White Paper

The Role of Layer 4-7 Services in Scaling Applications for the Cloud-Computing Data Center

Cloud computing is increasing demands on applications and the application-delivery infrastructure must change to meet the challenge

Abstract

Cloud-computing applications are characterized by stateful access, with differentiated service levels, charged to the end user using the pay-per-use pricing model. Implicit in this model is the assumption that a cloud application is always **on**. Scaling the cloud delivery model to an Internet scale (millions of users) is a challenge that next-generation Layer 4-7 infrastructure needs to overcome.

Scaling a cloud application involves scaling three mechanisms: location (mobility), replication, and load balancing. Virtualization was an early catalyst for cloud computing because it substantially lowered the cost of replication and mobility of a prepackaged application. It does not, however, solve the load-balancing problem. Load balancing involves scaling the address, name space, transport, session, identity, and business logic of the application. Clustering enables scaling of application business logic but leaves the rest of the problem to a proxy infrastructure.

Traditional data center proxy infrastructure (Layer 4-7) is stateless and focuses on transport-level session and content management. The state management in these proxies is limited to cookies that hold bare-minimum identity of the application server. Further, this state is not shared across a network overlay of proxies to enable an application for multi-data center applications or to distribute the processing to edges. For example, a load balancer does not a play any role in user customization in a data center application such as encryption of private data in the response. Not having this function in the proxy infrastructure means the data center has to incur a higher cost of development of the core application and is not able to use existing proxy infrastructure to deploy this function.

In the cloud-computing model, waiting for the application developers to add marginal functions to a cloud application (for example, user customization) will derail the business model. Adding functions to a cloud application need to become the focus of proxy infrastructures in a cloud data center. To play this role, a load balancer needs to extract intelligence from metadata and integrate with infrastructure in the data center at layers higher in the Open Systems Interconnection (OSI) stack than Layer 2 or Layer 3.

The current thinking in cloud computing is to disaggregate the load-balancer function and assimilate it into the switching infrastructure, hypervisor, and the virtual machine itself. One way to assimilate this function is to extend its footprint at the edge, offload stateless functions into the upstream switching and routing infrastructure, and subsume stateful functions that are currently part of the application server. In other words, convert the proxy infrastructure into a control-plane function with a distributed data plane. The evolution and eventual integration of the control plane into a cloud manager is a topic of discussion for thought leaders in the industry today.

This white paper discusses the challenges in scaling a cloud application and the role Layer 4-7 services can expect to play to scale application services in a cloud data center.

Evolution to the Cloud

The dominant model for application delivery today, the web application model, was triggered by advancement in technologies that enabled platform-independent representation of input and output of an application. Having a platform-independent representation of application data enabled creation of the browser, which over time extended itself beyond simple rendering of mark-up text to a mini-run-time interpreter for fourth-generation languages such as JavaScript. Code migration from back-end server to the browser enabled an application to initiate multidomain data-access connections and enhanced the application interactivity afforded by asynchronous interactions with the back-end web server. HTML 5.0, which is now supported by Internet Explorer and Firefox, enables standard browsers to render rich-media graphics and support models of interaction that are currently possible only with proprietary technologies such as JavaFX, Silverlight, and Flash. These technologies, which were initially greeted as a security hazard, are today looked upon as a building block to deliver fully functional applications resident on a server cluster - in other words, cloud computing. The ultimate goal of cloud computing is having a stateless client with varying sizes of caches and a changeable identity accessing applications resident across WAN distances in a stateful manner. This concept is not new - it has been the preferred application-delivery model since the advent of Web 1.0.

The evolution to the cloud started with a client-server model of the late 1980s when applications were resident on a server within a Domain Name System (DNS) domain. What has changed is that the data center is the new server, and it is located several DNS recursions and WAN hops away. The economic factors that led the industry away from mainframe to client-server computing continue to be in effect and, in fact, have been catalyzed by the widespread use of virtualization in data centers. However, simply hosting an application in virtual machines is not cloud computing. The critical difference is the application composition. Virtualization has not changed the distribution, composition, or invocation of the application. **In other words, virtualization has not changed the usage model of the application, nor has it changed the run-time model of the application - what it has done is changed the management model of the infrastructure of an application.** In contrast, cloud computing aims to make the application itself the infrastructure by offering the application components as services (Platform as a Service [PaaS]). The application itself transforms into coarse-grained (or lightweight) services that can reside anywhere (mobility or location), can be invoked from anywhere (Internet scale), and can reside on any device. The management of these services is governed by a service-level agreement (SLA) between the cloud and the end user, and a substantial portion of the management function is delegated to the end user.

Like all disruptive innovations of the past in the data center, cloud computing too arrived unexpectedly. Virtualization - that is, a continuation of a prior trend toward server consolidation - created a bigger problem that it solved. The ratio of administrator to (virtual) machines rose to a level such that administrators themselves started to ask for automation. This advance to automation of a virtual data center is now serving as a catalyst to the creation of cloud data centers and cloud applications. It is now widely accepted that the road to cloud computing goes through a virtual data center. The debate has shifted to how best to design the next-generation data center where the design center is the data and not the application. Shifting the design center has wide ramifications on the architecture, organization, and operational model of the data center. Table 1 lists the important differences between virtual data centers and cloud data centers from an application perspective.

Table 1. Differences Between Three-Tier Data Center, Virtual Data Center, and Cloud Data Center from Application Perspective

Application Characteristic	Traditional Three-Tier Data Center	Virtual Data Center	Cloud Data Center (Public or Private)
Development model	• Structured data model with logic implemented as components bundled with the execution environment	• Same as three-tier data center	• This unstructured data model has logic implemented as services on top of the data store, composed into an end-user application by the user • The application is written to a PaaS that integrates multiple disparate data models and offers a single large data table with name=value pairs • Logic executes on input/output card containers, and the end user selects the view
Deployment model	• Separate model (data) from the view (presentation) from the controller • Results in a three-tier data center	• Same as three-tier, but affords co-location of three tiers on a single physical server (through isolation)	• Model differs by data center • Unstructured data is stored in a data store, indexes to which are stored in Relational Database Management Systems (RDBMS), and data is exposed using batch jobs. In other words, data is the design center of the data center optimized for querying • Applications are lightweight services that are deployed using a developer portal onto a cluster, and presentation is an end user-customized template that is fetched upon access
Execution model	• Fine-grained distribution across tiers of data center with fine-grained access to data • Application servers: Optimize connection management across tiers; are enhanced using proxies	• Same as three-tier but affords mobility of execution environment across physical server boundaries (migration) • Enables elasticity of execution environment modular support of operating system	• Users have coarse-grained access to data in batches • Services work off shared storage • Application servers are disaggregated into pools of Java Virtual Machines (JVMs) running Plain Old Java Objects (POJO) files • Session management of the application server is separated out as a session manager • The proxy layer queues requests and is not in the path of responses
Access model	• Optimized for browser and desktop • Centered on synchronous HTTP access to data over a portal	• Same as three-tier; however, intravirtual machine traffic needs to be redirected for proxy function processing	• This model supports asynchronous access and cache synchronization, in addition to portal access • The access controller is the session manager • Proxy exists only to negotiate network topology
Security model	• Focused on transport security, identity management, and sandboxing of execution environments	• Same as three-tier; however, intravirtual machine traffic is not subject to security policies	• This model is a work in progress • A private cloud is posturing as a security model, but in reality it is not
Management model	• Separation of roles into server administrator, network administrator, and storage administrator, each with its own models • Cross-function management done using workflows	• Forces a convergence of management models because of virtualization • Server administrator subsuming network administrator role because 80 percent of the traffic stays inside a data center	• Support for a self-service model creates the “user” as an additional administrator

The important point to note is how little effect virtualization has on the application itself. The effect of virtualization has primarily been on the resource-management portion of the execution, deployment, and management model of the application. Introducing elasticity into these models enables an application to scale in location and replication dimension. The third dimension for scalability is elastic load balancing.

