Realtime publishers

The Definitive Guide to Cloud Acceleration

Dan Sullivan

sponsored by

CDNetworks

Copyright Statement

Chapter 5: Architecture of Clouds and Content Delivery

Although cloud computing has been widely adopted across a range of organizations, this technology might not meet all the key requirements of enterprise applications. The previous chapter of this guide examines how the architecture of the Internet affects application performance, especially on a global scale. This chapter turns the focus to the architecture of public cloud services and considers how public cloud providers meet some, but typically not all, enterprise requirements. The chapter considers this issue from three perspectives:

- Public cloud providers and virtualized IT infrastructure
- Application designers and their responsibility for application architecture
- Content delivery networks as complementary to public cloud provider services

Designing and deploying enterprise applications on a global scale requires attention to infrastructure, architecture, and network optimization. As this chapter highlights, public cloud providers are well positioned to address infrastructure and related service issues. Content delivery networks address the network optimization problem. Combining public cloud services with content delivery services in a well-designed application architecture can provide a solid foundation for building enterprise applications.

Although the IT industry sometimes refers to public cloud providers in general, it is important to understand that there are different types of public cloud services and deployment models. Let's begin a discussion of public cloud providers with a review of distinguishing characteristics.

Public Cloud Providers and Virtualized IT Infrastructure

The common characteristic of all cloud providers is a service model based around on-demand access to shared, typically virtualized, computing and storage resources. Cloud providers distinguish themselves in the way they virtualize IT infrastructure. You can see these differences in varying service and deployment models. Service models vary according to the types of infrastructure and application services offered, while the deployment models vary according to the types of users with access to the provider's resources.

Before diving into a detailed description of service and deployment models, let's first define essential characteristics of all clouds.

Essential Characteristics of Cloud Computing

Public cloud providers offer relatively easy access to computing and storage services. The US National Institute of Standards and Technology (NIST) has identified five essential characteristics of cloud computing:

- On-demand service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

(For full details on the definition, see "The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology." NIST Special Publication 800-145.)

Network performance is outside the scope of the cloud computing definition, but it is still an important element of overall application performance. Already in this discussion, it is becoming apparent that cloud computing alone does not address all the requirements of globally deployed, enterprise applications. Latency is a dominant problem in Web application performance. Cloud applications run in data centers that might be a long distance from end users, which introduces distance-induced latencies. There are other potential performance-degrading problems as well. For example, peering agreements between ISPs may be structured in ways that lead to degraded performance when data is routed over networks controlled by other ISPs.

On-Demand Service

Cloud users have the ability to provision and use virtualized resources when they are needed for as long as they are needed. Specialized systems administration skills are not required. Users work with a cloud computing dashboard to select the types and number of virtualized servers needed. For example, if a group of analysts needs a server to analyze a large data set, they would log into a dashboard or control panel, select an appropriate type of virtual server, identify the machine image with the necessary operating system (OS) and analysis software, and launch the virtual machine.

It is important to emphasize the self-service nature of this process. Virtualization has long been used in data centers to improve the efficiency of server utilization, but prior to cloud computing, setting up a virtualized server required significant knowledge about OSs, hypervisors, and system configurations. Cloud computing platforms automate many of the steps required to instantiate a virtual machine.

Broad Network Access

Cloud computing allows for access to services over standard network and application protocols. This approach decouples virtual servers and storage resources from the client devices that access them. This decoupling is especially important when applications have to support multiple client devices, such as mobile devices, laptops, servers, and mainframes.

In addition to the ability to reach applications, broad network access enables access to data stored persistently in the cloud. Data access may be mediated through an application or through a standard Web services method such as REST that allows users to store, manipulate, or delete data using URL-based commands.

Resource Pooling

Cloud computing customers share infrastructure. When a user starts a virtual machine, it is instantiated on a physical server in one of the cloud provider's data centers. The customer may be able to choose the region or data center in which the virtual machine runs but does not choose the physical server itself. In all likelihood, the virtual machines running on a single server belong to several customers. Similarly, one customer's blocks of storage in a storage array may be intermingled with other customers' data.

The cloud computing platform and hypervisors are responsible for isolating resources so that they are accessible only to resource owners and others explicitly granted access to the resources. With secure resource pooling, cloud providers can optimize the efficiency of server utilization by pooling the resource requirements of large numbers of customers.

Rapid Elasticity

In a cloud, it is relatively easy to instantiate a large number of virtual servers. For example, the group of analysts working on a large data set may determine that the best way to analyze the data is to use a cluster of servers working in parallel. Launching 20 servers is not much more difficult than launching one. (Coordinating the work of the 20 servers is a more complex problem, but there are applications for managing distributed workloads that can be readily deployed in a cloud.)

