may also be embedded within applications themselves, allowing cloud-aware systems to manipulate the platform on which they run. For example a web service could detect a spike in traffic and automatically deploy new resources to compensate. Thus, the API is critical for self scaling PaaS or SaaS offerings on top of the IaaS offering.

The vCloud API is closely linked to the cloud abstractions that we have discussed, allowing users to programatically manipulate the cloud entities that they control. It therefore inherits many of the desirable properties that exist in the model: it is purely virtual and simple to use, while supporting legacy applications and enterprise users. It is limited in scope to the functionality needed by cloud resource consumers, rather than producers. The producer API necessarily depends on the implementation of the cloud, and we want the same standard to be supported on top of different cloud implementations.

Building upon the entity model described in Section 3, we expose virtual resources that can be manipulated using a REST-based protocol. We chose REST for several reasons. REST is fast becoming a standard by which web services can be consumed programatically, and it works well with existing web infrastructure (such as firewalls and proxies). It is platform-independent, allowing the maximum flexibility when implementing both the provider installation and the client application, it is simple to use and has wide support both in clients and libraries. Finally, a REST API is built around a resource-oriented interface, with objects manipulated using explicit actions. This ties in well with our purely-virtual view of cloud resources, where the implementation of a given action is the purview of the provider rather than the client.

The basic operations in the vCloud API are shown in Figure 2. The API allows users to programatically instantiate, configure and control vApps, as well as query the inventory of an organization. It also allows for catalog management and monitoring of long-running, asynchronous, tasks that execute on behalf of a customer (such as powering on a vApp).

The vCloud API is designed to be extensible, allowing it to serve as a base for new research and development. To this end, all the schemas that it offers are extensible, giving a means to specify new metadata or commands. We enable security in the vCloud API using standard password-based or LDAP authentication, and manage sessions using cookies. This ensures that every operation that occurs through the API is restricted according the the rights of the user executing it.

The major source of complexity in dealing with the API is in the serialization of vApps for transfer between clouds, and in the authoring of vApps. This is not actually a part of the vCloud API; we use an existing standard, OVF, removing this complexity from the user (or implementor) of the API. Instead we can depend on rich tooling available in existing virtualization platforms and in other authoring tools to create and serialize the vApps that we use. For example, vCloud applications can be extracted from existing internal clouds or individual hypervisors, giving a simple means to move computation to the cloud.

vApp operations
POST <vapp-uri>/action/{deploy, undeploy}
POST <vapp-uri>/power/action/{powerOn, powerOff}
POST <vapp-uri>/power/action/{reset, suspend}
POST <vapp-uri>/power/action/{shutdown, reboot}
GET <vapp-uri>/screen
POST <vapp-uri>/screen/action/acquireTicket
vApp Configuration Operations
POST <vapp-parent-element-uri>
DELETE <vapp-element-uri>
PUT <vapp-element-uri>
Inventory Listing
GET <vapp-uri>
GET <vdc-uri>
GET <vAppTemplate-uri>
GET <media-uri>
GET <network-uri>
Catalog Management
GET <catalog-uri>
POST <catalog-uri>/catalogItems
Upload/Download/Provisioning Operations
POST <vdc-uri>/action/composeVApp
POST <vdc-uri>/action/instantiateVAppTemplate
POST <vdc-uri>/action/instantiateOvf
POST <vdc-uri>/action/annotate
POST <vdc-uri>/action/uploadVAppTemplate
POST <vdc-uri>/media
PUT <upload-uri>
GET <download-uri>
DELETE <resourceEntity-uri>
Task Management
GET <tasks-list-uri>
GET <task-uri>
POST <task-uri>/action/cancel

Figure 2: vCloud API: basic operations

OVF is an extensible format which, combined with the extensibility of the REST API, allows for end-to-end support for new features in the system. We expect that, over time, required SLAs will be specified in the OVF format and passed through the API to particular cloud implementations. For example, a VM may have have a requirement for high availability. One cloud may implement this using highly available hardware, while another cloud may exploit VMware's fault tolerant product to achieve the same goal.

We have released the vCloud API open source, and submitted it to the DMTF for standardization. There are already several implementations we are aware of, and several more in the works. For more information on the API see [28, 29]

3.4 Standardization

To create an effective standard, the cloud computing model must satisfy a wide set of use cases, must be simple and common enough to afford multiple implementations, and must obscure the technology of any given vendor. We have argued that the features we present are critical to support enterprise and HPC workloads. Clouds built upon those abstractions will be useful for a far larger set of workloads than are catered to by current web-focused clouds.