Why Cloud?

Cloud computing is gaining acceptance because it offers something to every stakeholder in the IT ecosystem. It appeals to IT vendors because it offers them revenue growth even as it reduces their cost of operations. It appeals to value-added resellers (VARs) and distributors because it has built-in support for syndication (PaaS). It appeals to developers because it preserves their skill set by not reinventing a new language (such as Web 2.0) and offers tools for migration of their current applications to a cloud. Finally, it appeals to the end user because cloud services are priced as a utility (pay per use). Success of Apple's iPhone iTunes and AppStore, Netflix, and Amazon Web Services (AWS) are examples that validate the compelling economics that are accelerating the industry's march to cloud computing. However, getting there is not easy.

The concept of cloud computing is deceptively simple. All variations of the cloud-computing model call for a fundamental shift in the way the IT industry develops, markets, and distributes its products. The cloud application architecture calls for refactoring of all layers in an application stack - from the physical network to packaging of an application, to its delivery. The critical challenge to this refactoring is the lack of scalability. Cloud applications require orders-of-magnitude increases in the scalability of an application, with concomitant increases in its availability.

Increasing scalability of an application requires identification of the critical shared resource and increasing its bandwidth through scaling the properties of the critical shared resources such as address, name space, transport, identity, and business logic. At every layer of an application stack, we have critical shared resources, and if we are to have faith in our layering (that is, the critical resource is isolated in a layer), then we could attain scalability goals through scaling of the shared resource. Table 2 identifies some of the critical shared resource in the layers of the cloud application stack. It highlights their limitations and estimates the scalability requirements in a cloud-computing environment.

Table 2. Application Scalability Requirements for Cloud Computing

Layers in Application Stack …	Are limited today to …	For Cloud Computing They Need to Scale to ...
User interface	• Limited to desktop bitmapped display	• To any device with or without display
Presentation data format	• Limited to HTML 4.0 and proprietary solutions such as JavaFX, Microsoft Silverlight, and Adobe Flash (Air); this model constrained the development of applications for the browser	• HTML 5 open standard, which is supported by IE 8 and FF Note: Rich graphics will no longer have plug-ins. The web applications will not be the same again.
Application logic	• Limited to thousands of simultaneous users	• Scale to millions of users Note: Cloud computing signals the rebirth of high-performance computing (HPC) and parallel computing.
Operating system services	• Monolithic kernel • Services limited to kernel space • Support for user space services, but support is kludgy • Resource name space limited to an instance of the OS and sometimes to a process space	• Exokernel (Xen) or Hybrid Kernel (VMware) • Services scaled to data center domain Note: The name space must be extended to include the entire application domain, which can potentially straddle a data center boundary.

Layers in Application Stack …	Are limited today to …	For Cloud Computing They Need to Scale to ...
Delivery services	• Server load balancing (SLB) (proxy)-based traffic redirection with limited intelligence • State limited to cookies Note: All users are equal, and services are not differentiated. Note: Content-delivery networks (CDNs) are used only for content distribution. Note: There are no application-specific proxies.	• Distributed data paths with a single control plane; the system scales with the number of data paths • To enforce differentiated access based on cloud state such as type of user, type of access, type of device, etc. Note: CDNs are used extensively for applications and content. Note: There are numerous application-specific proxies, such as for Google AppEngine, Nirvanix, and Amazon EC2.
Fabric services	• Storage services abstraction limited to file system and database • Load balancing limited to LAN traffic flows Note: I/O is between two network stacks on two OSs between two endpoints. Note: The network is configured to remove redundant paths between endpoints. Note: Quality of service (QoS) is enforced on packets on the network and defined at network stack-layer granularity.	Note: I/O is an application process to a server process, increasing the use of direct memory access (DMA) and Cisco® Remote Data Memory Access (RDMA). Note: The network is configured to increase the number of paths between endpoints. Note: QoS is enforced at flow level and defined at thread-level granularity. Note: Storage services abstraction includes unstructured blobs and tables. Note: Load balancing includes LAN traffic flows, I/O accesses, and jobs.

The important point from Table 2 is that scalability affects the critical business metric in a cloud-computing business model; that is, average revenue per user (ARPU). In other words, any product or technology that undermines the scalability or reduces the availability is highly unlikely to be deployed in a cloud data center. Fortunately, cloud application architecture is layered and interlayer communication is accomplished using open standards. Each layer of the architecture has a multiplicity of instantiations and is scaled independently of the other layers.

Scalability Is Not Easy

Scaling a system means there is no drop in its performance as the number of users increases. The naïve way of scaling a system (dynamically) is to add capacity upon demand. This model is a close cousin of the widely used static capacity planning (over-provisioning). In a cloud infrastructure, however, the business value of the added capacity is a new factor that needs to be included in the capacity-planning equation. If the additional demand has no business value, the system should not allocate more capacity. If it did, the cloud-computing business model will fail. The artifact that ties business value of a customer to the infrastructure operations is the SLA. In a cloud-computing environment, the system should scale as prescribed by the SLA that the data center has in place for that user. In other words, the system should remember the user or store the “state” of the user.

Scaling “state” is the biggest challenge to scaling any system. The web application architectures of the Web 1.0 era circumvented this challenge by eliminating the state (dumb browser), but quickly found that the application is not very useful when it does not store state. The state was brought back into the architecture through kludges such as session cookies. State is scaled through two mechanisms, replication and sharing. To be useful, replicated state is refreshed through complicated coherency mechanisms. These coherency mechanisms, however, are not very scalable. The alternative to replicated state is to share a single copy of the state, thereby removing the refresh problem but introducing a complicated access-control mechanism to prevent thrashing. The good news is that access mechanisms are more amenable to scaling than coherency mechanisms.