Cloud providers also offer services to monitor loads on servers and bring additional servers online as needed. For example, if there is a spike in demand for a Web application, the cloud management platform can detect the increased demand, bring additional servers online, and add them to a load-balanced group of servers. Those servers can be shut down automatically when demand drops.

The ability to rapidly add servers or storage can help address some aspects of peak demand, but this type of rapid elasticity does not address network-related performance issues. Consider the following simple example.

A Web application running in a cloud data center in the eastern United States is experiencing higher than usual demand from users in Europe and Asia. The load-monitoring system detects an increase in workload and brings an additional virtual server online. The set of application servers can now process a larger number of transactions in a given amount of time. This setup does not, however, affect the time required to transmit data between the servers and the client devices. The latency of the network between the data center in the US and the client devices in Europe and Asia is not altered by changes in the application server cluster running in the cloud.

Rapid elasticity is clearly an important part of optimizing application performance, but it is not sufficient to address all performance issues.

Measured Service

Cloud providers use a "pay as you go" or "pay for what you use" cost model. This setup fits well with the rapid elasticity and pooled resource aspects of cloud computing. Customers do not have sole use of dedicated hardware, so it is important to charge according to the share of resources used. Cloud providers typically use tiered pricing based on the size or quality of a service. For example, a high-memory virtual machine will cost more than a low-memory virtual machine. Similarly, charges for higher-performance solid state drives will be higher than those for commodity disk storage.

As you can see from this description, the essential characteristics of cloud computing are insufficient to address the full range of cloud optimization requirements. There are different deployment and service models, and it is worth considering how these may impact cloud optimization issues.

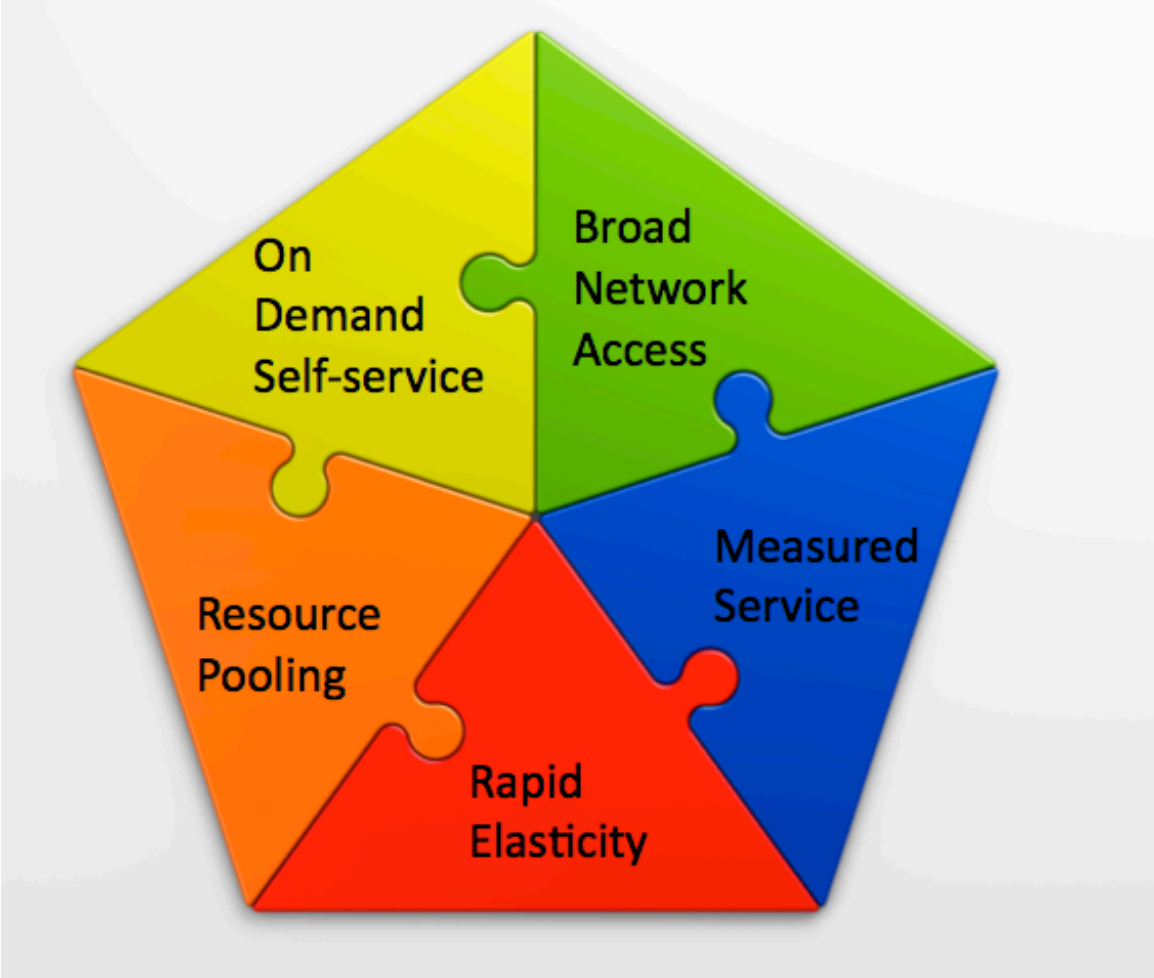

Figure 5.1: The essential characteristics of cloud computing address many areas relevant to application performance, but key considerations, such as latency and packet loss, are outside the scope of cloud computing fundamentals.