The fundamental abstractions needed are simple enough that we have been able to provide a small, easily-implemented interface

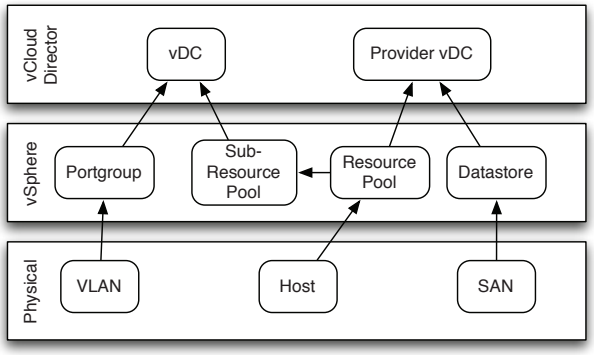

Figure 3: vCloud Director and the vSphere Stack

that supports them. The complexity of the model exists not in the control of vApps but in their specification. For this we rely on an existing standard, the Open Virtualization Format (OVF). This DMTF standard is widely-supported and is available on all virtualization platforms, with authoring tools from multiple vendors.

We have also made the conscious decision to remain agnostic in the model to any specific technology choices. As a result, a provider's use of VMware technologies such as High Availability, Fault Tolerance or Distributed Virtual Switches are not visible to the consumer. The exploitation of such services are implicit in the type of vDC that the provider establishes for a customer. In the future we expect to use OVF to specify the requirements of an application from the platform. However we expect these requirements to be general (e.g., that a high degree of availability is required) rather than specifically demanding a particular technology.

4. VMWARE VCLOUD DIRECTOR

VMware vCloud Director (VCD) is an implementation of the vCloud model described in the previous section. It is a layer of software above VMware's vSphere product [9], which includes the vCenter (VC) Server and the ESX Hypervisor [31].

A VC Server provides the administrator with the ability to manage a large number of ESX hosts and VMs. VMs can be deployed to a cluster without specifying the host upon which the VM will run; the VC server will automatically move the VMs between different hosts to meet its resource requirements. The platform provides a very rich API and UI targeted at system administrators and at partners extending the platform.

VCD provides the following functionality on top of the core vSphere platform:

Model support. VCD supports the pure virtual entity model and API described in Section 3. It enables the consumer to use virtual resources without knowledge of how the provider has provisioned that capacity.

Multi-tenant. Providers running VCD can securely host multiple customers on the same physical infrastructure, ensuring that they remain isolated from one another.

Scalability. VCD consolidates multiple vSphere resources, allowing scalability beyond that offered by a single vCenter server.

Self-service. Consumers can control the creation and deployment of virtual resources (within limits set by the organization) through a rich self-service UI.

In this section we first describe how VCD implements the vCloud abstractions on top of vSphere abstractions, and then we describe briefly some implementation details of VCD.

4.1 Layering of Abstractions

VCD forms a thin layer above the existing vSphere stack, orchestrating the activities of multiple VCs while deferring resource management, load balancing, VM execution and so forth to vSphere. Implementing VCD on top of vSphere gave us many benefits: we are able to take advantage of the rich functionality of the underlying infrastructure, significantly reducing our development cost; we allow cloud providers to continue to use the vSphere UI and API tools to administer their physical capacity; we exploit the vast ecosystem of storage, networking and management partners that target the vSphere API, as well as the very broad hardware compatibility list that it supports; and we enable existing vSphere enterprise and service provider customers to introduce VCD incrementally by gradually migrating their capacity.

Figure 3 shows the relationship between physical objects, vSphere abstractions, and vCloud abstractions. The physical layer consists of compute hardware (ESX hosts), network resources (VLANs provided by networking switches) and storage (typically SAN). vSphere builds on top of these resources, creating *resource pools* that consolidate the memory and CPU power required to run a VM, *portgroups* that create logical L2 networks between VMs, and *datastores* that contains a file system on which many virtual disks and associated metadata can be stored.

