the Availability Digest

www.availabilitydigest.com
@availabilitydig

Reliability and Availability of Cloud Computing

February 2015

The robustness of clouds, especially public clouds, is a topic of continuing concern today. There are significant economies available to companies for hosting their applications in clouds. However, the reliability of cloud computing continues to be spotty, and most companies are still reluctant to move their critical applications to the cloud.

In their book "Reliability and Availability of Cloud Computing" (John Wiley & Sons, 2012), authors Eric Bauer and Randee Adams analyze the factors contributing to cloud downtime and make recommendations for steps to achieve uptimes of 99.999% (five 9s), a level of availability that they consider appropriate for mission-critical applications.

As the authors state, the book provides an "analysis of reliability and availability risks and architectural opportunities [to] offer guidance on how to develop cloud-based solutions that meet or exceed service reliability and availability requirements of traditional systems."

Characteristics of Cloud Computing

The U.S. National Institute of Standards (NIST) defines cloud computing as:

> "… a model for enabling ubiquitous, convenient, on-demand, network access to shared pools of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

The key words here are "on-demand access to pools of resources." The authors point out that "access" means anywhere there is access to an IP network. "Resource pooling" implies that clouds are multitenant and that resource pools are accessed simultaneously by multiple applications resident in the cloud. "On-demand" means that applications can request resources from pools and return them to pools at the applications option. The ability for an application to control the amount of resources that it uses (and gets charged for) is called "elasticity."

Thus, the essential characteristics of cloud computing are

- on-demand self service
- broad network access
- resource pooling
- rapid elasticity
- measured services (pay-as-you-go)

Cloud Implementations

A cloud is a collection of hardware "host" servers, each running a *hypervisor*. A hypervisor allows multiple guest operating systems, each with their own applications, to run on the host server. So far as each guest operating system is concerned, it is the only entity running on the host server. A guest operating system with its applications is known as a *virtual machine* (VM). It is the job of the hypervisor to allocate the compute, memory, storage, networking, and service resources of the host server to the VMs according to their needs and to effectively isolate the VMs from each other.

There are several models for cloud architectures:

- Bare metal virtualization – The hypervisor resides directly on the host server and allocates host-server resources to the VMs directly.
- Full virtualization – A host operating system runs on the host server. Requests for services are made by the VMs to the host operating system, which is responsible for allocating resources. The guest operating systems can be different from the host operating system.
- OS virtualization – The same as full virtualization, but the guest operating systems must be the same as the host operating system.
- Paravirtualization – The guest operating systems are modified to contain their own hardware device drivers so that they can access external storage and network resources directly.

VMs can be moved from one host server to another for load balancing or for recovery following a host server failure. If *live migration* is provided, a VM can be moved with little if any impact on the application.

Many clouds provide georedundancy by implementing multiple geographically distributed data centers, each its own self-contained cloud. Cloud bursting is often supported in which an application (or parts of an application) can be moved to a different cloud for load-balancing or disaster-recovery purposes.

Cloud Reliability and Availability

The authors define three measures of quality for cloud computing that have to be measured and controlled:

- *Availability* is the percent of time that the cloud is available (its uptime). It is often measured as a number of 9s – 99.99% uptime is four 9s.
- *Reliability* is the percentage of proper responses that the cloud provides. It is often measured as defects per million (DPM). For instance, 100 responses out of one million may be in error. This is a DPM of 100. Reliability can also be characterized by 9s. A DPM of 100 means that 99.99% of all responses will be correct, a reliability of four 9s.
- *Latency* is the delay imposed by the cloud in returning data to the client. Latency could be the time from a request to a response, or it could be a gap in streaming data. If the latency is too long, this may be interpreted as a downtime event or as a defect. Thus, latency affects both availability and reliability.

In terms of application acceptability, the authors suggest that two 9s is sufficient for routing applications. Essential applications should be provided four 9s of availability and reliability, critical applications five 9s, and safety critical applications seven 9s.

Throughout the book, the authors suggest many criteria that should be included in a Service Level Agreement (SLA). They generally focus on the needs of critical applications needing availability and reliability of five 9s (an average of five minutes of downtime per year, or 10 DPM).