Cloud Computing Deployment Models

You can distinguish cloud services by the types of users who are granted access to the services. Four deployment models are:

- Public
- Private
- Hybrid
- Community

Public clouds are open for use to the general public. Generally, anyone with a credit card and access to the Internet can make use of public cloud resources. Public clouds are the least restrictive of the four deployment models with regards to who is granted access.

Private clouds are at the other end of the access spectrum. This type is the one of the most restrictive cloud deployment models. Access to a private cloud is restricted to members of a single organization, such as a business or government entity.

Hybrid clouds are clouds with two or more clouds that are linked in such a way to allow for portability of data and applications between the two clouds. This definition allows for different combinations of cloud types (for example, two private clouds, a private and community cloud, and so on) but a private cloud and a public cloud is most typical.

Community clouds are designed to serve a group of users from multiple organizations who share common requirements. For example, a community cloud could provide specialized HIPAA-compliant services to healthcare providers. Only members of the specialized community are granted access to these clouds.

Deployment models are important considerations for those concerned with security and compliance issues. These models do not directly affect application performance because the same cloud infrastructure and management platforms can be deployed in any of these models.

Deployment models may indirectly affect performance in cases in which you want to leverage the benefits of content delivery services or network optimization services. Consider an example of a hybrid cloud based on one private cloud and one public cloud. The public cloud may offer a proprietary content delivery network that works with content stored in the public cloud. Content maintained in the private cloud may have to be replicated to the public cloud before it can be served through the content delivery network.

In addition to considering deployment models, it is important to consider how different service models may impact overall application performance and our ability to optimize application and network services.

Cloud Service Models

Service models of cloud computing differ according to the amount of control customers have over virtualized resources and the level of service provided. The three types of cloud service models are:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Some vendors employ even finer-grained classification schemes with terms such as Databases as a Services (DaaS) and Analytics as a Service (AaaS); for this chapter's purposes, we will not delve into those specialized areas. The goal here it to understand how various cloud service models can impact application performance and optimization. The IaaS, PaaS, and SaaS models are sufficient for the needs of this discussion.

Infrastructure as a Service

IaaS models allow users the greatest level of control over the provisioned infrastructure. IaaS customers, for example, can choose the size of the virtual server they run, the OS, software libraries and applications, as well as various types of storage. Users control the OS, so they have substantial control over the application platform and system configuration. The owner of a provisioned server could:

- Grant administrator access to others if needed
- Install software applications
- Create user accounts for others to log into the server
- Change access control privileges on files stored locally
- Configure local firewalls and other security measures

IaaS clouds are a good fit when users have specialized requirements and the need for control over OS and application stack software. If you need to run a legacy, custom application in the cloud, the IaaS model is probably the best option.

IaaS clouds provide a service catalog consisting of machine images that can be run in the cloud. These can include images with a variety of minimally configured OSs (for example, a variety of Linux and Windows Server versions) or machine images that include OSs as well as parts of an application stack, such as relational databases, application servers, and specialized applications, such as search and indexing systems.

Although IaaS provides the greatest degree of control, it also requires the most systems management expertise. For example, you might need a combination of OSs, patch levels, and software libraries that is not available in the service catalog. In this case, you would have to start with a base image and install the additional patches and components you need. If you plan to use this image for extended periods of time, you can save the image but you will have to maintain the image going forward. This maintenance includes patching and updating the OS as needed.

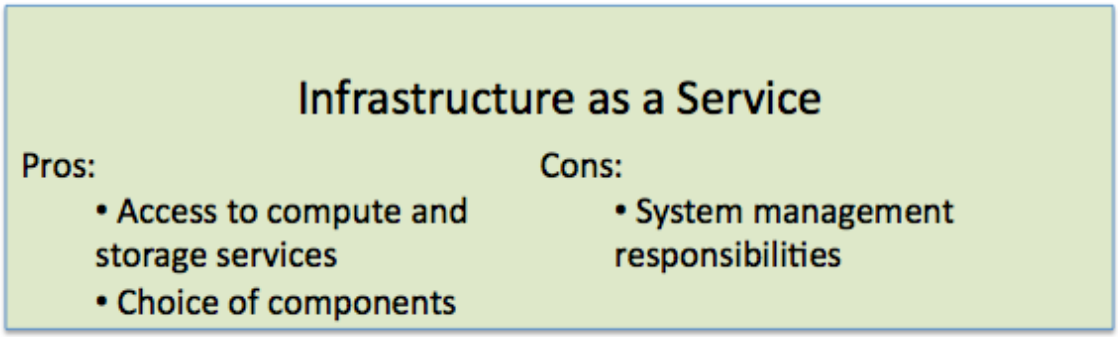

Figure 5.2: IaaS allows for high levels of control but entails high levels of responsibility on the part of cloud users.

