Question 1 / Solution 1:

If there are two physical servers available and one public network connection, please:

a. Provide a physical servers plan for all systems.

b. Provide a systems architecture.

c. Provide a network design for all systems.

Overview

This architecture plan is designed to support five systems (S1, S2, S3, S4, S5) with a microservices architecture style. The systems will run 24x7, with public access for end-users, and a weekly backup strategy. Given the constraints of two physical servers and one public network connection, we will optimize resource allocation and network design.

1. Physical Server Plan

Physical Server Allocation:

- Server 1: Host for S1, S2, and S3
    - S1: Linux-based standalone application with MySQL
    - S2: PHP/MySQL Web application
    - S3: PHP/PostgreSQL Web application
- Server 2: Host for S4 and S5
    - S4: Python/Django/SQLite3 Web application
    - S5: Java Web application

2. System Architecture

Component Breakdown

- S1 (MySQL Application)
    - Components:
        - Data Processing Module
        - MySQL Database

        - Responsibilities:
            - Collect and process data, share with S2.
- S2 (PHP/MySQL Web Application)
    - Components:
        - Web Server (Apache/Nginx)
        - PHP Application
        - MySQL Database
    - Responsibilities:
        - Receive data from S1, serve web requests from public users.
- S3 (PHP/PostgreSQL Web Application)
    - Components:
        - Web Server (Apache/Nginx)
        - PHP Application
        - PostgreSQL Database
    - Responsibilities:
        - Share data with S5.
- S4 (Python/Django/SQLite3 Web Application)
    - Components:
        - Web Server (Gunicorn)
        - Django Application
        - SQLite3 Database
    - Responsibilities:
        - Share data with S5.
- S5 (Java Web Application)
    - Components:
        - Web Server (Tomcat)
        - Java Application
        - PostgreSQL Database
    - Responsibilities:
        - Receive data from S3 and S4, serve web requests from public users.

3. Interaction Patterns

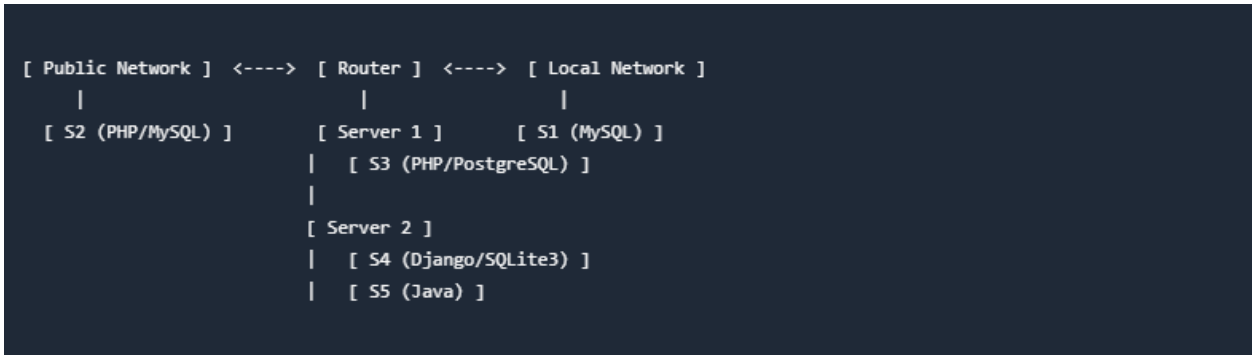
- Data Flow
    - S1 to S2: Data is shared via a RESTful API or direct database replication.
    - S3 to S5: Data is shared through a RESTful API or message queue (RabbitMQ).
    - S4 to S5: Data is shared through a RESTful API or message queue (RabbitMQ).
- User Access
    - Users access S2 and S5 through the public network, while S1, S3, and S4 are accessible only within the local network.

4. Network Design

Network Topology

- Router Configuration
    - A primary router connects the local network (for S1, S3, S4) and the public network (for S2 and S5).
    - Firewall rules should be implemented to restrict access to S1, S3, and S4 from the public network.

Network Layout



5. Implementation Recommendations

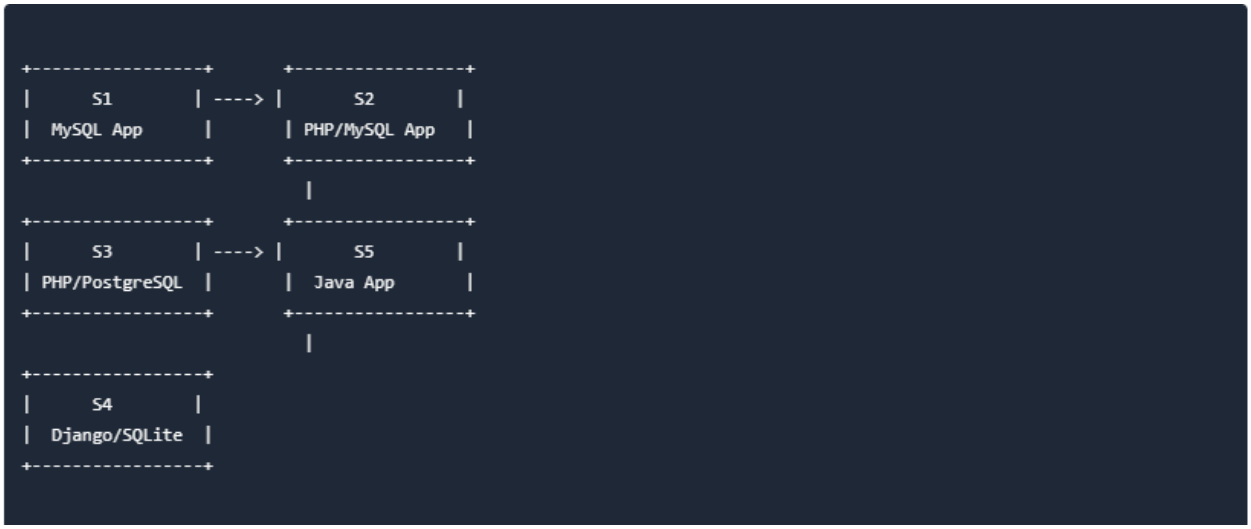
- Technology Stack
    - S1: Use MySQL with a stable Linux distribution (Ubuntu/CentOS).
    - S2: PHP with Laravel framework and MySQL.
    - S3: PHP with Symfony framework and PostgreSQL.
    - S4: Django with SQLite3 for development, with a plan to migrate to PostgreSQL if needed.
    - S5: Java with Spring Boot and PostgreSQL.
- Backup Strategy
    - Schedule weekly backups using `cron` jobs on each server.
    - Use tools like `mysqldump` for MySQL, `pg_dump` for PostgreSQL, and `sqlite3` for SQLite to create backups.
    - Store backups on a separate storage solution (e.g., NFS or cloud storage).
- Monitoring and Logging
    - Implement monitoring solutions (e.g., Prometheus, Grafana) to track system performance.
    - Use centralized logging solutions (e.g., ELK Stack) for better traceability and debugging.

6. Key Considerations

- Scalability: While the current design fits within the constraints, consider containerization (e.g., Docker) for future scalability.
- Security: Implement HTTPS for public-facing applications and secure API endpoints.
- Testing: Conduct thorough testing of data sharing mechanisms and user access controls.
- Documentation: Maintain detailed documentation for system architecture, deployment processes, and backup procedures.

UML Diagrams

Component Diagram



This architecture plan provides a comprehensive overview of the system design, addressing the requirements and constraints while ensuring scalability and maintainability.