Cloud Reliability

Yury Izrailevsky
Consultant

Charlie Bell
Amazon

Cloud computing allows engineers to rapidly develop complex systems and deploy them continuously, at global scale. This can create unique reliability risks. Cloud infrastructure providers are constantly developing new ideas and incorporating them into their products and services in order to increase the reliability of their platforms. Cloud application developers can also employ a number of architectural techniques and operational best practices to further boost the availability of their systems.

Rapid adoption of cloud computing over the past decade left virtually no industry untouched: from video streaming (Netflix, Amazon Video) and gaming (Riot Games) to banking (Capital One), healthcare (Cerner) and services delivered by various government agencies (NASA, CIA, and the CDC). With more mission critical services running in the cloud, the expectations for cloud reliability are higher than ever. In this introduction, we will identify some of the common design principles and operational best practices that are being successfully employed to increase cloud service reliability, and illustrate them with examples.

RELIABILITY VS. COMPLEXITY AND THE RATE OF CHANGE

Cloud computing offers significant benefits in terms of scalability and performance. Elasticity of the cloud gives engineers the ability to rapidly deploy vast amounts of compute and storage resources. A global network of cloud regions enables low latency and interactive access to cloud-based services for customers around the world. At the same time, the vast array of edge locations brings content even closer through its delivery networks. Developer productivity is also significantly accelerated in the cloud. API-driven infrastructure and platform services, from core networking and storage components to advanced AI capabilities such as automatic speech-to-text (Amazon Transcribe) and image recognition (Google Cloud Vision API), enable engineers to integrate complex functionality into their applications with minimal development effort. Hosted continuous integration/continuous deployment (CI/CD) services and tools, such as Jenkins and Spinnaker, drastically reduce the amount of time it takes to push changes into production.

Engineers can use all of these components and services, cloud "building blocks," to develop complex systems and deploy them globally, faster than before. An online service may consist of hundreds of microservices, each running in its own distributed cluster and containing multiple dependencies of its own, increasing the number of individual components that can fail. Rapid

Copublished by the IEEE CS and IEEE ComSoc
2325-6095/18/$33.00 USD ©2018 IEEE

innovation is enabled by frequent changes in the code, configuration, and data. Yet each change also carries the risk of triggering a failure, potentially resulting in a customer facing outage.

Fault tolerant design principles and operational best practices can mitigate these reliability risks. Used properly, they can enable engineers to build "self-healing" systems that can mitigate failures without requiring human intervention and no-to-minimal impact on the users of these systems.

ARCHITECTING FOR FAILURE

On December 24, 2012, Netflix experienced an 18-hour outage affecting the majority of its customers.[1] It resulted from a regional failure of the AWS Elastic Load Balancer (ELB) service in the US-East-1 cloud region, where Netflix had all of its cloud operations at that time. Subsequently, AWS made significant improvements to its networking control plane, ELB service architecture and configuration management process.[2] Also learning from this outage, Netflix moved to an Active-Active cross regional architecture, where customer traffic is load balanced across three cloud regions, and data is kept in an eventually consistent sync using Cassandra (a distributed, cross-regional NoSQL database) replication.[3] In case of an outage, traffic from the affected region can be redirected to the remaining healthy regions, as Figure 1 shows.