In addition to performing systems administration operations, you might need to perform database administration and other types of application management when using IaaS. Again, there is a tradeoff between flexibility and responsibility. In an IaaS setup, you have a choice of relational database management systems, search and indexing applications, and application servers. The disadvantage is that you are then responsible for properly installing, configuring, and maintaining these components.

In some situations, developers and application designers do not need the level of control offered by IaaS. In those cases, a PaaS cloud may be a better option.

Platform as a Service

In a PaaS cloud, customers do not have substantial control over virtual servers, OSs, or application stacks. Instead, the PaaS provider supports programming languages, libraries and application services used by developers to create programs that run on the PaaS platform.

PaaS providers offer different types of services. Some specialize in a single language or language family, such as Java and languages that run on the Java virtual machine (JVM). Others tend to be language agnostic but offer a variety of frameworks and data stores that can be combined to meet the particular needs of each customer.

An advantage of PaaS over IaaS is that the PaaS cloud provider is responsible for managing more components in the application stack and infrastructure. As with IaaS clouds, the PaaS cloud provider manages underlying hardware and network infrastructure on behalf of customers. PaaS providers manage additional software components as well.

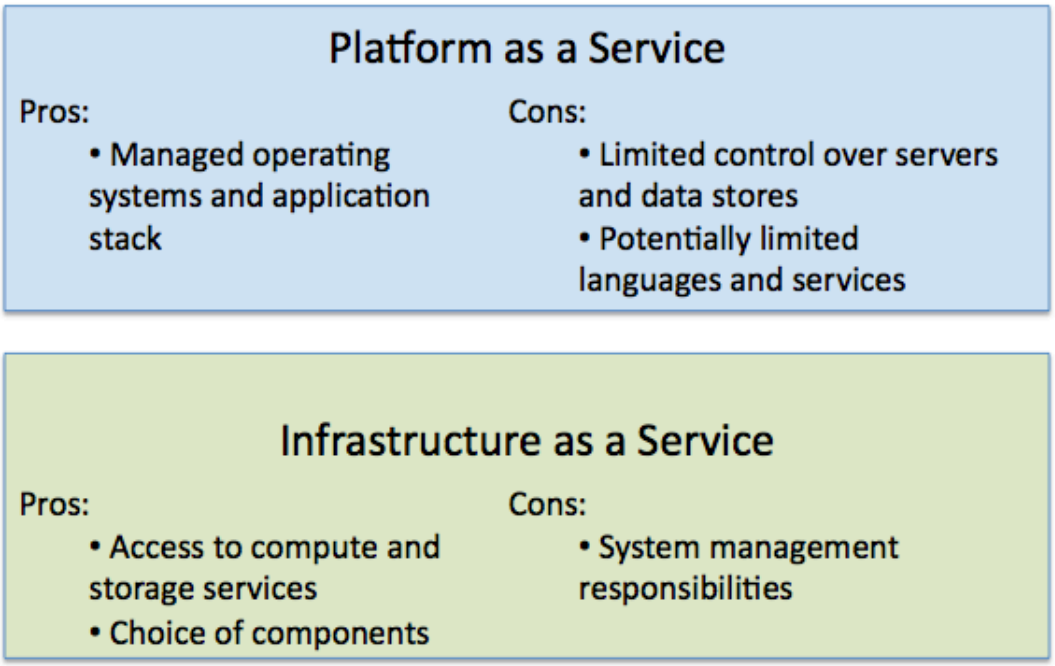

Figure 5.3: PaaS models build on the same kind of infrastructure provided by an IaaS setup but alleviate some of the management responsibility of an IaaS.

The tradeoff for developers is that they are more constrained in their choices in a PaaS. For example, a PaaS cloud provider may implement a message queue service for its cloud. If developers need a message queue, they will have access to the PaaS provider's chosen tool. Developers working an IaaS cloud could choose from a number of message queue systems and manage it themselves.

Extending this model of increasing levels of service and decreasing levels of control brings us to the SaaS model.

Software as a Service

SaaS providers are the most specialized of cloud providers. Rather than focus on providing access to virtualized servers and storage or offering developers a managed application stack, SaaS providers offer full applications. Common SaaS use cases include:

- Human resources (HR) management
- Customer relationship management (CRM)
- Financial management
- Project management
- Time tracking
- Lead generation

As the list implies, a wide variety of back-office functions are available through SaaS providers. SaaS providers offer turnkey solutions. Developers do not have to design data models or build user interfaces as they would in a PaaS. They also do not assume the systems management tasks associated with using IaaS clouds.