Scaling the access mechanisms of an application to Internet scale (millions of users) involves scaling its location (mobility and distribution), its replication (provisioning and resource management), and its load balancing (instance resource optimization). Cloud computing adds a new dimension to scaling: self-service management. A cloud application offers self-management to its users. Scaling an administrative system to a million users as they set their preferences and change SLAs is a critical challenge in scaling cloud computing. In fact, this dimension has ramifications on the traditional three dimensions in that this dimension nullifies the concept of standard configuration. In a cloud application, the configuration of the application is determined by the end user, and the system has to support mechanisms that allow the user to change the configuration. In other words, all users are super-users for their application and its underlying infrastructure.

Table 3 describes the critical best practices along the four scalability dimensions in a cloud application. The important point to note is how the load-balancing dimension is disaggregated by the type of resource to which it is optimizing the access. Also noteworthy is how the user as an additional administrator is being handled through changes in the cloud-delivery models.

Table 3. Scalability Dimensions and Best Practices

Scalability Dimension	Infrastructure Best Practices
Location (mobility) • Organization of an application is transparent to the end users • Components of the application can be co-located or distributed	Location-independent packaging; for example: • Component-level packaging for application servers • Virtual appliances Asynchronous communication; for example: • Components engage in non-blocking communication using messaging Distributed execution • Components are executed in parts and results collated to create response for end users
Replication (provisioning and resource management) • Execution infrastructure is transparent to the end users • Infrastructure can be local to client, remote on a server, and mobile	• FAST Virtualization • Replication (CDN, caching, and prepositioning) • Redundancy
Use (load balancing)	Location load balancing • DNS load balancing Transport load balancing • Proxy-based SLB and gateways Session load balancing • Server affinity • Virtual worker and real worker Compute load balancing • Hypervisor
Self-service management	• Multitenancy • Private clouds • Virtual private clouds

As the load balancer is disaggregated, it is important to stay within the scope of the function. Disaggregation does not mean aggregation of adjacent functions. There is often some confusion between scope of load balancing and resource management. Load balancing optimizes the use of an instance of a resource, whereas resource usage concerns itself with optimal size of the portfolio of the resources or capacity available to meet demand. Separation of these two dimensions and disciplines is important for cloud computing. If a load balancer subsumes the role of a resource manager, it leads to suboptimal load balancing and resource management. It leads to augmentation of capacity to meet a demand whose economic value does not justify such an augmentation. In other words, load balancing operates within a constraint of fixed capacity and tries to optimize the usage of that fixed capacity.

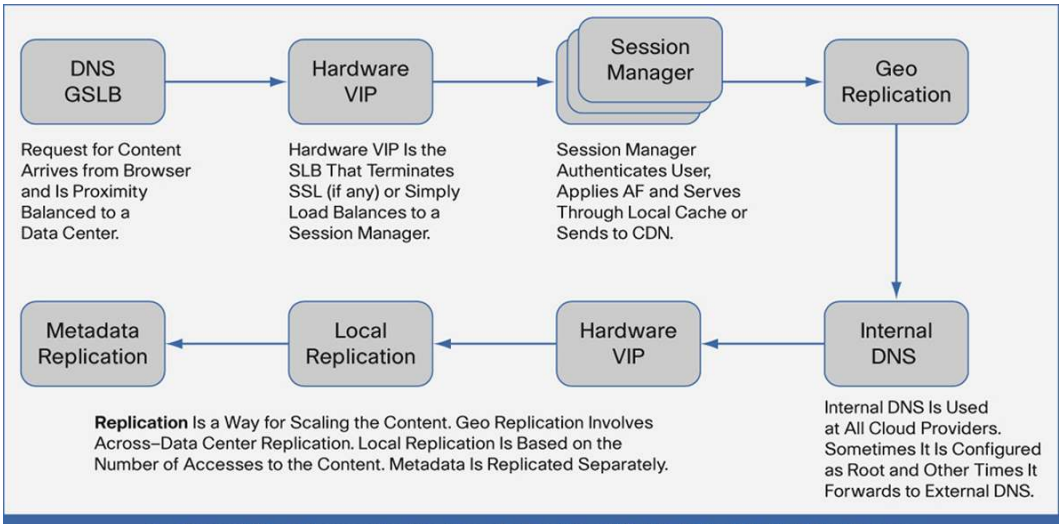
Augmenting existing capacity is a business function that should be implemented in a controller that enforces an SLA. The load balancer is not the best location to place that function.

Revisiting the four dimensions of Table 2 from an application-delivery perspective shows that load balancing is the most relevant dimension. There are two types of load balancing: transactional and computational. Another way of characterizing the difference is to call one **request load balancing** and other **response load balancing**. In the early days of Web 1.0, content was mostly static and thus the request load balancing was the dominant method of load balancing. With the advent of dynamic content, the application response time is increasingly correlated with the optimizations in response load balancing. In Web 2.0, the order of response substantially determines the interactive experience of a web user.

Both types of load balancing involve scaling the location, name space, transport, session, customization, and the logic instance of the application. Several products and technologies interplay to enable optimal load balancing. SLB and adjacent Layer 4-7 services are but a small part of this group of products, as illustrated in Figure 1.

Figure 1 illustrates a typical cloud-storage transaction. Notice that the role of the traditional SLB is that of a request load balancer. It is not expected to play a major role in the session management of the transaction, nor is it expected to be in the path of the response. It is definitely not expected to participate in the resource management of the cloud infrastructure. The critical role of an SLB in Figure 1 is to multiplex requests and help the application negotiate the topology of the physical network.

Figure 1. Typical Cloud Storage-Delivery Application



Cloud computing in essence is explicitly dividing the request load-balancing function from the response load-balancing function. This transaction pattern of a cloud complicates the bookkeeping logic in a traditional load balancer. In a more sophisticated cloud service provider, the request itself is a dummy request to fetch the URL on which the real request is sent. The real URL is not behind a versatile interface processor (VIP) and thus the SLB is not in the data path of the real request. In general, the bookkeeping logic of a cloud-computing SLB is expected to get more complicated. Further, cloud-computing applications are forcing the architects to revisit old and forgotten ideas such as content and protocol negotiation over HTTP, availability of alternatives in content format and

location, content prepositioning (in addition to caching and replication), and URL rewrite-based session management.

Virtualization Is Not a Panacea

Virtualization is the new resource manager in the data center. The dynamics afforded by virtualization is an assumption in the architecture of a cloud data center; in fact, virtualization is the critical enabler of the “elasticity” feature of a data center. However, virtualization has its limitations. The five biggest drawbacks of virtualization follow:

- Its (perceived) lack of security
- Its application-packaging model
- Oversubscription on intra-node bandwidth in a compute-intensive cluster
- Its lack of extension framework
- Layer 2 confinement

These shortcomings undermine the critical selling point of the cloud: economies of scale.