Designing for Resiliency

The key to high availability and high reliability is resiliency. Should a VM fail, the time that it is down counts against availability and the requests that it cannot process (or the streaming that it cannot keep up with) count against reliability.

There are several factors that can contribute to degradations in availability and reliability, including:

- hardware
- software
- power
- environment (e.g., cooling)
- network
- payload (messages or data streams)
- human controls
- policies for dealing with problems
- configuration data
- disaster-recovery plans

The authors analyze each of these vulnerabilities. They point out that virtualized systems are more complex than traditional systems because of the existence of the hypervisor and the risk that VMs are not truly isolated from each other. In their words, the purpose of the book is to show:

> "How can one assure that the benefits of cloud computing are achieved without diminishing service reliability and service availability to levels below those achieved by traditional application deployment methods?"

Host Server Reliability

Host servers in a virtualized environment may not be as reliable as those in a traditional environment. One of the economies that is achieved by virtualization is the more efficient use of the data center's servers. Rather than running one application that may keep a server only 10% occupied, the server can be running several VMs that keep it 80% occupied.

This has several negative impacts on host server availability:

- Increased hardware utilization reduces the time that components can engage power management mechanisms to reduce thermal stress.
- Increased thermal stress due to elevated ambient temperatures reduces the lifetime of the hardware.
- Hardware suppliers may reduce derating rules and design margins to reduce costs.
- Increased duty cycles on hard disk drives and other components will reduce their lifetimes.

VM Mobility

The ability to migrate VMs from one host server to another provides two primary advantages. The failure of a host server can be rapidly recovered by migrating the VMs on the failed server to other surviving

servers. Also, the load on servers can be balanced easily by moving VMs from heavily loaded servers to those with more available capacity.

Moving a VM is conceptually straightforward. A VM is actually a file holding its operating system and application software executables. VM files reside on storage devices that are accessible to all host servers. To move a VM, all that is required is to stop the VM on its current host server and to have its new host server load the executables so that the VM will run on that server.

If live migration is provided for load balancing, the new VM is brought up while the old VM is still running. The current application state (i.e., the application's volatile data) is moved to the new VM as the application is still running on the old VM. When most of the application state has been moved, the old VM is paused until the final application state has been fully migrated; and the new VM is then activated. The new VM continues processing where the old VM left off. Pausing for final state transfer can be measured in the sub-second range, so that users often will not even realize that there has been a pause.

VM mobility also allows zero-downtime upgrades to guest operating systems and applications. A new VM can be brought up and upgraded as necessary. The old VM can then be shut down and the new VM energized to continue application processing with the upgraded software. If live migration is used, users may not even be aware of the upgrade.

Latency

In cloud computing, there are additional opportunities to induce delays, or latency, into processing. Latency compared to a traditional approach can be aggravated by several factors. The host servers are typically running at a much heavier load, thus slowing down their response time. The failure of a host will induce delays as VMs are migrated to surviving hosts.

Excess latency can be perceived as downtime or as response defects to the users of the applications.

Redundancy

As with any redundant system, it is essential to avoid single-point-of-failures. Care should be taken to avoid multiple copies of the same application from running on one host system or several critical applications from running on a single host system. If that host system fails, all applications running on it fail.

With the ability to move VMs from one host to another, virtualization inherently incorporates redundancy to allow fast recovery from the failure of a host system. But what of the failure of an entire data center due to a force majeure such as an earthquake or flood?

Many public clouds provide multiple data centers that are geographically distributed. This is called georedundancy. By replicating VM executables to the mass storage of several clouds, VMs in a failed data center can be recovered by bringing them up in another data center.

Cloud Bursting

In some cases, workload can be moved from the cloud of one cloud provider to that of another for load balancing. The Open Virtualization Format (OVF) is a common standard for the format of a VM executable. To the extent that two different clouds use OVF, VMs can be moved from one cloud to another and continue in operation.