The disadvantage of SaaS is that customers have the least control over the service. For example, a SaaS customer cannot generally dictate the type of data store used to persistently store application data. The SaaS provider makes such design decisions and all customers use a common application platform.

Since SaaS providers have control over the underlying architecture of the system, it is important for customers to understand how the SaaS provider's design choices affect data integration, data protection, and other aspects of application management. For example, data from a SaaS financial management system may be needed in an independent management reporting system. Users may be able to perform bulk exports or query smaller subsets of data through an application programming interface (API). Some customers may need to maintain their own backups of data from the SaaS application. In this case, the customer will need to define and implement an export process in order to maintain up-to-date copies of data on-premise. Customers would also need to understand the data model used by SaaS to extract data for use in other applications.

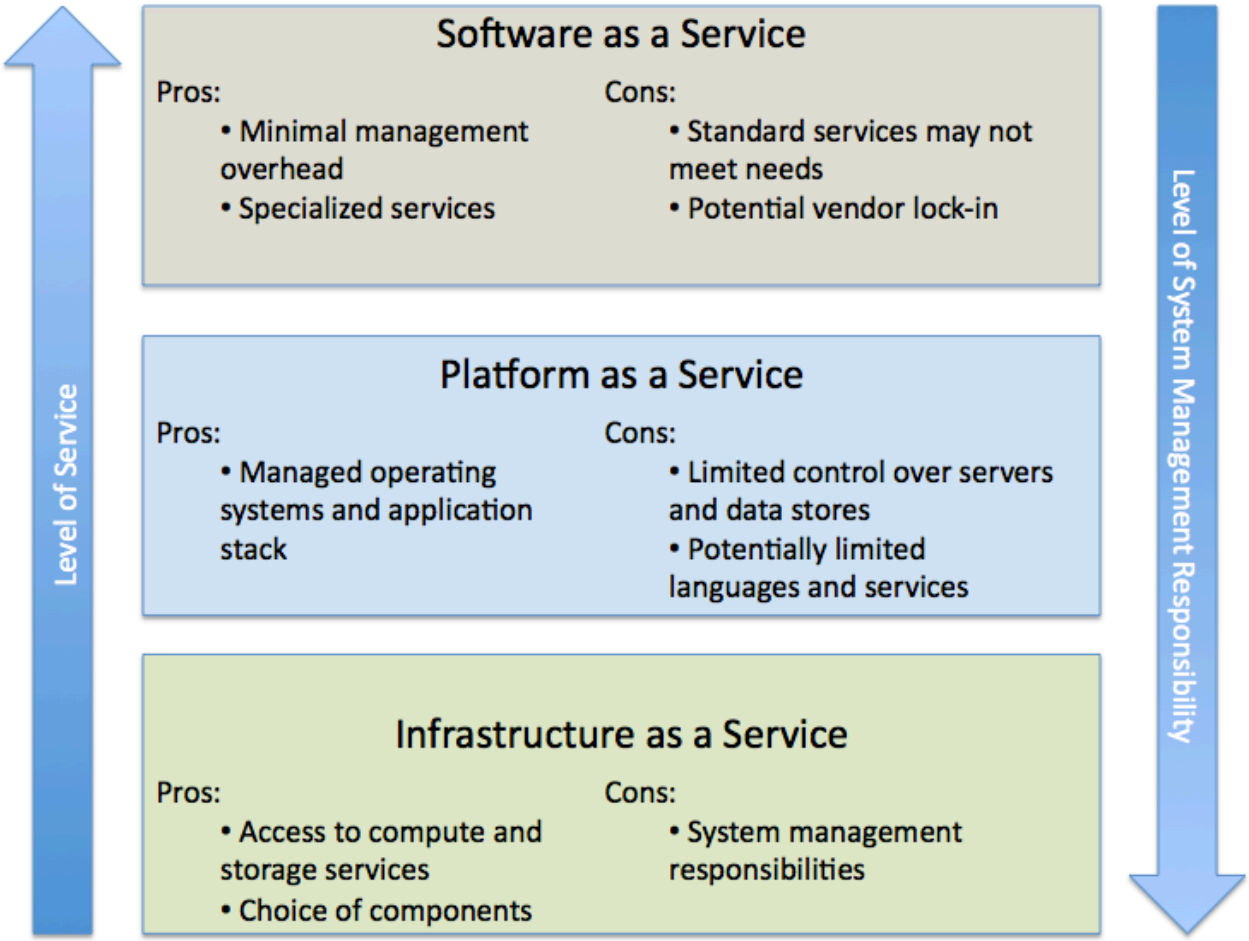

Figure 5.4: SaaS provides turn-key services with the lowest levels of systems management responsibility but limited ability to customize the implementation of the service.