At the VCD level, the compute and storage resources of a site are represented by one or more *Provider vDCs*. Just as an organization uses a vDC to present resources to its users, a service provider configures a Provider vDC that encapsulates its physical resources. This way a provider can establish a single large provider vDC with a particular service offering, and can sell access to it to many different customers (assigning a vSphere-level *sub-resource pool* to each in order to maintain separation between organizations). In the same way as an organization administrator may overprovision the resources in a vDC, so a system administrator can overprovision vDCs to a provider vDC.

As can be seen from this figure, all vCloud abstractions have a simple one-to-one mapping to an underlying vSphere abstraction. This allows seasoned vSphere administrators to view VCD's abstractions in their familiar and powerful UI, and can manage capacity and allocate resources between customers as they would have within vSphere. We expect that other vendors that implement the vCloud API will similarly build it on top of an existing virtual and/or physical management system.

The creation of a new vDC is not done in a self service fashion. The provider must create the vDC based on an understanding of the customers need, a pricing model, and the physical capacity available. For example, a provider might create several vDCs with different service levels:

- A *bronze* elastic vDC on low-powered compute resources with low-end NAS storage, a 1 GB switch, and an expandable resource pools with a low reservation of resources,
- A *gold* vDC with VMware's *High Availability* product enabled on a high-end SAN with 10 GB Switch, and a expandable resource pool with a high reservation of backing resources,
- A fixed-size vDC with dedicated resources that an enterprise customer can overprovision themselves,

- A highly-secure vDC that is allocated on a dedicated provider vDC with private hosts and datastores to minimize covert channels.

We expect that there will be a great deal of experimentation in how cloud resources are backed by virtual and physical infrastructure, as well as the charging models associated with various services. The vCloud API and entity model allow for this large range of offerings while presenting a single interface for customer tools and interactions.

The unfortunate side of the flexibility we provide for VCD is that it is impossible for us to fully automate a cloud in the same way that special purpose public clouds, with an extremely prescriptive HCL and types of offerings, can. The vCloud API that we described in Section 3.3 targets only the user-level operations within the cloud. This is intentional; we aim for the API to become standardized, and so it does not rely explicitly on the vSphere stack. However, we understand that administrators will wish to automate the maintenance of their cloud. To enable this we have developed vSphere-specific extensions to the vCloud API which allows scripting of cluster administration. The combination of this and the the vSphere APIs lets third parties with a prescriptive offering do this automation. Large providers and partners have already started delivering this kind of automation, but we believe that this topic will be a rich area of research for a long time.

4.2 Implementation

Figure 4 illustrates the implementation of VCD. The majority of the system is implemented by one or more *cells* that run the VCD software. Cells are stateless; all state is stored in a shared (possibly clustered) database. This means that the system will continue operating if any cells fail, and that new cells can be easily added behind a load balancer if the demands on the system increase.

A single VCD site can orchestrate the activities of multiple vCenter servers, allowing a unified management interface to a massive pool of computational resources. Alternatively, a site can be as small as a single VCD cell managing one vCenter server with a single host. This way an organization can stand up a small private cloud, using the same technologies and model as they would see from a large cloud provider, at minimal cost. In the small scale deployment the installation is simple, with few external dependencies. On a large deployment the provider can use complex database clustering technologies and a sophisticated load balancer to achieve scale and performance. Such a range of scales has many advantages, including the ability for enterprises to experiment with the technology before making a larger investment, isolating part of an organization's computation that must remain in-house, and enabling specialized small- to medium-sized providers to enter the market and compete with larger operations. Additionally, the ability to create a small VCD site may encourage academic institutions to experiment with and extend the technology within the limited resources commonly available to researchers.

Cells communicate through three shared media: the VCD database, a message bus and a shared file transfer area. The first of these approaches is the most common, with status messages and shared data written to the database where it can be read by all other cells. The message bus is used primarily for event broadcast, alerting other cells about a change in the database or vCenter state. Finally, the shared transfer area is used as a staging point when uploading or downloading files to the cloud, isolating external clients from the vCenter servers.

Cell liveness is determined by a heartbeat service running on each cell that periodically checks the database for a signal from each other cell. Any cell that detects a late signal can declare the relevant cell to be failed, at which point the remaining cells will take over its workload. Cells balance workload by writing long-running tasks to the database, allowing another cell with spare capacity to execute a given task. This also provides a mechanism by which the tasks running on a failed cell can be restarted or cleaned up by the remaining cells.

