Dylan Tucker, dst833, 11235055

The following experiments are intended to compare the differences in performance between simple object access protocol, or SOAP, and representational state transfer, or REST. Two separate web services will be created, one will use the REST architectural style, while the other will utilize the SOAP messaging protocol. The experiments will be conducted on a local machine using the programming language Python with the REST service using the Flask restful API, and the SOAP service using the Spyne API. These web services will consist of a simple fibonacci function which takes in one integer parameter called “n”, and returns the nth number in the fibonacci sequence. The clients for the RESTful web service will be making requests using hypertext transfer protocol, or HTTP, and will be getting a JSON key value pair as a response. The SOAP clients will use SOAP envelopes to make requests and receive XML objects as a response.

There will be 3 types of experiments conducted: a comparison of the arrival times for each web service, a stress test of the throughput of each web service, and the last will compare the scalability, or how many clients can connect to the service at once. The first experiment will use a for-loop to send 100 requests to calculate the 5th fibonacci number. The arrival times will be calculated using Python's time library and each time stamp will be appended to a list. The data in this list will then be plotted on a graph using Python’s matplot library, which will be shown inside the “Figures” folders attached to this project. The second experiment will make 5 requests for the 1000000 th fibonacci number, and save the timestamps for each arrival time and store them in a list. The arrival times for the throughput experiment will also be graphed using Python’s matplot library. The final experiment will use multiple clients to send simultaneous requests to the same web service. The clients will use the same code from the first experiment, with 10 clients making 100 requests for the 5th fibonacci number in parallel, while the arrival times will be stored in a list and plotted on a graph. The scalability experiments will be completed using Python’s multiprocessing library.

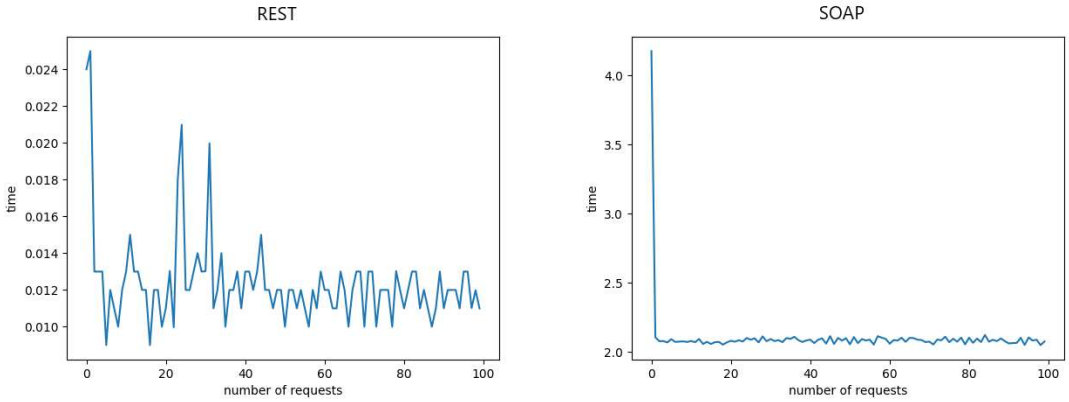

Figure 1. The results of the first overhead experiment where requests are plotted against their arrival times.

The results from the first experiment show that the REST web service performed significantly better than the SOAP service. The average arrival time of a REST request during the overhead experiment was around 0.012 seconds with a maximum time at 0.025 seconds and a minimum time of 0.009 seconds. The average SOAP request during the overhead experiment was approximately 2.10 seconds with a maximum and minimum at 4.20 seconds and 2.05 seconds respectively. The standard deviation of the REST web service was also significantly

better at 6.11e-06 seconds as opposed to 0.043 seconds for the SOAP service. The throughput experiments also had similar results with REST having an average arrival time of approximately 0.11 seconds and SOAP with 2.09 seconds. The scalability experiments were also in favor of REST with an average arrival time of 0.12 seconds for the first experiment while SOAP achieved a 2.10 second average arrival time. However, the REST service had a standard deviation of 0.002 seconds for the first scalability experiment, which is significantly larger than the overhead experiments. SOAP managed a standard deviation of 0.049 seconds which is a similar result to the overhead experiments. This indicates that SOAP may be a more consistent and reliable way of supporting a large number of clients than REST, despite its longer arrival times.

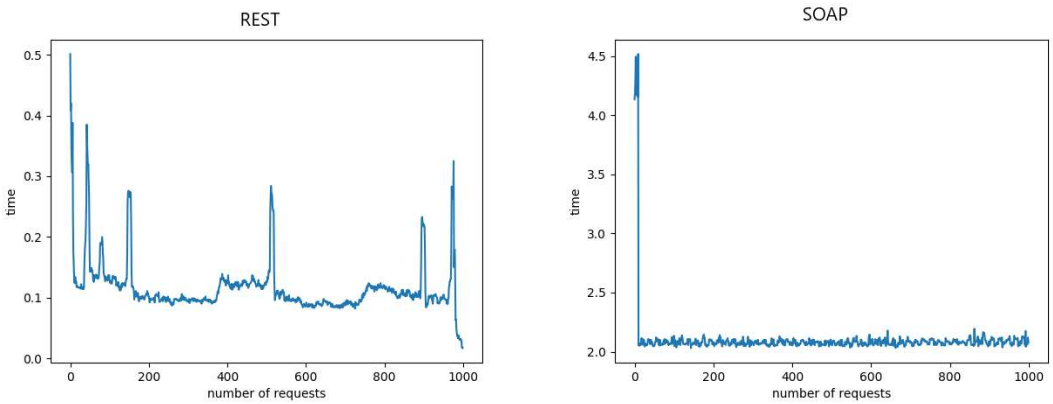

Figure 2. The results of the first scalability experiment where requests are plotted against their arrival times.

Both the REST and SOAP web services were consistent in how they performed for each experiment. The SOAP service had an average arrival of 2.01 seconds for all 3 overhead experiments, while REST had 0.01 seconds for the same experiments. None of the SOAP requests across all 9 of the experiments managed to be less than 2 seconds. This indicates that Spyne or other SOAP API's may have more overhead causing significantly higher delays between each request than Flask. Most of the variability between the SOAP requests occurs after 2 seconds of the request being sent. If 2 seconds is subtracted from the SOAP requests then the results are more comparable to REST with an average time of 0.10 seconds during the first overhead experiment. With the first 2 seconds subtracted, SOAP consistently outperformed REST in the scalability experiments. Outside of performance, the benefit to REST is its flexibility and ease of implementation, while SOAP is inflexible and has limited support on Python. The benefit to SOAP is its built in error logic that handles communication failures and ensures security and reliability.[1] If security and reliability are important features when building a web service, then SOAP is the proper choice, however, REST provides speed, flexibility, and ease of implementation that make it a better choice in most scenarios.

[1] Amazon. "SOAP vs REST." Amazon. Accessed November 29, 2023. https://aws.amazon.com/compare/the-difference-between-soap-rest/#:~:text=SOAP%20and%20REST%20are%20two%20different%20approaches%20to%20API%20design,exchange%20data%20in%20multiple%20formats.