The essential characteristics of cloud computing along with the deployment models and service models give a broad, structured view of cloud computing services. From this perspective, you can see that cloud providers virtualize, some but not all, the key components of enterprise-scale applications. In particular, cloud service providers virtualize:

- Servers
- Storage
- Platform and software services

The servers may be substantially under the control of cloud users, as in the case of an IaaS cloud; abstracted to logical computing units, as in some PaaS providers; or essentially hidden from users, as is the case with SaaS.

Storage is also virtualized. IaaS cloud providers offer several types of persistent storage, including object storage, file system storage, and database storage systems.

Services, such as message passing, search over unstructured data, and specialized data services are also virtualized. As one moves into PaaS and SaaS service models, higher-level services are provided.

This movement from lower-level infrastructure management to turn-key system services may give the impression that these various cloud models provide a comprehensive platform for deploying enterprise-scale applications. Such is not always the case. Application designers still have multiple areas of life cycle management and performance management to consider when deploying such applications.

Application Design and Application Architecture

Application designers and architects use cloud infrastructure and services to deliver business services to customers, employees, collaborators, and other stakeholders. IaaS and PaaS clouds provide building blocks upon which to design and deploy custom applications. SaaS providers offer turnkey solutions that can either solve a standalone problem or fit into a larger workflow that incorporates SaaS applications with other systems. This situation leaves many choices for software designers and architects. Rather than try to examine all design issues, this section will consider four representative issues that can help offer a sense of challenges that designers and architects face. This discussion will examine issues relevant to IaaS and PaaS users; software designers and architects developing SaaS applications have to contend with these issues, but SaaS customers do not.

The four issues considered here are:

- Server failover
- Application server replication
- Content caching
- Network optimization

Cloud providers offer solutions, or building blocks for solutions for some of these but as we will see, there is no single approach that will work for all plausible use cases.

Designing for Server Failover

Highly reliable systems are designed to continue functioning in the event of a failure in one or more of the components. Failover techniques are applied at virtually all levels of design. Physical servers may have multiple power supplies to ensure the server continues to function even if one of the power supplies fails. Storage devices use multiple disks in various RAID configurations to mitigate the risk of data loss due to a hardware failure in the disk drive. Moving up in systems complexity from individual servers to clusters of servers, you can see additional mechanisms for promoting resiliency.

A server-level failure could take an application offline unless there is a mechanism in place to shift the workload from the failed server to a functioning server running the same application. One simple way to do this is to deploy a cluster of servers running the same software and route traffic to those servers through a load balancer. The load balancer distributes the load across servers in the cluster. If the load balancer detects a failure in one of the servers (for example, the server does not respond to a ping request), the load balancer can route traffic to other servers in the cluster until the failed server is back online.