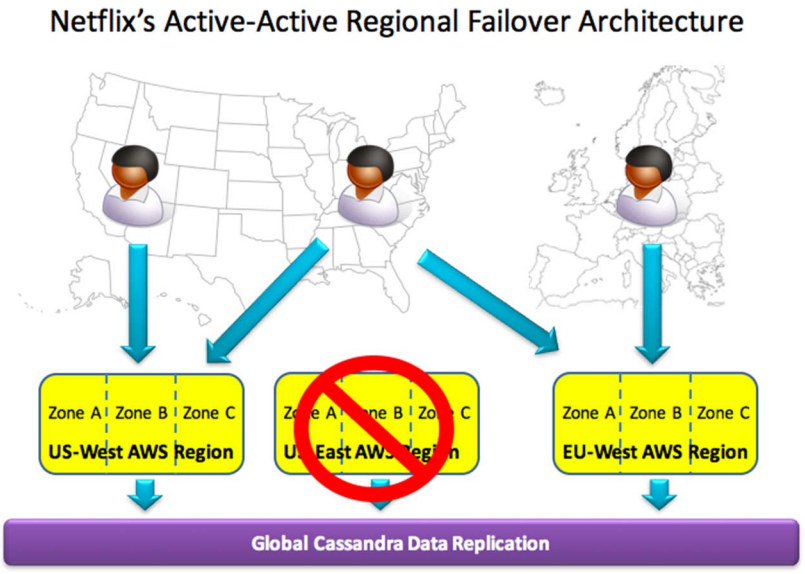

Figure 1. In the Netflix active-active cross regional architecture, customer traffic is load balanced across three cloud regions, and data is kept in an eventually consistent sync using Cassandra (a distributed, cross-regional NoSQL database) replication. In case of an outage, traffic from the affected region can be redirected to the remaining healthy regions.

Cloud infrastructure providers continue to invest significant resources toward increasing the resiliency of their services, both at the hardware and software levels. AWS added DynamoDB global tables and multi-master Aurora SQL engines, Azure is previewing multi-master support in Cosmos, and Google developed Spanner, a global quasi-relational database service that uses atomic clocks to guarantee strong consistency.[4] A paper featured in this special edition, "Virtual Machine Resiliency Management System for the Cloud," describes two complementary mechanism boosting virtual machine resiliency in IBM enterprise cloud managed service.

It is the goal of every cloud infrastructure platform to provide highly reliable components and services, but occasional failures are inevitable in any complex distributed system. While it has been demonstrated that the overall availability of a system is determined by the sum of its critical components,[5] it is possible to reduce the criticality of infrastructure dependencies by embracing

fault tolerant architectural principles, such as *redundancy* and *isolation.* Cloud application developers can employ well established design patterns, such as circuit breaker[6] and bulkhead,[7] to build systems that are resilient to failures of one or several underlying components. In essence, cloud provides infrastructure building blocks, but it is the responsibility of the application developers to stack them up and operate them in a way that maximizes the overall system availability.

Consider Netflix's current cross-regional architecture. Three cloud regions provide redundant capacity: if one of them fails, the other two can continue serving customer traffic from the affected region. In order to avoid global outages, changes must be isolated and deployed to one region at a time. Using libraries such as Hystrix,[8] which implements circuit breaker and bulkhead design patterns, allows to isolate and contain the "blast radius" of the failures of sub-components (such as individual microservices). From the cloud-provider side, AWS cloud regions utilize separate, independent installations of services, such as ELB, S3 (object storage), SQS (queueing), Auto-Scaling, and EBS (block storage). These regions also obey an independent deployment rigor, including immediate escalation to top leadership if an engineer creates a deployment pipeline with the capability of making simultaneous changes in multiple regions. When an outage hit S3 on February 28th, 2017, the issue was confined to a single region, allowing Netflix to continue operations with virtually no impact on its customers.

OPERATIONAL BEST PRACTICES AND CHAOS ENGINEERING

Any experienced DevOps engineer will tell you that there is no such a thing as a "safe" code deployment or configuration update: any change you make in production carries the risk of an outage. To minimize such risks, two techniques have been found to be particularly effective: blue-green deployments and canary releases.

With blue-green deployments, you set up two parallel production environments: one running the old version of the code (blue) and one running the new release (green), each capable of handling the full production workload. As traffic is routed from blue to green, the blue cluster remains fully operational, and if any problems are encountered with the new release, traffic is immediately rerouted back to the blue cluster. Once there is enough confidence in the new code, the blue cluster is spun down. This is a very powerful technique, which is enabled by the elasticity the cloud, where for a short period of time (e.g., a few hours) you effectively double your production footprint, and release the extra capacity when it is no longer needed.

Similarly, with canary releases, you set up a parallel "canary" environment running a new version of the code, but you only route a small subset of customers to it (1% is typical) while collecting performance and operational analytics, such as CPU/memory usage and error rates. This enables engineers to detect operational problems with the new release without exposing it to the full set of customers, and generally requires fewer additional cloud resources. Kayenta is an automated canary analysis tool integrated within the Spinnaker continuous delivery platform, which pulls user-defined metrics, runs statistical tests and generates an aggregate score for each canary release.[9] Typically, one runs canary analysis first, and if successful, proceeds to the full deployment using the blue-green mechanism.