VMs	10,000
Users	10,000
Hosts	1,000
vCenter Servers	25

Figure 5: Supported Scalability of VMware vCloud Director

VCD manages the vCenter servers under its control using the published vSphere VIM API. While VCD oversees the allocation and configuration of virtual machines, vCenter remains the source of truth for many VM operations (such as power state or mounted media). Since VIM calls are relatively expensive, we cache the state of each vCenter server's inventory in the database. One monitoring process per vCenter server is run on a cell in the system (the cell running the process is irrelevant, although we attempt to spread them between cells when a new vCenter is attached or when a cell fails). This process copies the inventory information to the VCD database, and monitors for event notifications that indicate state changes. Relevant events are forwarded to the message bus, allowing all cells to act upon vCenter events. This caching offers a significant performance increase, since inventory queries occur frequently and can be serviced by the database rather than the vCenter server.

VCD cells are generally controllers for the underlying vSphere resources, and are not on the data path. There are few situations where data moves through the VCD layer: during the upload and download of vApps and media; enabling console access to a VM; and when copying vApps between VC servers. The limited use of our cells for data intensive operations has greatly simplified implementing a scalable system. Figure 5 gives published supported scalability numbers for a single VCD site. These numbers are conservative, and are based on a fairly extreme assumption of customer operations. Greater scale can be achieved in practice, although depending on the the nature of the interactions from VCD consumers, performance may suffer.

5. RELATED WORK

John McCarthy's 1961 MIT centennial talk first formulated the idea of utility computing, and Nicholas Carr's book "The Big Switch: Rewiring the World, from Edison to Google" [7] gives a non-technical discussion of the trend toward outsourcing computing infrastructure. Both provided high-level inspiration for this work.

Of the three cloud service types discussed in Section 1, IaaS is both the newest and the most loosely-defined. A detailed ontology of this view of cloud, demonstrating a dissection into five main layers as well as the inter-relations and inter-dependencies on preceding technologies was performed by researchers at IBM [33]. This comprehensive work identified several layers and abstractions that make up Cloud Software Infrastructure: IaaS for computational resources; DaaS for data (or storage); CaaS for communication; and HaaS for the hardware and firmware that underlies them. We view these separate layers as part of the larger classification of IaaS. Customers should not have to handle these discrete abstractions explicitly, rather they should delegate that to the IaaS provider. This allows for a simpler consumer model and for far greater economies

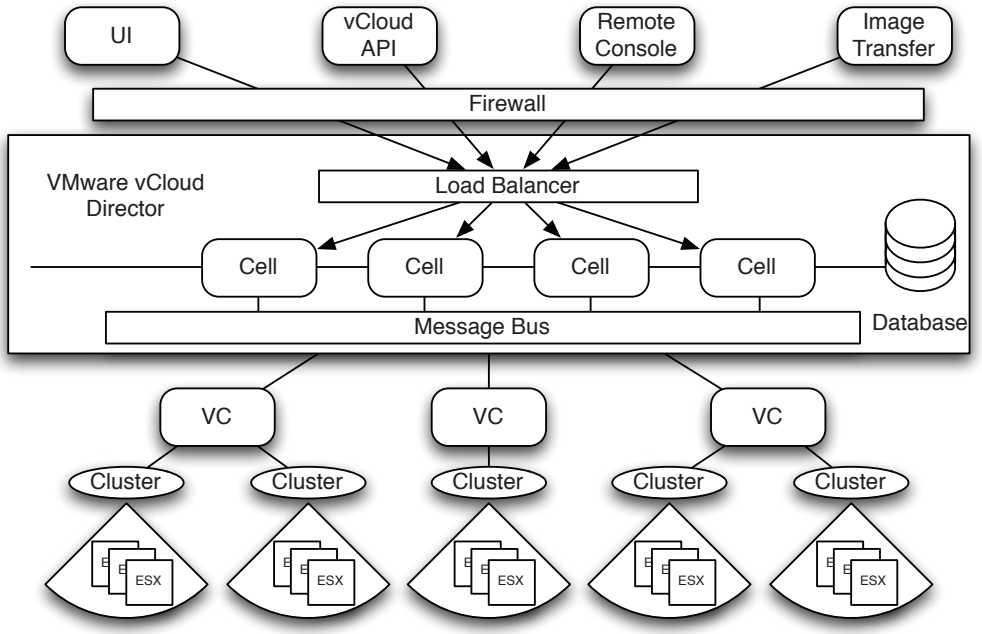

Figure 4: VCD System Diagram

of scale for the provider. In VCD these issues are handled in the vSphere layer, but for completeness we will address them.