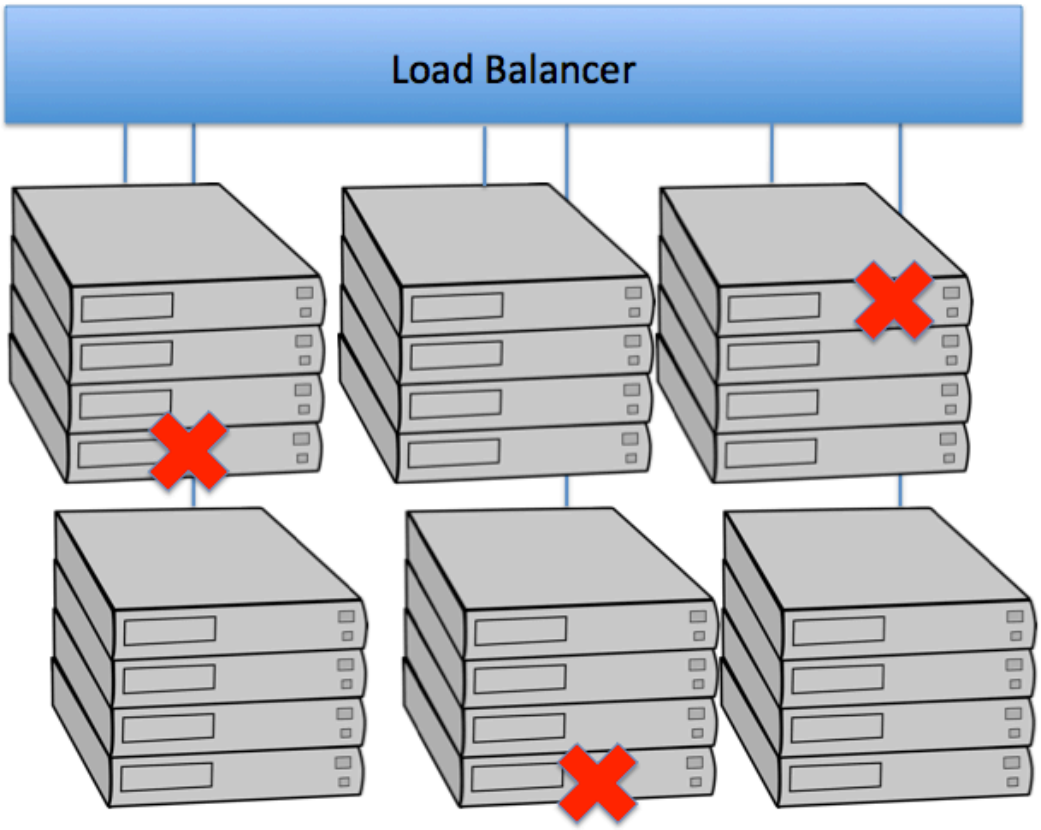

Figure 5.5: A load balancer provides a simple but often effective form of failover in a cluster of servers.

Application Server Replication

Enterprise application architects might have to consider failures at higher levels of organization than even clusters. For example, if a natural disaster, systematic software error, or other catastrophic failure prevents access to a data center or slows network traffic to unacceptable levels, server-level redundancy and cluster-level failover strategies will be insufficient to address this risk.

Enterprise applications may require redundancy across data centers. If application servers within a single data center are unavailable, traffic to those servers can be routed to an alternative data center. This situation is not ideal, of course. Depending on the distance, peering arrangements between ISPs, and other network configuration issues, there may be longer latency for users working with a distant data center. The additional workload on the application servers in the redundant data center might also slow application response time. Adding more servers to the application server cluster may help alleviate some of this problem, but there may be a limited number of servers available to add to the cluster. Such might be especially true if a large data center is experiencing a failure and multiple enterprise applications are shifting workloads to the redundant data center.

Figure 5.6: Multiple data centers can provide failover recovery in the event of a catastrophic failure at the data center level (Image source: CDNetworks).

The first thoughts about failover recovery may be focused on application servers and ensuring that backup servers are available. These items are a necessary part of failover recovery but are not the only crucial components.

In addition to application servers, data must be accessible to the failover servers. When a single server fails in a cluster, the other servers will still have access to data on storage arrays in the data center. To ensure the ability to failover between data centers, you must make sure data is replicated between the data centers.

One of the considerations in designing a failover strategy is determining how frequently to synchronize data between the data centers. More frequent updates reduce the risk of losing data, but the cost is additional, perhaps significant, traffic between data centers. You can extend the window of time between synchronization operations to reduce network traffic. The maximum amount of potential data loss is the amount of data that is created or modified within that window of time. In the worst-case scenario, a data center failure would occur just prior to a synchronization operation. Network optimization techniques, such as those described in earlier chapters, can help reduce the time and traffic required to replicate data between data centers.

Replication is an important element of failover recovery but another type of replication, content caching, is an important part of many application designs.

Content Caching

Application designers can take advantage of the properties of static content to improve application performance. Static content is any type of data that can be generated and stored for use at some time in the future. This type encompasses content ranging from Web pages to data files that rarely change. Static content changes infrequently, so there is minimal risk to maintaining multiple copies.

Caching is an efficient strategy for a number of reasons. By keeping a copy of static content in multiple locations, users can receive data from the closest location and therefore typically reduce latency. Static data is stored in a local cache after the first time it is accessed.

