

Enhancements

Is this system scalable? How can it cope with high traffic? Can you identify bottlenecks and suggest an improved design and architecture? Feel free to suggest a complete different approach, but be sure you can obtain the same goal.

Potential issues with the current system

Scalability

There can be different demands for *Transaction* API and *Get-Stock-Price* APIs. Because at the peak trading hours, Transaction API will have higher demand as compared to GET. This may lead to system overload and degraded performance.

Its better to split these two modules into separate modules.

High Coupling

There is a high level of coupling between the module responsible to receiving and processing real time transaction and the module to supply the Stock Price.

Ideally, the module related to real time processing should be a separate component using a message broker like Azure Event Hub.

Frequent Data Computation

We are calculating the average of all stock prices by scanning all transactions in the database to get the latest stock price. Over a period, as the number of transactions grow in number, this approach will result in degraded performance.

The ideal way is to have a different service which will compute the average stock price in real time and persists the average price in a highly available database like cosmos or a globally distributed cache.

Proposed new architecture

- Introduce a message broker or event hub which will have high throughput and is easily scalable depending on the incoming transaction volume
- Each incoming transaction triggers an Azure Function which will persist the transaction details in the database and recomputes the average price of the stock
- Expose a Web API to supply the average stock price
- Based on the performance NFR, a globally distributed cache like Redis can be used to hold the average price of all stocks