Other Architectures

There are many other architectures that are available for cloud-computing users. For instance, a company may turn to a cloud to provide compute resources, but elect to maintain its data in its own data

center. Architectures such as this add additional complexity to the cloud and can adversely affect its availability and reliability characteristics.

Availability and Reliability Measurements

There are several points of failure in a cloud system – more than in a traditional system. These points of failure include:

- end users
- networks
- application software
- the cloud service provider
- the cloud consumer (the user of the cloud services)

It is important to continually measure the effectiveness of the cloud in delivering the required availability and reliability. The authors define four measurement points (MPs) that should be monitored:

- MP1 – component instance (servers, storage, firewalls, load balancers, etc.).
- MP2 – primary data center.
- MP3 - aggregate service performance across a pool of data centers, excluding networks.
- MP4 – end-to-end service level – MP3 including networks.

Cloud Elasticity

Elasticity is a fundamental requirement for cloud computing. With elasticity, an application can request additional resources (growth) and request that resources be released (degrowth). Thus, the user is billed only for the resources that he uses.

Growth/degrowth requests can either be automatic or manual. With automatic requests, the application is aware of its resource needs (for instance, perhaps it is monitoring the rate of transactions that it must process) and submits the requests. With manual requests, the user requests changes in resource allocation. This may be, for instance, in anticipation of much heavier traffic during certain time periods. In either case, sufficient capacity should always be available so that growth/degrowth decisions can be made in the order of every fifteen minutes, not seconds.

Traditional systems are designed with a maximum overload capacity. If the offered load is below the design load, the system is reliable and available. It the offered load is at the maximum load capability, the system may be available but not reliable due to latency. If the offered load exceeds the maximum design load, the system is neither available nor reliable. Virtualized systems avoid this problem via elasticity.

Elasticity may be achieved in several ways, depending upon the nature of the elastic request:

- Vertical growth – more resources are allocated to the VM.
- Horizontal growth – more VMs are created.
- Outgrowth – some of the application load is moved to another data center.

There are several scenarios that can cause an elasticity failure. These scenarios must be thoroughly tested to ensure that they are reliable and include:

- Growth of application capacity
- Degrowth of application capacity
- Growth of persistent storage
- Degrowth of persistent storage
- Overload conditions (require rapid elasticity)

Mapping Applications to VMs

There are several considerations for mapping application software to VMs:

- Will a failure group be active/standby or N+K?
- If growth of one component always necessitates the growth of another, then the components should be in the same VM.
- If two components share resources or have tight latency requirements, they should be in the same VM.
- If the application is made up of components that don't share any resources, they should be in separate VMs.
- VMs that frequently communicate with each other should be on the same hypervisor to reduce latency.
- VMs that represent a single-point-of-failure should be on separate hypervisors.
- Though VMs should be hardware agnostic, some VMs may perform better on some host servers than on others.

Service Management

Service management entails linking together architectural tasks and tools necessary to initiate a service and to automatically manage a service. Services include:

- on-demand self-service
- resource pool management
- service monitoring
- service load distribution
- service provisioning

An effective service management framework should include:

- Mechanisms that collect and monitor measurement data against thresholds.
- A policy management system that defines rules, conditions, and actions to be taken.
- A cloud management system that performs, manages, and reports on the actions dictated by the policy management system.
- Automation of the service orchestration framework.

Summary

A cloud computing environment should optimize the reliability and availability criteria required by the user:

- Accessibility – continuously available to users with no planned downtime.
- Retainability – maintain sessions across failures.
- Quality and Reliability – correct application results, streaming without latency delays.
- High Availability – auto-detection, isolation, and recovery.
- Moderate operating expense – rapid growth and degrowth.

The cloud must be designed for failure so that it can rapidly recover from any fault.

The contents of the book are highly redundant. Each chapter includes the concepts that are important to the topics covered in that chapter. Therefore, the reader can choose any chapter that seems important to him and read it with full understanding of all of the underlying concepts.

About the Authors

Eric Bauer is reliability engineering manager in the Software, Solutions, and Services Group of Alcatel-Lucent. Randee Adams is a consulting member of the technical staff in the Software, Solutions, and Services Group of Alcatel-Lucent.