Adoption of clouds in the enterprise is hindered by a lack of confidence in the security of enterprise data in a virtual environment. One way to view virtualization is as three functional blocks: isolation, migration, and application programming interfaces (API; management stations of virtual infrastructure are a reference implementation of these APIs). Several groups have demonstrated attacks on the isolation block of the virtualized environment. The approach is to co-locate a malicious virtual machine with the target virtual machine. Co-location enables the malicious virtual machine to guess network identity of the target and cause denial of service (DoS) or severe disruption in operations of the target virtual machine. The attacks are invisible to external physical security devices because the intra-virtual machine traffic does not leave the physical host. Data centers can experience a spectrum of security lapses from DoS to outright data stealing. The migration block in virtualization sends machine state across an unprotected network. During migration, the data on the network is unencrypted and carried over encrypted transport using super-user privileges. The acknowledged constraint of intra-Layer 2 migration is actually serving as a security mechanism. The API block is perhaps the most vulnerable of the three blocks. The programming model of the API converts virtualized resources into serializable objects. A malicious management station or human error can usurp the identity of a legitimate management station and cause major disruptions. For example, it could use the patch update API to update a secure virtual machine with software that creates vulnerability.

Taking the traditional component-level view of virtualization - that is, the virtual machine, hypervisor, virtual network, and a management station - also validates the security concern in virtualization. Each of these components has been a target of successful attacks from hackers.

Another deficiency of virtualization is the packaging and delivery of the application. Virtualization has standardized the packaging of an application in the form of a virtual machine. Increasing the size of the application package by including run-time components is a well-understood practice. The JVM started this practice by including the language run time with the application. As long as an application (such as DNS, Dynamic Host Configuration Protocol [DHCP], and Lightweight Directory Access Protocol [LDAP]) fits into this packaging model, it will benefit from virtualization. The problem is that most of the current server-side enterprise applications are delivered as components that rely heavily on their run-time containers. Cloud computing is changing this packaging further to the point that components now rely on services in a data center. The services are delivered as components with

their own containers. This reality is the main reason why virtualization in an enterprise stalls at 30 percent of the servers and 10 percent of the applications.

The third deficiency of virtualization is the architecture of the virtual network. Two virtual machines co-located on a physical host do not take advantage of the advancements in the physical network. In fact, they operate on a fixed-bandwidth network with implied oversubscription that is directly proportional to the number of virtual machines on the physical host. This oversubscription makes the virtual infrastructure unsuitable for the compute-intensive applications that are the early adopters of cloud computing. To get any performance parity with physical infrastructure requires refactoring of the application into small virtual machines with local data. The cost of refactoring an application nullifies the main value proposition of the cloud, that is, economies of scale.

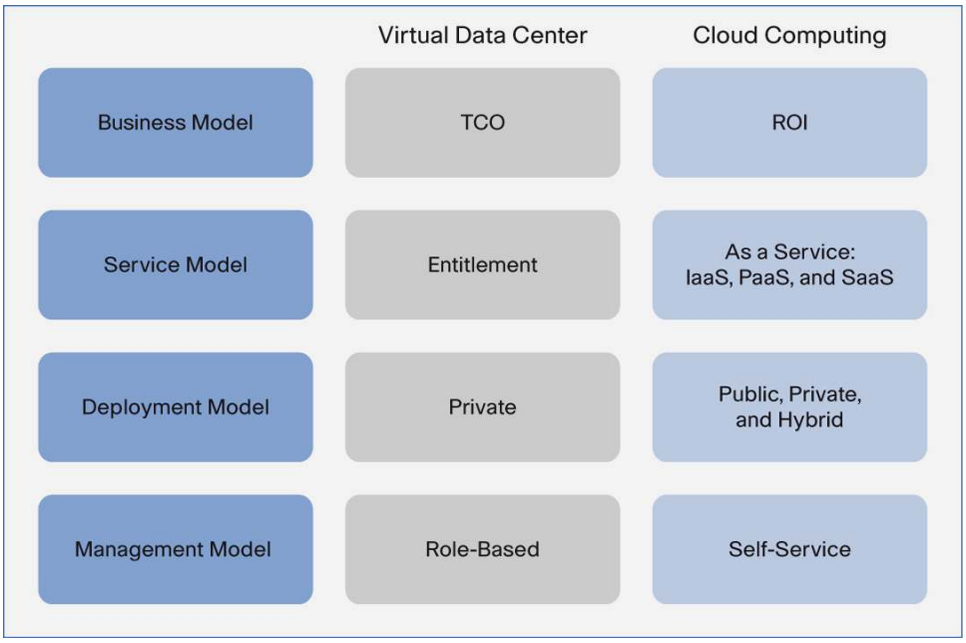
The fourth weakness of virtualization is its lack of extension mechanisms. The design decision to have a thin hypervisor (and sometimes proprietary too) means that all extensions to a virtual infrastructure need to be implemented as virtual machines. The problem with this model is that infrastructure services deployed as virtual machines compete for computational resources with the applications and are vulnerable to all the threats that a virtual machine faces. If the virtual server hosting a DNS service virtual machine crashes, the data center loses not only the application but also the critical network service required to activate the virtual server.

Finally, virtualization confines itself to a Layer 2 boundary. In a cloud data center where the average number of servers is increasing at a 20- to 25-percent rate every year, this shortcoming leads to multiple virtualization domains with their own management. A typical Layer 2 network today cannot support more than 4000 servers, also meaning that a typical cloud cannot offer elasticity beyond 4000 servers worth of compute capacity.

It is for these reasons that successful cloud vendors such as Google and SalesForce.com do not use virtualization.

Figure 2 illustrates the gap between what is required for a cloud data center and what is offered by today's virtual data center. The gap is an opportunity for new products and technologies and offers an avenue for evolution of existing products and technologies.

Figure 2. Gap Between Cloud Data Center and Virtual Data Center



Elements of Cloud Architecture

The leading edge of software engineering has shifted from enterprise Java to social networking software. An important difference between the two is increasing use of events (as opposed to threads) technologies in an application design. Events enable an application to communicate (and therefore scale) across multiple nodes without loss of real-time interaction. It uses code replication to reduce code mobility, uses data caching to reduce data query and transfer, and encourages the use of parallelism over concurrency. In its essence, cloud application is a fusion of HPC distributed computing technologies with enterprise Java and .NET technologies.

This shift away from Java container-based architecture to cloud architecture will affect application packaging and its run-time infrastructure. Java container had abstracted all network services and bundled them with the language run time. Consolidating all these functions into an Out-of-Policy (OOP) language led to the creation of multiple frameworks, which over time sunk under its own weight. In fact, the Web 2.0 trend was nothing more than creation of lightweight containers for web programming. Cloud is the next point in this evolution away from heavy enterprise Java containers to lightweight run-time containers that run in parallel to analyze large datasets.