To ensure the overall resiliency of the system, it is critical to have good visibility into the state of the system at all times in order to detect a failure quickly, and ways to mitigate the failure once it occurs. Tools such as CloudWatch[10] and Stackdriver[11] provided by cloud infrastructure vendors, and open source solutions such as Atlas,[12] enable engineers to instrument operational metrics in a scalable way, and trigger alerts based on any number of possible failure scenarios.

Incorporating fault-tolerant architectural principles and following deployment and monitoring best practices can significantly reduce the likelihood of production failures, but will not eliminate them completely. A live production environment, containing complex infrastructure and application dependencies and constantly stressed by ever-changing usage patterns (such as spikes in traffic or faulty network conditions), will expose a number of bottlenecks and failure scenarios over time that were never anticipated in the design, development, testing, and deployment phases. To identify and address such "ticking time bombs" proactively, a new discipline called Chaos

Engineering has emerged.[13] Chaos Engineering is focused on experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production.

In 2011, Netflix announced its Simian Army,[14] which is a set of operational "Chaos" tools that trigger controlled failures in a live production environment, in order to ensure that the system can detect and automatically recover from such a failure, and if necessary, the engineering team is able to provide a timely and effective response. For example, Chaos Monkey randomly terminates live production instances, testing hardware redundancy. Latency Monkey injects additional latency into the client-server communication layer, testing the system's isolation and graceful degradation capabilities. Chaos Kong evacuates an entire cloud region, testing cross-regional failover capabilities in case of a catastrophic regional outage. These and many other types of tools have been developed over the years, and there is an active open source community developing and sharing Chaos tools.

The Chaos Engineering methodology, originally popularized by Netflix, has been embraced by a number of enterprises. However, many business owners have been reluctant to follow the approach, concerned about the associated costs, such as the impact Chaos production experiments might have on customer experience. In a paper included in this special edition, "The Business Case for Chaos Engineering," a group of Netflix engineers provide quantitative and qualitative reasoning for the necessity and cost-efficiency of the Chaos Engineering approach.

RECENT TRENDS ENHANCING CLOUD RELIABILITY

AI has been driving recent innovations across many different technology areas, and cloud reliability is no exception. Production environment of a modern enterprise, such as that of Amazon or Netflix, can generate over a billion operational metrics every minute;[12] runtime logs, aggregated over time, can scale into petabytes. It is simply impossible for humans to manually process such enormous volumes of data in order to identify small relevant bits, such as unique, non-obvious combinations of metrics that may lead to customer facing outages over time. Instead, supervised and unsupervised deep learning techniques, in combination with statistical methods, can be used to identify and help mitigate failure patterns.

Similar ML-based anomaly detection approaches can be applied to enterprise security (malicious insider behavior or intrusion detection), resource management (predictive auto-scaling of clusters in a cloud environment) and other problems. ML engines such as TensorFlow and MxNet, coupled with powerful GPU-powered hardware, such as NVIDIA Volta architecture and the P instance family from AWS, make deep learning capabilities readily available to a wide variety of cloud customers.

Rapid rise in the level of compute abstraction has been another significant development in the last couple of decades that has revolutionized the way companies manage their infrastructure. At the turn of the century, most production environments were deployed directly onto physical hardware instances, or "bare metal." Adoption of virtualization, moving to the cloud and architecting around container-based frameworks have successively allowed for a much more efficient use of hardware and supported dynamic workloads that can adapt to customer demand, survive hardware failures and reduce vertical scalability constraints of a physical instance. All of these capabilities have brought significant benefits in terms of reliability and operational efficiency, but they still require engineers to provision the underlying compute capacity, whether physical hardware on-premise or virtual instances in the cloud.

The emergence of serverless computing in recent years have propelled architectural and operational capabilities to a completely new level. The term "serverless" refers to a broad range of cloud capabilities and services, enabling developers to build complex systems by accessing prepacked applications and components, and allowing them to deploy custom application logic, while outsourcing most operational concerns such as "resource provisioning, monitoring, maintenance, scalability, and fault-tolerance to the cloud provider."[15] A powerful serverless paradigm called Function-as-a-Service (FaaS) in particular has been gaining rapid momentum. All major cloud vendors feature a FaaS offering, including AWS Lambda, Google Cloud Functions,

Microsoft Azure Functions and IBM OpenWhisk. FaaS enables developers to encapsulate custom logic within stateless ephemeral code blocks, or functions, which can be invoked both synchronously and asynchronously, triggered by a wide range of conditions and events, and whose resource provisioning and execution environment are managed entirely by the cloud provider.

