LAB ASSIGNMENT: 02

SOFTWARE DESIGN &ARCHITECTURE

FA22-BSE-043

SUBMITTED BY: TAYYAB ALAM

SUBMITTED TO: SIR MUKHTIAR ZAMIN

Major Architectural Problems and Their Solutions

(Software: Amazon)

Outline

Introduction

Architectural decisions shape the core of software systems, influencing scalability, performance, and maintainability. Over time, evolving requirements and technological advancements introduce challenges, prompting architectural modifications. This assignment delves into five prominent architectural problems faced by Amazon at various stages of its evolution and illustrates how system modules or entire architectures were revamped to resolve these issues.

Thesis Statement

Amazon's architectural evolution showcases how addressing scalability, performance, and data management problems through microservices and distributed systems ensures seamless growth and enhances user experience.

Overview of Architectural Problems

Amazon's journey from an online bookstore to a global e-commerce and cloud computing giant involved several architectural overhauls. These problems, if left unaddressed, could have hindered user experience, slowed down development, and reduced platform reliability. The following sections highlight five significant architectural issues faced by Amazon during different phases of its growth and the solutions implemented to mitigate them.

Major Architectural Problems and Case Studies (Focus: Amazon)

1. Scalability Bottlenecks (Early 2000s)

Problem: As Amazon expanded its product catalog and user base, the system could not efficiently handle the increasing load, leading to slow performance.
Solution: In the early 2000s, Amazon transitioned to a microservices architecture, breaking down the monolithic application into independent services responsible for specific tasks. This significantly improved scalability and availability.

2. Monolithic Structure Constraints (Mid-2000s)

Problem: Amazon's monolithic architecture resulted in slower development and deployment cycles, creating bottlenecks in feature delivery.
Solution: By 2006, Amazon moved further towards microservices, allowing development teams to work independently on different services. This facilitated faster development, easier scaling, and continuous deployment.

3. Performance Degradation (2010s)

Problem: With millions of users accessing the platform simultaneously, performance issues emerged, impacting page load times and transactional throughput.
Solution: Amazon addressed performance issues by implementing distributed databases, caching systems, and content delivery networks (CDNs) to reduce latency and improve performance.

4. Security Vulnerabilities (2015 and Beyond)

Problem: As Amazon expanded its services globally, the platform faced increased cyber threats and vulnerabilities.
Solution: Amazon Web Services (AWS) adopted advanced security protocols, including encryption, secure API gateways, and comprehensive monitoring tools to fortify system security.

5. Inconsistent Data Management (Late 2010s)

Problem: Managing data across multiple regions led to inconsistencies and synchronization issues.
Solution: Amazon introduced event-driven architecture and centralized data pipelines, ensuring real-time data consistency and reliability across distributed services.

Replicating and Solving One Problem

For this assignment, we replicate the **scalability bottleneck** problem faced by Amazon during the early 2000s. This demonstrates how transitioning to load balancing and distributed systems resolves the issue.

Coding Example

```
// Single Server Handling All Requests (Before Scaling)
public class WebServer {
    public void handleRequest(String request) {
        // Process request logic
        System.out.println("Handling request: " + request);
    }
}
// Distributed Load Balancer Approach (After Refactoring)
import java.util.ArrayList;
import java.util.List;
```

```
public class LoadBalancer {
    private List<WebServer> servers = new ArrayList<>();

    public LoadBalancer() {
        servers.add(new WebServer());
        servers.add(new WebServer());
        servers.add(new WebServer());
    }
    public void distributeRequest(String request) {
        int serverIndex = request.hashCode() % servers.size();
        servers.get(serverIndex).handleRequest(request);
    }
}
public class WebServer {
    public void handleRequest(String request) {
        // Process request logic
        System.out.println("Handling request: " + request);
    }
}
```

Conclusion

Amazon's architectural problems and solutions serve as a prime example of how proactive system refactoring ensures scalability, performance, and user satisfaction. By addressing these issues through microservices and distributed systems, Amazon solidified its position as a leading e-commerce platform. This assignment highlights the importance of adapting architectural designs to meet evolving demands and sustain growth.

THE END