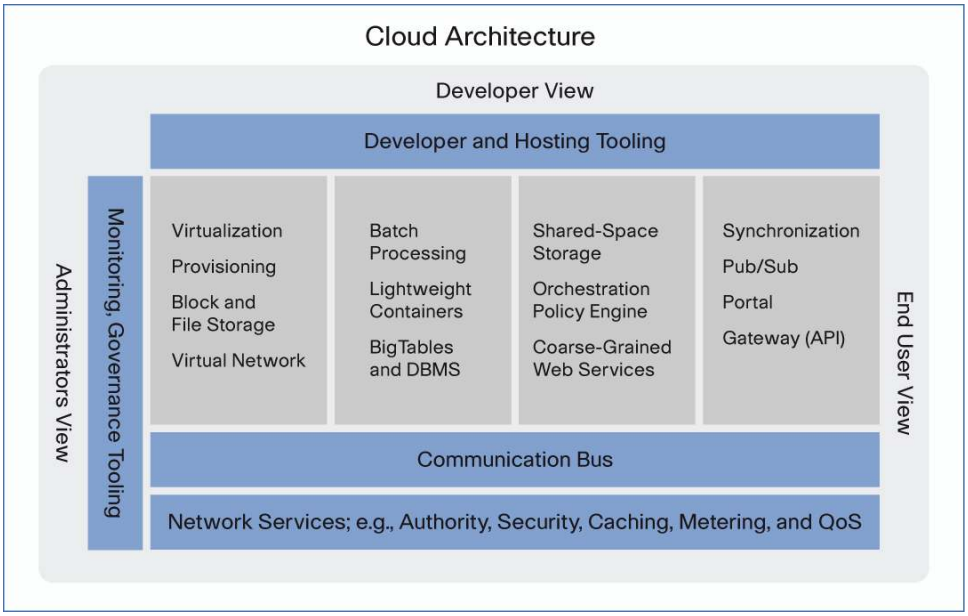
It would be wrong, however, to think that the cloud is creating its own container and run-time environment for applications. In fact, the cloud - through its adoption of virtualization - is borrowing heavily from the Java and OOP world. Specifically, the cloud abides by the design principles of encapsulation (isolation block) and inheritance (provisioning). The current state of the art in the world of virtualization also includes implementation of the dependency injection (templates and cloning) and inversion of control patterns (orchestration and vCenter alerts and alarms) in the virtual infrastructure - essentially creating a programmable infrastructure. This endeavor at lightening of the application container will effect a migration of some functions back into the cloud infrastructure. It remains to be seen if the functions migrate out of the virtual infrastructure and into the physical infrastructure, and more importantly, if it does eliminate the need for an operating system.

Figure 3 illustrates the evolving cloud architecture that is being implemented at various pioneering cloud service providers. What stand out in the cloud architecture is the diminishing role for a language container (JVM or

Common Language Runtime [CLR]), a growing role for large data-access mechanisms, and incorporation of social networking technologies into the mainstream application. Some of the communication functions of the language container are now encapsulated in a communication bus that enables horizontal component communication. The cloud architecture enables a developer to build applications that can analyze petabytes of data and generate kilobytes of reports in formats consumable by desktop-productivity applications.

Large data analysis was the vulnerable point of language containers. In the cloud architecture, that function is subsumed by newer data-analytics technologies such as Map/Reduce. By continuing to offer language run-time containers and the supporting presentation technologies, the cloud in essence scales a traditional enterprise application by a few orders of magnitude, even as it preserves the user interface of the application.

Figure 3. Cloud Architecture and Services for Application



To enable migration of an enterprise application into the cloud, the cloud infrastructure needs to expose services to the applications that traditionally were deployed in the data center independent of the application server. These services include network services (Layer 2-7) and non-container application infrastructure services such as naming, directory, description, messaging, etc. For example, the cloud infrastructure needs to offer a firewall service that an application developer could incorporate at design time and the cloud administrator or cloud user could provision at run time. How will the developer know which service to incorporate at design time? The cloud infrastructure will need to be service-oriented architecture (SOA)-enabled, meaning that when a service is deployed in the cloud infrastructure, the service registers with a directory and the directory is available at “well-known” locations inside a cloud. All cloud-enabled applications will find, bind, and query the registry for the service and configure their run time based on its availability. Another example of service would be VIP- or DNS-based TCP load balancing. A web application queries the cloud infrastructure for, say, a load-balancing service. The directory gives the address (URL) of the service to the application and the application uses the URL to create a public VIP - without human intervention.

To generalize - what is missing in the cloud infrastructure is the mechanism and policy that enables an application developer to incorporate functions into the application at design time that at run time query, discover, select, and bind to a service that is substantially similar to the one available on an enterprise physical network. In addition,

what is missing is the mechanism and policy to enable a cloud administrator to set the policy for provisioning, management, and setting QoS of the service in substantially the same manner as the administrator does today.

What is clear from these two examples follows:

- Separation of concerns between a network and compute infrastructure undermines the cloud model.
- The language container that hosts the business logic needs to play the role of an operating system and offer a container-dependent interface to these new cloud infrastructure services.
- There needs to be a hierarchy of controllers in the cloud data center that enforces policies at various levels of the cloud stack. As a collective, these controllers are responsible for the SLA fulfillment in the cloud.

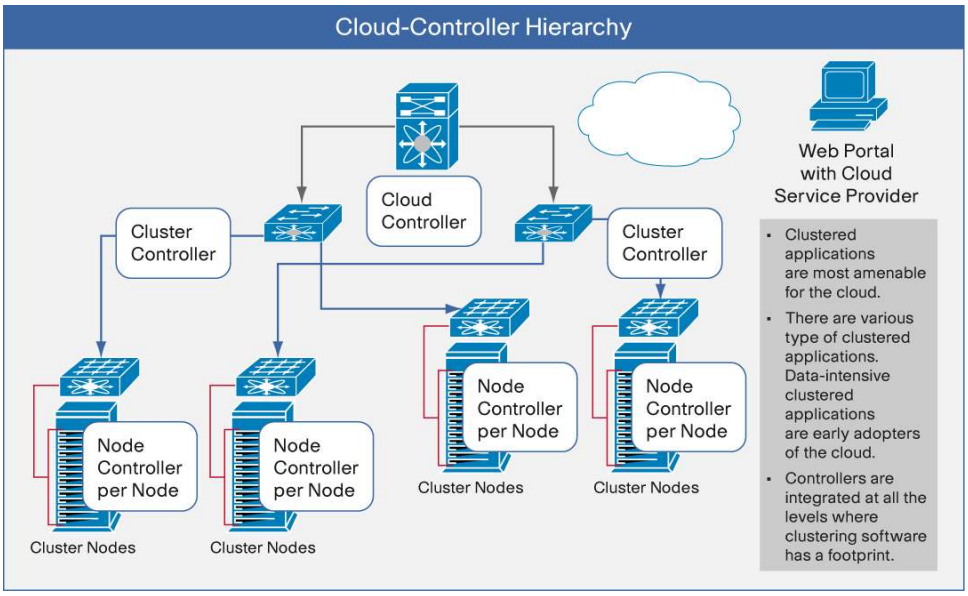
Regulating access to a resource - which itself could be a function exposed as service - is the job of a controller. An avenue for further research is to ascertain if this controller is a monolith or distributed. There are advantages to having a centralized controller, but it may not scale to the Internet scale that is a basic requirement for a cloud data center. A distributed controller with well-defined peering interfaces might be the best implementation path for the near future. Whatever the architecture of the controller, it should be independent of any programming language run time.