Computational services and various virtualization technologies (**IaaS** by the IBM ontology) include bare metal virtualization [6], paravirtualization [6], hardware-assisted virtualization. These have commonly been used on individual nodes in order to improve lower-level resource sharing, utilization, isolation and SLA guarantees. However, there are several examples where computational infrastructure is offered on its own, with the other components required for a true IaaS offering provided separately. Examples of this model include Amazon's EC2 [1] and Enomaly [8].

Communication or networking as a service (**CaaS**) is a clear requirement for providing the quality of service that many applications need. Not surprisingly, most of the work in this area has occurred in commercial clouds [13, 10, 12, 17]. We believe that it is ultimately better to let customers manage their own *virtual* networks as stated in section 3.2.6, specifying their networking requirements in the vApp. This enables IaaS service providers to deploy it as they see best on their hardware in order to meet these requirements.

Storage or data as a service (**DaaS**) is provided in different ways by several vendors. Some provide a simple backup and restore solution, with others offering a large archival cloud-based blob store. While this model provides a means of storing data, customers quickly find that they want to do more — to run applications such as data analytics on their stored data. Thus, DaaS storage must be closely co-located with compute facilities, as in the case of Amazon's S3 service or AT&T Synaptic. Unfortunately, simple storage is not sufficient for many applications, so IaaS Providers must add more data services (databases, block storage or messaging), and tools for importing and exporting the data to them.

Ultimately, DaaS is attempting to solve two separate problems. The first is provisioning local storage for VMs to run on, to use for scratch space and to store applications. The second is sharing data between VM applications, or storing it for later accesses. We believe that the first requirement should be the purview of the IaaS provider rather than the customer. Providers can match the VM application to storage that will meet customer requirements (possibly providing the storage as a VM application [11]). The second issue may require specialized underlying hardware resources (for example HPC applications in the cloud [18] which may require low latency networks and some special capabilities like RDMA). Such storage solutions can be provided by several products. Ultimately this case can be thought of as any other specialized market where individual providers can tailor an offering to their customers' needs. The marketplace of clouds model that we advocate will simplify the deployment of such offerings.

Hardware as a Service (**HaaS**) has some appeal to enterprises who want to set up an internal cloud. The key part of the solution becomes its management. IBM's project Kittyhawk [4] is based on the highly-integrated architecture of the Blue Gene supercomputer, and explores how to provide a platform that is an order of magnitude more efficient than those built on commodity servers. Similarly, CMU's Fast Array of Wimpy Nodes (FAWN) [3] uses many small, energy-efficient servers to produce a fast and scalable architecture.

Ultimately, we believe that the added burden of dealing with separate component services exposes users to extra cost, hassle and errors. These layers are better handled by IaaS service providers who can present a single integrated solution to their customers.

Armbrust *et al.* is a great overview from a set of researchers at Berkeley on cloud computing, showing its promise, reality, products, cost models, largest opportunities and obstacles [5]. While we share the long term vision of massive shared utilities, we believe that small internal clouds will offer value to many customers even in the short term.

The threat of cloud mono-culture had been discussed in various papers and blogs [20, 15]. The impact of even a small problem for

major IaaS service providers has been front page news [23, 14, 32], such as the "#Googlemayharm" bug which caused Google search problems for several hours [26], and Amazon's S3 outage due to a single bit error that lead to gossip protocol blowout [24]. While most of the examples seen so far have been PaaS failures, they illustrate the danger of vendor lock-in for all types of cloud computing: customers had no way to get these services from other service providers, or to move their applications to another service provider. While the reliability of major service providers is still much better then most individual IT shops, enabling customers to move their virtualized applications between providers or to run on multiple sites will allow higher availability and resilience to such failures.

