Week 7: Improving Availability and Performance of Cloud Systems

Nate Bachmeier

TIM-7140:Software Engineering

April 11th, 2021

Northcentral University

**Improving Availability and Performance of Cloud Systems**

Applications that move into the cloud gain access to elasticitiy, instaneous provisioning, sophisticated security and cost controls, among other reasons. These new capabilities shift the modern architecture away from monolithic designs toward micro-service systems. While many applications are modernizing their approach to high-availability is not evolving (Verbitski, et al., 2018). This includes several tried-and-true methodologies, such as state checkpointing and fail-over clustering (Zhao, 2014). Fundamentally, these strategies make assumptions regarding the physical hardware constraints. However, the cloud’s virtualization enables bending some of these rules. Engineering teams need to reassess cloud-native patterns as a mechanism to improve their systems’s performance, reliability, and economics.

# Background

Distributed systems are the most complicated computing environments because of their parallel and asynchronous nature. Many implementations also make false assumptions regarding the network’s reliability, security, homogeneousness, latency, bandwidth, and transport costs (van Steen & Tanenbaum, 2019, p. 986). When systems introduce one of those fallacies into the design, it produces subtle defects under production loads. Businesses compat these risks through high-availability architectural patterns that promote self-healing (Yang, Min, Yang, & Li, 2014). These strategies follow combinations of reactive (e.g., heart-beating) and proactive solutions (e.g., rejuvenation tactics).

There are inheritent challenges with every high-availability solution. For instance, state check-pointing requires periodically writing memory to disk. This operation is extremely I/O intensive and significantly degrades performance (Cheng, Huang, & Lee, 2019). Yet, few of those snapshots are used (Wu, Shang, Peng, & Wolter, 2020). Organizations that can remove these performance penalities could reduce resource requires, improve Quality of Service (QoS), and become more competitive through cost reductions.

Another standard approach is through fail-over clustering and disaster recovery technics. For many organizations this requirement translates into resource over-allocation and accepting wastefulness (Yan & Wang, 2020). Meanwhile, cloud native systems support instanteous provisioning, elasticity, and can go global in minutes. These capabilities promote more efficient scheduling and allocation methodologies. However, even mature businesses limit their cloud exploitation to stateless, not stateful, services. Architects need to define frugal patterns that leads reliable systems operating above unreliable and dynamic hardware.

# Problem Statement

Traditional monolithic systems implement high-availability within the *finite* constraints of private data centers. Incontrast, cloud native solutions exploit virtually *infinite* scalability across multiple global regions. Despite this additional flexibility, most businesses do not fully exploit the performant high availability potential that comes from operating on public cloud platforms. Researchers must define new architectural tactics that leverage the cloud’s unique characteristics.

The dichotomy of traditional and cloud-native high-availability is most apparent with stateful services. Unlike stateless services, it is challenging to elegantly handle stop-faults. Platforms like Apache Spark, Flink, and Storm mitigate these issues with checkpointing. However, this solution decreases overall throughput by 35-40% (455-570MB/s versus 755-900MB/s) (Cheng, Huang, & Lee, 2019, p. 12). Businesses must provision extra resources to offset this degragation. Increasing the cluster size also means greater chances of a component failing due cross-component communication and I/O requirements.

# Goal

# Relevance and Significance

# Literature Review

1. Disaster Recovery Techniques
   1. Reasons that systems become unavailable
2. Cluster Service Schemes
   1. Using Clustering tech is fairly traditional
   2. Expand to include Performance Evaluators (based on aging)
3. Design and Implement PaaS
   1. Patterns on Kubernetes
4. Comparison of zero downtimes
   1. Blue/Green patterns
5. H+K High Availability for Stateful Services
   1. Modernizing from 1+1 to N+K overhead
6. Instanatneous Networking
   1. Operating in partially recovered state
7. Building Dependable Distributed Systems (Zhao, 2014)
   1. See 8120-7 about checkpointing and related strategies
8. Performance of distributed systems
   1. AF-Stream avoiding checkpointing
9. Aurora: Avoiding Distributed Consensus

# Approach