The controller - whether distributed or consolidated - would have to offer a single view of the cloud data center control plane. Figure 4 illustrates the various control points in a cloud data center, where the controller needs an agent or has a part distributed. A hierarchy of controllers that communicate over an open protocol and advertise the capabilities of the services for which they are responsible, starting at the lowest level of a node in a cluster to the highest level of a cloud, will enable a single view of the control plane. The best way to describe the controller hierarchy is to describe its critical functions by controller type (Table 4).

Table 4. Controller Hierarchy and Its Function

Controller	Function
Node controller	Hypervisor interface
Cluster controller	Work in progress (WIP)
Cloud controller	API to internal MOS

Figure 4. Typical Cloud-Controller Hierarchy and Their Locations on Network



Another way to describe the controller is to identify what it does not do. The controller is **not** in line and as such does not perform translations of addresses, names, protocols, or languages. It serves more a an information bus that facilitates communication between multiple devices, an integration point for a data center management system, or as a managed Layer 4-7 service object server.

Cloud Service Models

Network computing has three service models: batch computing, code mobility, and remote computing. In batch computing, program and data reside on the client machine and are sent to the infrastructure for computation; the results of the computation are sent back. In code mobility the program resides on the infrastructure and is sent to the client to operate on local data. In remote computing, the program and data reside on the infrastructure and the client gets the results of the computing.

Cloud computing is a new business model that also uses network computing. Not surprisingly, the industry has defined three service models: infrastructure as a service (IaaS; batch computing), platform as a service (PaaS; code mobility), and software as a service (SaaS; remote computing). These service models appeal to some applications more than others. IaaS appeals to large data-computing applications oriented toward science, engineering, and analytics. PaaS appeals to thin-client computing applications such as those oriented toward desktop-productivity computing. Finally, SaaS appeals to easily deployed applications oriented to business processes and workflows. Table 5 lists the three service models and the main stakeholders.

Table 5. Cloud Service Models, Target Applications, and Main Stakeholders

Type of Cloud	Main Pain Point Addressed	Target Applications	Thought Leaders
IaaS	• Bidirectional scalability (elasticity), availability, and manageability	• HPC applications • Engineering applications	• AWS and GoGrid
PaaS	• Interoperability for an application across OSs	• Productivity applications	• Google and Force.com
SaaS	• Anytime, anywhere access	• Workflow and BPM applications	• Salesforce.com

	• Managing license cost		

Each of these cloud service models has distinct requirements from a Layer 4-7 service perspective. The IaaS service model is the least disruptive to SLB evolution. In an IaaS, the SLB is offered to the end customer as a service and charge is by throughput (MBps). It is the PaaS and SaaS service models that are disruptive to SLB evolution.

To remain relevant as a mandatory building block for data centers as they adopt the cloud service and delivery models, current deficiencies of SLB need to be overcome in future generations of the product category. These deficiencies fall into five main categories:

- Lack of true network (Layer 1-3) virtualization
- Lack of support for newer high-availability models
- Lack of integration with data center resource-management framework
- Lack of semantic understanding beyond a flow
- Lack of footprint in the server (real or virtual) (leading to inability to offer service to east-west traffic)

The next section discusses these shortcomings of an SLB from the perspective of the three cloud service models.

Shortcomings of SLB for Cloud Data Center

SLB does not truly virtualize the underlying network. Today one can pass LAN and storage area network (SAN) traffic through a single physical interface (Layer 1) and the hypervisor allows a virtual MAC and host-bus-adapter (HBA) addresses to be associated with the physical interface. SLB has not kept up with these advancements in network virtualization, and it continues to offer its services over an Ethernet-only interface. Not having a role in the traffic management of SAN limits the usefulness of a SLB to TCP connection load balancing. Technologies and standards are being developed that extend a network-interface-card (NIC) interface from confines of the physical server closer to the switch port. An SLB can take advantage of these advances to offer better traffic management to both LAN and SAN traffic into and out of a server. Creating a point of indirection using the VIP artifact means that SLB continues to propagate the hierarchical network that is being designed out of a cloud data center network. Cloud data center networks are increasingly adopting flat topologies, which increase the number of nonroutable IP addresses in a data center. However, the SLB in its standard deployment cannot load balance to a nonroutable IP address. It is for this reason that Stateful Network Address Translation (SNAT) with Dynamic Source Routing (DSR) is being explored as the deployment of choice in data centers.

A first step in virtualization of the network is to implement concurrency in the switch fabric, which enables isolation for higher layers. One could argue that VLAN can be considered as an isolation mechanism and that multiple VLANs are allowed on a switch port. However, lack of concurrency inside a switch fabric relegates the VLAN mechanism to access isolation only; that is, there is no way to regulate the bandwidth consumed by a VLAN. What is needed is technology that enables an external control plane to slice the switch fabric and determine the priorities of the packets in virtual input queues and virtual output queues. Offering an API to this mechanism will enable Layer 4-7 devices - including servers - to implement QoS that supports a customer SLA. In other words, the control plane of a switch would itself run on top of a hypervisor that supports multiple virtual control planes. Each control plane programs its Forwarding Information Base (FIB) into the line card. Incoming packets are classified not only by their tuples but also by the virtual switch for which they are destined.

SLB does not play a role in the newer high-availability models in a data center. The bar for high availability in a cloud data center is set much higher than for a standard data center. The focus is on high availability from the

perspective of the user of a service. If a server goes down, it is expected to be transparent to the end user and not require a re-login. At the transport level, this reality means live migration of the TCP session from one server to another. Applying this standard to high availability for an SLB requires that SLB too migrate a live connection from one SLB to another.

SLB lacks integration mechanisms, preventing it from integrating with the data center manager. Cloud has a management that is centered on "self-service" for the customer and automation for the administrator. To fit into this management model, the SLB has to offer its control plane as a web service that an external entity can instantiate (virtualize) and use. Every mechanism inside the SLB needs to be converted into an object and offered as a managed object. Today's SLB lacks severely in this dimension. The embedded software of an SLB has remained untouched by the wave of creating objects for control and data that swept the application space in the 1990s and 2000s. This lack of integration functions threatens to design-out the SLB from a cloud data center architecture.

SLB has no mechanism to extract intelligence beyond flow. SLB is an optimized packet processor with connection management bookkeeping and limited content intelligence. For it to be a "must-have" in the cloud data center, it needs to develop a mechanism to extract intelligence beyond the knowledge of a flow. The most sought-after intelligence in the proxy infrastructure is the identity of a user and facilitation of the propagation of that identity to various requesting services.

SLB does not have a footprint on the server. SLB does not have a footprint in the networking stack or integration with the hypervisor on the server side. Lack of a footprint is the main contributor to the limited intelligence available in the SLB. Distribution of the SLB across all the stakeholders in a transaction will enable the SLB to make intelligent forwarding decisions.

