Asynchronous API Benchmarking System

Overview

In this challenge, you will develop an asynchronous benchmarking system using Python's `asyncio` and `aiohttp` and a simple dummy API server. The main focus is on the client-side benchmark system that measures and reports the latency and throughput of the API.

Objective

Create a benchmarking system that interacts with a dummy API server to:

- Simulate multiple workers making parallel API requests.
- Collect and analyze performance data such as latency and throughput.
- Demonstrate the API's performance under different load configurations.

Requirements

API Server

1. Simple Server Setup: Implement a minimal server using Flask or FastAPI that returns predefined responses quickly and reliably without complex logic.
2. Response Content: The server should generate responses with adjustable parameters (like number of tokens) for benchmarking purposes.

Benchmark System Design

1. Worker Function: Develop an `async` function `worker` that processes API requests to the dummy server.
2. Rate Limiting: Think how you will handle rate-limiting such that if the API endpoint allows `X requests per minute` you'd be able to send requests in parallel always maintaining the `X` threshold.

3. Benchmark Function: Build `single_benchmark` to manage multiple workers handling parallel requests.

Performance Metrics

- Calculate metrics such as average input/output tokens, median latency, and tokens per second.
- Test different output token configurations to evaluate performance impacts.

Results and Reporting

- Develop a reporting format for the benchmark outcomes.
- Summarize performance metrics and provide insights based on the collected data.

Evaluation Criteria

- Functionality: Accuracy of the measurement and reporting of the specified metrics.
- Scalability: How well the system handles increased numbers of workers or requests.
- Code Quality: Organization, clarity, and best practices in asynchronous programming.

Submission Guidelines

Submit your solution as a Git repository containing:

- Source Code: All necessary source files for both the benchmarking system and the minimal API server.
- Dependencies: A `requirements.txt` file listing all necessary dependencies.
- README: Instructions for setting up and running the benchmark.

Setup Instructions

1. Set up a new Python environment
2. Install the required dependencies

3. Add a makefile to start the dummy API server
4. Implement your benchmarking system as specified
5. Run the server and the benchmarking system to test API interactions and collect data
6. Analyze the results and document your findings.

Good luck, and we look forward to your solution!