The Tashi cluster management software [16] is an open-source platform that divides the resources of a single physical cluster between many virtual clusters that consist of VMs with differing resource requirements. Tashi is focused on high-performance computing, and aims to efficiently manage the common set of physical resources among its various virtual clusters. Eucalyptus [19] provides an open-source software platform that implements IaaS-style cloud computing using the existing Linux-based infrastructure found in a modern data center. It is interface-compatible with Amazon's AWS [2] making it possible to move workloads between AWS and the data center without modifying the code that implements them. This allows users to setup a private cloud or another provider to setup a public cloud based on Eucalyptus and provide some of the same services as AWS. These research projects, and many industry efforts, have focused on the implementation of a cloud. While these projects are competitive with our VDC implementation, they are complementary to our goal of establishing a competitive marketplace of clouds with many different implementations. We have been actively discussing the vCloud API with many vendors of cloud infrastructure in the DMTF and are making substantial progress on establishing a standard interface for IaaS cloud computing.

6. CONCLUDING REMARKS

General purpose cloud computing has the potential to revolutionize how IT services are produced and consumed. In this paper we have argued that a wide marketplace of IaaS providers will enable flexibility and innovation in the services offered, and avoid the dangers of a provider mono-culture. A marketplace of providers running different implementations of a common interface will enable innovation and strongly-differentiated services, while allowing customers to choose the provider and offering that best suits the customer's needs. As well as enumerating the advantages of such an ecosystem we have discussed the challenges involved in building and promoting cross-compatibility between cloud providers. We have described the fundamental set of abstractions that we believe all IaaS cloud providers must support, and extrapolated from them a purely-virtual model that will provide a common interface while leaving the flexibility needed for providers to innovate and differentiate their services. We also discuss an open-source REST-based API that manipulates these abstract entities. Finally, we have presented VMware's vCloud Director: a currently-available implementation of our model that allows providers to quickly and easily set up cloud offerings that can scale from a handful of machines to massive data centers.

The model, API and implementation that we have presented offer a deliberately minimal interface to cloud services. One of the major advantages of a wide marketplace is the opportunity for research and experimentation into new techniques and approaches to cloud computing. We have been encouraged so far to have seen VCD installed in several research institutions, as well as in commercial enterprises, and we would encourage other researchers to build upon the platform. In the remainder of this section, we will elaborate on some of the open research questions that our infrastructure could support.

Automation at scale. The vCloud API gives a simple and standardized means for third-party developers to create rich tool suites that operate against cloud installations. This will enable research into the automation, instrumentation and monitoring of large cloud instances, as well as the security implications of such tools.

Federation between clouds. We propose a standardized approach for cloud producers to offer computing resources, allowing consumers to pick and choose the services that they require. However, organizations may want to span multiple clouds in order to take advantage of varying prices, services and so on. There is interesting work to be done on the abstractions that can be built on top of those in the current version of VCD to address the security, scalability and performance issues of merging multiple clouds.

Fungible computing. While our abstract, purely-virtual model enables customers to compare and reason about cloud offerings, the reality is that all computing resources are not equal. Our extensible interfaces leave room for new representations of the underlying infrastructure to be communicated to users, allowing a more informed choice regarding cloud services. The questions of how to present metrics, physical capacity and requirements remains to be addressed. There may also be research opportunities into more market-driven concerns such as pricing models, as well as the implications of computing as a resource including trading, futures and arbitrage.

Enabling HPC. The requirements of the HPC community are very different from those developing web or enterprise applications. However, cloud computing has the potential to revolutionize the HPC market, providing the affordable short-term access to massive computing resources that will enable individual researchers to perform experiments that would previously have required an unreasonable hardware outlay. Optimizing provider and organization vDCs for the needs of high performance computing will represent a major step towards this goal.

Augmented desktops. While much of the discussion in this paper has focused on commercial use of the cloud, there are also potential benefits for the individual user. The most apparent is a cloud-enabled desktop, where a profile can follow the user between work sites. Realizing such functionality will cover a wide range of research topics, from security and administration concerns to efficient protocols for streaming 3-D graphics across the network.

Replacing the OS. Finally, while we have based much of the presentation of our cloud vision around the traditional model of a virtual machine (including CPU, memory, operating system and applications), there is no reason why a component VM within a vApp must include the overhead of a traditional OS. Lightweight, custom operating systems that serve a single function within an enterprise system have the potential to improve performance, and better exploit underlying resources for the specific needs of the application.