Anecdotal quotes on SLB shortcomings in a cloud data center follow:

- SLB does not virtualize the network (from an application perspective):
  - There is no virtual session, so failover causes the user to log in if a server fails.
  - There is no independent data-plane bookkeeping; that is, there is no hypervisor that schedules a flow on the data plane; connection tables are not exposed over the API for the application to manage.
  - SLB does not participate in I/O Virtualization (IOV).
  - SLB cannot migrate a TCP session; failover is disruptive.
- SLB does not integrate with CDNs; it does not participate in content refresh or content prepositioning.
- Network insertion creates static paths.
- Inline insertion constrains cluster servers to a Layer 2 segment.
- SLB does not support clustering.
- SLB does not have active-active inline mode.
- The role of SLB is diminished when a response is returned directly to the client (dominant model in the cloud).
- SLB does not understand semantics beyond a flow boundary.
- SLB does not expose an interface for integration with the cloud-controller hierarchy.
- Latency overhead causes server CPU underuse (reason behind disaggregation).
- SLB cannot correlate two sessions in an application context; for example, Web 2.0.

- SLB was originally developed for a short request-response traffic pattern.
- SLB was originally developed for insertion into north-south traffic. It has a limited role in east-west traffic, and does not offer any traffic management or engineering for east-west traffic (dominant traffic pattern in the cloud).
- SLB offers one-ended optimization; it lacks a host (or guest) footprint to offer two-ended optimization services.
- SLB has no resource reservation mechanisms (because latency overhead causes server CPU underuse).
- SLB is not aware of user identity (and therefore no has role in identity propagation).
- SLB supports the Cloud Application Access Profile (the session is treated equally if accessed over public network, private network, or mobile network).
- SLB has no role in SLA enforcement.
- SLB does not integrate with the hypervisor.
- SLB has a significant overlap with the virtualization infrastructure manager (high-availability function, scaling function, and health-check function are usurped by virtualization).

The New Role of Layer 4 -7 Services in Application-Delivery Infrastructure in the Cloud

The evolution of Layer 4-7 services in a data center faces several cross-currents:

- Changes in data center traffic pattern
- Expected changes in physical network architecture
- Increasing traffic volume on the virtual network

Changing Traffic Patterns

The cloud application exhibits a traffic pattern that is different from a typical production data center. Only 20 percent of the traffic in a cloud is of the type in-out (north-south), and most of the traffic is intra-cloud (east-west). In other words, 80 percent of the packets switched inside a data center remain inside a data center, yet the SLB is designed for the 20 percent of the traffic that enters and leaves the data center. Analyzing the data center traffic by duration reveals that short-duration traffic is deeper in the data center server tiers, whereas the long-duration traffic is at the edge of the data center. However, today's SLB was designed to handle a short-duration mostly read traffic pattern. Further, the type of traffic observed at the application server (80 percent) is mostly synchronous read/write of very short duration (1 msec), whereas the type of traffic observed at the web server (20 percent) is mostly long-lived with intermittent transfer. The median flow size for in/out traffic is 100 MB, and the average number of concurrent flows observed on a server in a cluster is 10.

In summary, the kinds of traffic patterns that SLB was designed for are no longer found at the edge of a data center but deep in its server tiers. Today, the SLB does not proxy traffic deeper in the server tiers, keeping the SLB from offering advanced Layer 4-7 services. The evolution of the SLB and the Layer 4-7 services that it offers will seek to plug this discrepancy between design center and the use case.

Expected Changes in Physical Network Architecture

The data center network has traditionally been designed as an inverted tree with core, aggregation, and distribution (access) tiers. Higher tiers have excess switching bandwidth, whereas the lower tiers are oversubscribed. This scenario worked well for a traditional three-tier data center where the ratio of the traffic within a tier to intra-tier was less than 1. In other words, the bisectional traffic in a three-tier data center is lower than

in/out traffic. In a cloud data center, however, the traffic bottleneck is not at the edge of a data center but at its bisection (middle). Several efforts are underway in industry to fundamentally change the data center network architecture away from the inverted tree to a HPC style fat tree. A fat-tree data center network increases the bisectional bandwidth of the data center by allocating non-blocking links to east-west traffic and blocking links to north-south traffic. Essentially, it removes the explicit traffic engineering that is implemented in today's network design, thereby enabling the network to support any traffic matrix.

SLB deployment in a one- or two-arm mode at the core of the inverted tree keeps it from offering its services to the traffic in the bisection of a cloud. In other words, most of the traffic does not pass through an SLB in a cloud data center. The next-generation SLB will need to offer its services to the traffic that traverses the bisection of a data center network. Opportunity exists for the SLB in a cloud to lower the cost of the data center network by efficiently redirecting east-west traffic - thereby requiring fewer non-blocking links. One way to achieve this requirement is to insert a shim layer into the east-west traffic, which redirects relevant traffic to an external SLB. This method is similar to inline insertion (albeit virtually) in the east-west data path. The shim layer runs on the virtual machine that is hosting the application being serviced by SLB or on the NIC of the server hardware.

Increasing Traffic Volume on the Virtual Network

The virtual-network traffic is switched by a soft switch that is found in the hypervisor of a virtual infrastructure. This traffic does not leave the physical confines of a server and is not visible to an outside SLB. For virtual environments, a better way would be to intercept the traffic transparently by adding functions to the hypervisor. The hypervisor is always in the data path for all east-west and north-south traffic within a virtual server. Distributing the SLB functions into a transparent proxy for north-south traffic and intercepting the Layer 7 hypervisor in line with SmartNIC offload will not only enable SLB to offer its services to east-west traffic but also lower the end-to-end latency for the web transaction. Figure 5 illustrates the distribution of functions among all stakeholders in an end-to-end web transaction.

Figure 5. Distribution of SLB Functions Among Elements Involved in Web Transaction

Another way to look at the distribution of functions is to classify the functions into a transparent mode function vs. an intercepting function. To lower the latency, the SLB has to limit the number of times it intercepts a flow. Migrating all transparent functions (per packet operations) into a upstream switch and terminating functions in to

the hypervisor (with acceleration functions in the NIC to bypass the hypervisor) reduces the overall cost of the SLB infrastructure for the cloud service provider and improves end-to-end latency for the end user.

Table 6 lists the most frequently deployed SLB functions, their classifications, and their new locations.

Table 6. Layer 4-7 Services in Cloud and Role and Location of SLB and ADC

Layer 4-7 Services in Cloud	Classification	SLB and Analog Digital Converter (ADC) Role	Distribution of the Functions in Cloud
Hardware VIP and other redirection	Per packet	Terminate first-time inbound connection and select session manager	SLB and ADC devices as proxies in DMZ
Secure Sockets Layer (SSL) acceleration	Per flow	First terminate or redirect HTTP to Secure HTTP (HTTPS) on session manager	SLB and ADC devices as proxies in DMZ
Authentication	Per flow	Redirect to authentication agent, which is mostly on session manager but can also be part of SLB infrastructure	Authenticating agent outside the DMZ and not in the same VLAN as servers or VIP
Distributed DoS (DDoS)	Per flow		
URL Rewrite	Per flow		
Compression and encryption	Per packet		
Caching	Per flow		
Persistence (session, file extension, etc.)	Per flow		
Global server load balancing (GSLB)	Per flow		
Content prepositioning	Per flow		
Firewall (web)	Per flow		
Firewall (Layer 2)	Per packet		

What is clear from Table 7 is that SLB functions need to be distributed among the critical elements involved in a web transaction: browser, proxy, server network stack, and server application stack. The approach to insertion of SLB functions into the elements will determine the future of Layer 4-7 services.