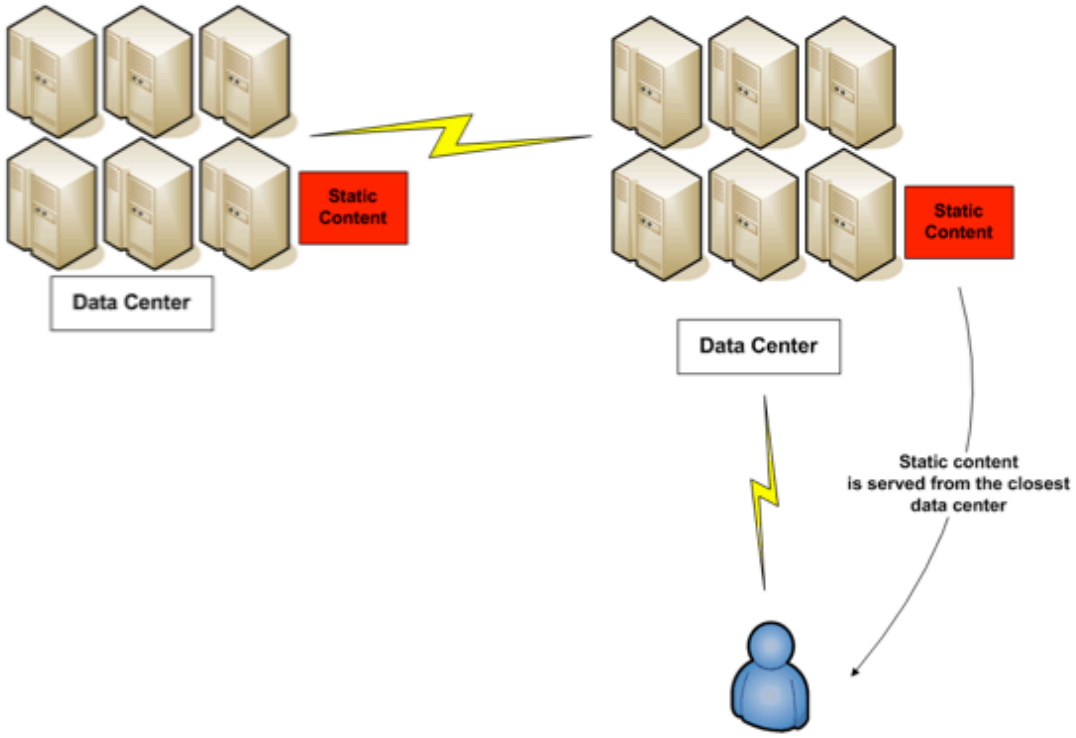

Figure 5.7: Static content rarely changes, so once it is retrieved from a distant server, it can be cached for future use by other users of the closer data center.

Application designers do not have to devise a metric to determine the content most likely to be requested. Instead, only the content that has been requested is cached. Consider a multi-language Web site. The number of users requesting content in Finnish is likely to be low outside of northern Europe. The data center serving that region will likely have cached copies of that content while data centers in other parts of the global probably will not.

The amount of memory dedicated to caching will determine the upper bound on the amount of static data that can be a cached at one time. An effective way to work within this limit is to track the last request time for each content object. Older objects that have not been requested are good candidates for removing from the cache without adversely impacting performance. News stories or major announcements, for example, may be popular for a time but eventually are surpassed in popularity by other newer stories or announcements.

Removing the least-used content is just one strategy for managing caches. One could consider the size of an object and develop a weighting scheme that favors retaining smaller objects. The idea here is that removing a single large object would allow multiple smaller objects to be stored. Other factors, such as the number of times an object is requested over time, can also be factored into the cache management algorithm.

As useful as caching is, it does not address the need to improve network performance when transmitting dynamic content.

Network Optimization

Application architects have to consider many aspects of application performance and reliability. Network optimization is especially important for enterprise and globally used applications.

Consider a hypothetical scenario: An organization is deploying a new financial market monitoring application to the cloud. The application will have users in North America, Europe, and Asia. The application collects near–real-time data from institutions across three continents. Up-to-date information is vital to the users of this application, so cached data will not meet requirements. A commodity trader in Chicago, for example, might want the latest information on commodity prices in Hong Kong. A cached data set that is 2 hours old is essentially useless to the trader in Chicago. Or, it could be worse than useless. Making a decision based on out-of-date information could lead to costly transactions that could have been avoided with timely data.

As data must move between global data centers on an as-needed basis, network optimizations are key considerations. Network optimizations can include:

- TCP parameter optimization
- Reducing overhead associated with retransmitting dropped packets
- Increasing data window sizes to reduce overhead communications

These optimizations operate at the implementation level of TCP. Other strategies can help as well. Compression can be used to reduce the total payload size, although there is the additional cost of compressing the data at its source and decompressing it at the destination.

In addition, organizations or their network providers may negotiate better quality of service (QoS) guarantees. When working with the Internet on a global scale, you need to consider how peering agreements between ISPs will impact performance. Providers with high-quality peering agreements can offer customers better performance in some areas than those who depend on lower-capacity, slower ISPs.

Application architects consider many factors ranging from server reliability and failover to content caching and network optimizations. As more organizations adopt cloud computing, infrastructures can help to consider how content delivery networks can complement cloud service providers.

Content Delivery Networks Complement Cloud Providers

Cloud service providers offer an array of infrastructure, software, and services for use in enterprise applications. As this chapter has discussed, these address many but not all the needs of high-demand Web applications. Fortunately for systems architects and application designers, content deliver networks can complement the services provided by cloud vendors in several key areas:

- Distributed caching
- Network protocol optimization
- Application delivery network services
- Secure content delivery
- Support for content life cycle management

Distributed caching as noted earlier helps to improve the performance of applications serving static content while network protocol optimizations help with dynamic content applications. Managing distributed applications is a complex and demanding process. Cloud providers have some basic controls and platforms to support distributed application management, but content delivery networks that also include application delivery services can offer additional services.

Discussions about enterprise computing and public clouds often include concerns about security and compliance. Here again, content delivery network providers can complement and supplement the services of public cloud providers with enterprise support for secure content delivery and support for the full life cycle of content management.

The next chapter will examine issues related to choosing a content network delivery service. As with other IT projects, making use of public clouds, content delivery services, and application delivery services requires planning and careful attention to integration issues.