Report

Experimental setup:

We will be using two applications as a part of our experimental setup for the project. Both the applications are the same but one application is built using the Rest API and the other application is built using the soap API.
These applications are chatroom applications which have the following features:

- The app allows users to create chat-rooms, list all existing rooms, join existing chat-rooms, and leave a chat-room.
- If the user connects to a chat room all previously sent messages of that room should be
- displayed. New messages sent by the user or other connected users should be shown to the user with a maximum delay of 1.5 seconds.

Technologies used:
ChatRoom (REST):
This app is build using Flask and C# as the client. The app also requires Docker to set up and run.
ChatRoom (SOAP):
This app is build using C# as the Server and client. The app also requires Docker to set up and run.
Npm Loadtest:
Loadtest will help us test our app in different scenarios, we can set the number of requests to send and also set the number of concurrent clients we want to conduct our tests with. Npm loadtest will give us response times of the requests we make.

Experiments:

Test behavior based on arrival rates:
Data volume:

- Test response time of both applications when we send a large text file as a string to create a group named that string.

No of requests:

- We will test this criteria by increasing the number of requests by a single client and see how response times change with requests per second.

Concurrency:

- We will increase the number of concurrent requests allowing us to simulate a large number of users and compare both apps based on response times.

Data volume Test:
We will use npm loadtest for the Rest App
Command: *npx loadtest -n 1 -p POST-Data1MB.txt http://localhost:80/create* .

This will allow us to send different volumes of data. Eg. 32KB file, 256KB file, 1MB file, 10MB file. The results would be noted in a spreadsheet.
For our soap App we will read a file using Client.cs and send the varying volumes of text (32KB, 256KB) to the chatroom and see how response times change with increasing volume of data.

Test with multiple requests and concurrency:

We will use npm loadtest**:**
Command: *npx loadtest -n 1 c -1 -p -k http://localhost:80/create* for the Rest App and
Command: *npx loadtest -n 1 -c 1 -k http://localhost:8082/NumberService.asmx/listRooms*.
We will incrementally increase the number of requests as well as the number of clients to see how the apps behave. And record the data in a spreadsheet.

Test when we have a lot of already existing chat rooms:
We will incrementally add more chat rooms and then check the response times for each request for some key methods of both of our Apps.
Test some key methods of the app:
we will send a request to our app with the following methods, when we have the following number of groups , 10, 100,1000,1000 and see what the response times are:
createRoom(): sendMsg(): getMsg():

Report And Conclusion:

As we conducted our experiments for both the apps REST and SOAP. There seemed to be a clear winner in terms of performance.
We first tested Both our Apps, by sending incrementally larger volumes of data. Using the method createRoom. We sent varying text sizes 32KB, 64KB, 512KB, 1MB, 5MB, 10 MB. and we noted the response times in the Results.xls spreadsheet. We resched http limit for soap after 1MB. But the results are clear regardless.

The results were as follows:
The response times of the REST App were significantly less than the response times of the SOAP App, As the data increased the response times of both the apps increased but the Rest app was on average 4-6 times faster than the SOAP app.
This was especially true for the Post requests. We also noted that when the SOAP app was first started up using docker-compose up . The first few requests to the app were a lot slower than the later requests made to the SOAP app.

In the second test we tested our app with concurrent requests. Again we found a big difference between response time of bothe the apps , Rest was significantly faster than SOAP. Again I believe the first set of requests sent to the SOAP were significantly slower. And as a result we got really big response times for fetching data concurrently from the SOAP app, the Rest app on the other hand showed significantly faster results.

Doing some research online I found out that the first load penalty was the reason behind the slower first set of requests made to the SOAP app.

“The problem is that for the first request, there is extra time taken for the JVM to load classes from the disk (IFS) into memory and do bytecode verification.
Additionally, when Java Classes have not been used for 20 minutes, the JVM reclaims these Class objects. So after one hour there are no Classes in memory, so the JVM needs to reload the Classes from disk (IFS) again. Thus the first-time Load Penalty applies again”. https://www.lansa.com/support/tips/t0483.html (2022).

In the third experiment we made a first request to the SOAP app and waited a few sec to get rid of this first time load penalty. And then used all three factors, data, number of requests and concurrency to test both the apps.
The results this time around showed a significant increase in performance for the SOAP app, but still the response times were 2-3 times slower than the REST app. The increase in performance could be also due to the fact that these were get requests.

And lastly we tested some of the key methods of our app ie. **createRoom(): sendMsg(): getMsg():**

Again the results showed that REST was 4-5 times faster than SOAP.

Failure handling:
SOAP has an inbuilt Retry mechanism which will retry the request if there was an error or if the server did not respond. Whereas REST leaves error handling to the client and the client program has to do all the error handling .

Security:
SOAP is more secure than REST, as it Supports SSL, and also WS- Security (XML Encryption and Signature).
On the other hand security depends on the service designer.

Scalability:

Both the REST and the SOAP app can handle mutiple clients and multiple concurrent requests, However if we take into account the First Load penalty of the SOAP app and its High bandwidth usage due to the XML overhead, it might have a significant impact on the performance of the

SOAP app. There are some ways to eliminate this First Load penalty of the SOAP app but still REST is super lightweight and uses less bandwidth and thus handles multiple clients better than SOAP in our use case..

SOAP or REST? Which is better?

Both the SOAP and the REST apps have different strengths, on one hand REST is lightweight, uses less bandwidth, uses caching to prevent multiple requests to the server from the browser. Hence Apps where performance is a major concern such as web browsing, social media, chat apps like ours. REST seems to be a better fit for such purposes, whereas SOAP shines in Security, Reliability and failure handling, and such systems where these factors are immensely important such as Financial, Payment Gateways, Telecommunication. SOAP would be a better fit.