Table 7. Elasticity Requirements for SLB by Cloud Service Model

Cloud Service Model	Critical Resource That Needs to Be Shared and Role of Load Balancer	Load-Balancing Function for Critical Resource
IaaS	• Virtual-machine host is the critical resource and cost to the IaaS provider • Load balancer is a service in itself that is offered as part of the package	• Load balancer should have a virtual infrastructure footprint • Its provisioning is managed by cloud provisioning system • Load balancer should support synchronous and stateful interface for configuration and polling • Load balancer should have mechanism to interpret cloud provisioning systems policies
PaaS	• Business logic of the application is the shared resource. Shared over an API, it is often referred to as multi-tenancy • The API calls to representational state transfer (REST) or Simple Object Access Protocol SOAP interfaces would need a Layer 7 load balancer	• Load balancer should have Layer 7 load balancer that proxies the authentication system and propagates identity • Load balancer should proxy for virtual services

SaaS	• Access to the service is the critical resource being shared • This role is the traditional one played by a load balancer today	• Load balancer should offer higher throughput and higher SSL TPS rates

Elasticity in Load Balancing

Management of the cloud has evolved from application of optimal configuration to automated provisioning of resources based on an SLA. This increased sophistication of the cloud management system (CMS) requires that a resource register itself with the provisioning system and be provisionable by the CMS. In other words, any resource that is not registered with the CMS is outside the cloud fabric and therefore not really cloud-enabled. This change in the cloud data center requires that all Layer 4-7 services be accessible over an API. Any discussion on the future of load balancing in a data center necessitates an estimation of the growth (a bet) on these clouds personalities.

From the application load-balancing point of view, the SLB will have to offer an integration stack that integrates its control plane with the nearest cloud controller. Today the SLB has a closed control plane and offers its services over an Ethernet interface. Today we map a URL request to a page on a specific server. In a cloud data center, we have to map a URL request to a controller responsible for infrastructure where this URL resides.

Figure 6 shows the application workloads that are gaining traction in a public cloud. It is important to note that there is rapid change in the affinity of an application to the cloud. In a span of 6 months, the remote storage application lost its dominant cloud application status to web applications. Another point to note is that clouds are increasingly being accepted for mission-critical revenue-generating applications. The early adopters of cloud were scientific computing and disaster recovery, but lately, they are revenue-generating web front ends for brick-and-mortar companies. The final point to note is that the leading applications consume primarily the PaaS and SaaS service models. The early adopters such as batch computing and remote storage consumed primarily the IaaS service model.

Figure 6. Public Cloud Workloads Study; Source: The 451 Group

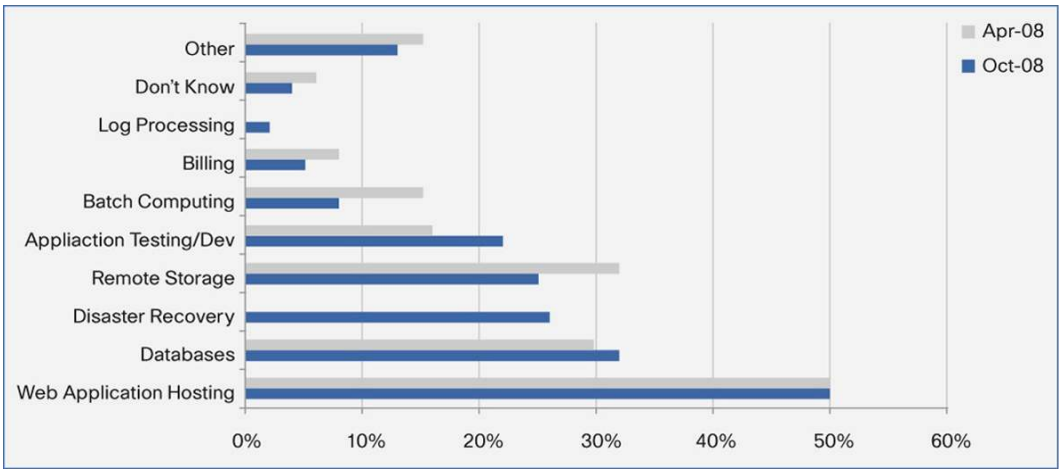


Table 8 lists the elements of an application-delivery infrastructure and their elasticity requirements.

Table 8. Elasticity Requirements of Application-Delivery Infrastructure

Elements of Application-Delivery Infrastructure	Elasticity Requirements for Cloud Computing
Location	• Independent of data center • Integration with public clouds for spillover and surge protection • Integration at compute and storage levels; for example, content could be prepositioned
Name space	• Path resolution based on rule matches
Transport	• Multiple data paths controlled by a single control plane • Transport profiles do not assume flat bandwidth; existence of low-latency paths • Transport-level protocol translation • Support for intermittent clients
Session	• Single session per user with properties including security policies, application ACLs, and host affinity; ACLs include time of day (or cloud front) • Transparent failover of sessions
Customization	• Support for user customization on a per-user basis; it is not cost-effective to change the core application
Application logic	• Adaptive clustering

Focusing on delivery services in Table 1, the first scalability limitation that we experience is the number of data paths that are controlled by a single control path in an SLB. In other words, in today's SLB, the policy definition point and policy enforcement point have a one-to-one relationship and are packaged as a single appliance. Addition of new services to this package entails adding higher-layer services to the control plane and packet and flow-level services to the data plane.

Elasticity is a big feature of the cloud - about as big as its pay-per-use pricing model. It is the ability to expand and contract capacity based on policy.

In the delivery services layer, the functions that need to be expanded or contracted all exist in the data plane. Table 3 lists the functions of the delivery services layer that need to be "elasticized".

The second scalability limitation is evident from Table 2. Today's data-plane functions are based on static maps that are cached in the data path. These maps are configured on the device control plane and have zero to no integration with external directory systems. To bring elasticity into the data plane, the maps would have to be created on a per-user basis.

Conclusion

Data centers are evolving to adopt a cloud service model. Whether the deployment model is public or private, the challenges that face application architectures are the same. To overcome these challenges, infrastructure used by the applications needs to evolve to support the cloud service models. Of the three models - PaaS, IaaS, and SaaS - IaaS and PaaS are expected to guide the evolution of Layer 4-7 services. The critical dimension of evolution for Layer 4-7 services embodied in an SLB is the programmatic dimension. SLB will have to expose all of its functions as managed objects to an external cloud manager to enable scaling of applications in the cloud computing data center.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA C11-675038-00 06/11