Serverless computing is still in its early days, but cloud vendors are rapidly integrating serverless functionality into their platforms, making it a central part of their offering, and supporting an ever-increasing number of applications' use cases. Cloud customers, free from the constraints of resource provisioning and management, can focus on their applications' business logic, while their systems can operate more efficiently and reliably in a centrally managed, tightly monitored elastic pool of serverless capacity.

CONCLUSION

In this short introduction, we have summarized some of the architectural techniques and operational best practices that can be employed to increase cloud service reliability. Two selected articles show how reliability can be improved at the infrastructure level by boosting VM resiliency to failures, and at the application level by embracing the principles of Chaos Engineering. AI capabilities offer more promise in increasing cloud reliability in the future.

ACKNOWLEDGEMENTS

We would like to acknowledge the support from Dr. Mazin Yousif, the Editor-in-Chief of *IEEE Cloud Computing* magazine, and Brian Brannon, editorial content manager.

REFERENCES

1. A. Cockcroft, "A Closer Look at the Christmas Eve Outage," *Netflix Technology Blog*, blog, Netflix, 30 December 2012; https://medium.com/netflix-techblog/a-closer-look-at-the-christmas-eve-outage-d7b409a529ee.
2. "Summary of the December 24, 2012 Amazon ELB Service Event in the US-East Region," *AWS Service Message*; https://aws.amazon.com/message/680587/.
3. R. Meshenberg, N. Gopalani, and L. Kosewski, "Active-Active for Multi-Regional Resiliency," *Netflix Technology Blog*, blog, Netflix, 12 February 2013; https://medium.com/netflix-techblog/active-active-for-multi-regional-resiliency-c47719f6685b.
4. E. Brewer, "Spanner, TrueTime and the CAP Theorem," Google Research, 2017; https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45855.pdf.
5. B. Treynor et al., "The Calculus of Service Availability," *Communications of the ACM*, vol. 60, no. 9, 2017, pp. 42–47.
6. M. Fowler, "CircuitBreaker," 6 March 2014; https://martinfowler.com/bliki/CircuitBreaker.html.
7. C. Bennage and M. Wasson, "Cloud Design Patterns," Microsoft Azure, 28 November 2017.
8. M. Jacobs, "How it Works," GitHub, 3 July 2017; https://github.com/Netflix/Hystrix/wiki/How-it-Works.
9. N. Kaul, "Introducing Kayenta: An open automated canary analysis tool from Google and Netflix," *Google Cloud Platform Blog*, blog, Google, 10 April 2018; https://cloudplatform.googleblog.com/2018/04/introducing-Kayenta-an-open-automated-canary-analysis-tool-from-Google-and-Netflix.html.
10. "Amazon CloudWatch," Amazon; https://aws.amazon.com/cloudwatch/.
11. "Google Stackdriver," Google Cloud; https://cloud.google.com/stackdriver/.
12. "Introducing Atlas: Netflix's Primary Telemetry Platform," *Netflix Technology Blog*, blog, Netflix, 12 December 2014; https://medium.com/netflix-techblog/introducing-atlas-netflixs-primary-telemetry-platform-bd31f4d8ed9a.
13. C. Rosenthal et al., *Chaos Engineering*, O'Reiley Media, 2017.

14. "The Netflix Simian Army," *Netflix Technology Blog*, blog, 11 July 2011; http://techblog.netflix.com/2011/07/netflix-simian-army.html.
15. I. Baldini et al., "Serverless Computing: Current Trends and Open Problems," *Research Advances in Cloud Computing*, Chaudhary S., Somani G., Buyya R., Springer, 2017.

ABOUT THE AUTHORS

Yury Izrailevsky advises venture capital firms and technology companies to help them optimize their cloud strategy and scale their engineering organizations. He holds BS and MS degrees in Computer Science from the University of Utah. Izrailevsky's research interests include distributed systems, cloud architecture and serverless computing. Contact him at izrailev@gmail.com.

Charlie Bell is Senior Vice President for AWS Products and Engineering. He has held this position since 2006 and served in various other technical leadership roles at Amazon since joining the company in 1998. He holds a BA in Business/MIS from California State University Fullerton. Bell's research interests range from specialized database systems to large-scale distributed systems reliability. Contact him at cbell@amazon.com.