Asynchronous Grading Architecture Documentation

Overview

The asynchronous grading architecture aims to provide clients with the ability to submit grading requests, receive quick acknowledgments, and check the status of their requests asynchronously. The design includes the introduction of unique request IDs, queuing mechanisms, storage for request data and results, and status tracking.

Components

Client Application

- The client application allows users to submit grading requests and check the status of their requests.
- It communicates with the server using command-line arguments:
  - `./submit new <serverIP:port> <sourceCodeFile>`: Submit a new grading request.
  - `./submit status <serverIP:port> <requestID>`: Check the status of a previous request.

Server Application

- The server application is responsible for receiving grading requests, processing them, and providing status updates.
- It uses a thread pool for processing grading requests.

Request ID Generator

- Generates unique request IDs for each new grading request.
- Ensures that request IDs are unique and not reused.

Request Queue

- Manages a queue of grading requests.
- Accepts new grading requests and assigns them unique request IDs.
- Ensures that requests are processed in the order they are received.

Status Tracker

- Tracks the status of grading requests.
- Associates request IDs with the current status, which can be "Accepted," "In Queue," or "Processing Complete."
- Provides progress updates when clients send status check requests.

Storage System

- Persists grading request data, request IDs, and grading results.

- Maintains a database or file system to store this information.
- Ensures that even if the server restarts, it can retrieve the results of past requests.

Log File

- Logs events and actions, such as request submissions and status checks.
- Provides a record of system activities for troubleshooting and auditing.

Workflow

Grading Request Submission

The client submits a new grading request using the `./submit new` command.
The server generates a unique request ID and associates it with the grading request.
The request is queued for processing.
The server sends an acknowledgment response to the client, including the request ID.

Checking Request Status

The client sends a status check request using the `./submit status` command with the associated request ID.
The server looks up the request ID to determine the request's status.
The server responds with the appropriate status information:

- "Accepted" if the request is in the queue.
- "In Queue" with the position in the queue if the request is still waiting.
- "Processing Complete" with the grading results if the request has been processed.
- "Not found" if the request ID is invalid or the original request no longer exists.

Data Persistence

- Grading request data, request IDs, and results are stored in the storage system.
- A combination of in-memory storage for recent requests and secondary storage (logs or database) for older requests ensures data availability even after server restarts.

Error Handling

- The system is designed to handle various error scenarios, including invalid request IDs and client inquiries for non-existent requests.
- Log files can help in diagnosing and troubleshooting errors.

Scalability

- The system can be scaled horizontally by adding more servers and balancing the load.
- Load balancing and clustering mechanisms can be implemented for high availability.

Security

- Implement authentication and authorization mechanisms to ensure that only authorized users can submit grading requests and check their status.

Resilience

- The system is designed to be resilient, even if clients submit requests and later check their status after a long time.

Monitoring

- Implement monitoring to keep an eye on system performance and resource usage.
- Monitor the status of the request queue and grading processor.

Logging

- Implement comprehensive logging to record events and errors for auditing and troubleshooting.

Conclusion

The asynchronous grading architecture with request IDs and status checking enhances the user experience by allowing clients to submit grading requests, receive quick

acknowledgments, and check the status of their requests at their convenience. The architecture ensures data persistence, scalability, error handling, and security to provide a robust and responsive grading system.