Take Home Assignment
Project Report

Name: Rishi Gupta

This system implements a cryptocurrency price-monitoring solution which collects data from a public API by Binance. It employs a dual database architecture: SQLite for transient storage of raw data fetched by the API, and PostgreSQL for persistent storage of running metrics.

Architecture

Data Collection Layer

- Calls the Binance API at 5-second intervals for cryptocurrency data
- Implements basic error handling for API failures
- In-memory running metrics for each symbol/cryptocurrency during collection

Storage Architecture

1. **Primary Storage (SQLite)**
    - Stores raw price data along with timestamps. Data cleared daily.
    - Chosen due to no-configuration setup and high in-memory write performance.

2. **Persistent Storage (PostgreSQL)**
    - Stores hourly aggregated metrics (OHLC, latest price, average price).
    - Provides support for maintaining historical data by down sampling.
    - Provides future scalability due to the ability to handle complex queries.

Design Concerns and Solutions

- The current system polls the Binance API every 5 seconds. As the number of symbols or sampling frequency are increased, it may hit the limits of the API. Currently, there are no mechanisms in-place to backoff or perform rate limiting.
- API rate limiting should be implemented.
- Since SQLite is an in-memory storage, it can be problematic in future if data (to be handled) is increased or cleanups are not performed regularly enough. Increasing the number of symbols to process may also hit a bottleneck with PostgreSQL during hourly inserts which are performed in bulk.
- Since persistent storage deals with hourly data, strategies can be used for partitioning of PostgreSQL data for better performance with queries which perform time analysis.
- Currently, no backup mechanism is provided for the in-memory SQLite database, which on crashing can lead to a potential loss of 1-hour of metrics.
- No retry mechanisms have been implemented in case of API failure.