7. ACKNOWLEDGEMENTS

The work described in this paper is the end result of several years worth of effort by a large team, with significant contributions made by far too many individuals to list. The authors listed should be recognized simply as those who wrote the text of the paper, and not as the intellectual source of its content.

8. REFERENCES

[1] Amazon elastic compute cloud. http://aws.amazon.com/ec2/.

[2] Amazon web services. http://aws.amazon.com.

[3] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: a fast array of wimpy nodes. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, New York, NY, USA, 2009. ACM.

[4] J. Appavoo, V. Uhlig, A. Waterland, B. Rosenburg, D. Da Silva, and J. E. Moreira. Kittyhawk: enabling cooperation and competition in a global, shared computational system. *IBM J. Res. Dev.*, 53(4):598–612, 2009.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[7] N. Carr. *The Big Switch: Rewiring the World, from Edison to Google*. W.W. Norton & Company, 2009.

[8] Enomalism elastic computing infrastructure. http://www.enomaly.com.

[9] K. Govil and R. Soundararajan. Challenges in building scalable virtualized datacenter management. *SIGOPS Oper. Syst. Rev.*, 44(4), 2010.

[10] A. H and et al. Network communication as a service-oriented capability. In *High Performance Computing and Grids in Action, High Performance Computing and Grids in Action*, March 2008.

[11] J. G. Hansen and E. Jul. Lithium: virtual machine storage for the cloud. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 15–26. ACM, 2010.

[12] J. Hofstader. Communications as a service. http://msdn.microsoft.com/en-us/library/bb896003.aspx.

[13] W. J and et al. Perfsonar: A service oriented architecture for multi-domain network monitoring. In B. B and et al, editors, *ICSOC Series Lecture Notes in Computer Science*, volume 3826, pages 241–254. Springer-Verlog, 2005.

[14] T. Jackson. We feel your pain, and we're sorry. http://gmailblog.blogspot.com/2008/08/we-feel-your-pain-and-were-sorry.html, August 2008.

[15] M. Juneja. Security in the cloud. http://www.expresscomputeronline.com/20100208/coverstory01.shtml, 2010.

[16] M. A. Kozuch, M. P. Ryan, R. Gass, S. W. Schlosser, D. O'Hallaron, J. Cipar, E. Krevat, J. López, M. Stroucken, and G. R. Ganger. Tashi: location-aware cluster management. In *ACDC '09: Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 43–48, New York, NY, USA, 2009. ACM.

[17] Microsoft connected service framework. http://www.microsoft.com/serviceproviders/solutions/connectedservicesframework.mspx.

[18] M. F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis. Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, 40(2):8–11, 2006.

[19] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131. IEEE Computer Society, 2009.

[20] A. Nusca. What if there were no google? a lesson in monoculture. http://www.zdnet.com/blog/btl/what-if-there-were-no-google-a-lesson-in-monoculture/12106, 2009.

[21] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, 2010.

[22] Oracle platform for saas. http://www.oracle.com/us/technologies/saas/index.html.

[23] A. Stern. Update from amazon regarding friday's s3 downtime. http://www.centernetworks.com/amazon-s3-downtime-update, February 2008.

[24] T. A. S. Team. Amazon s3 availability event: July 20, 2008. http://status.aws.amazon.com/3-20080720.html, July 2008.

[25] D. Thaler and C. Hopps. Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991 (Informational), Nov. 2000.

[26] K. Thomas. Thoughts on google monoculture and the cloud. http://www.realstorygroup.com/Blog/1490-Thoughts-on-Google-Monoculture-and-the-Cloud, 2009.

[27] J. Touch and R. Perlman. Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement. RFC 5556 (Informational), May 2009.

[28] VMware. The vCloud API. http://communities.vmware.com/community/developer/forums/vcloudapi.

[29] VMware Inc. vCloud API programming guide, 2009, 2010.

[30] VMware, Inc. *vSphere Resource Management Guide: ESX 4.1, ESXi 4.1, vCenter Server 4.1*. 2010.

[31] C. A. Waldspurger. Memory resource management in VMware ESX server. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI-02)*, Operating Systems Review, pages 181–194, New York, Dec. 9–11 2002. ACM Press.

[32] S. Wilson. Appengine outage. http://www.cio-weblog.com/50226711/appengine_outage.php, June 2008.

[33] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *GCE '08: Grid Computing Environments Workshop*, pages 1 – 10, Austin, TX, Nov 2008.