Distributed Systems: Session 3

Cloud-based deployment with IaaS

Yixuan Li r0927212

Wiktoria Radecka r0787181

Nadia Putri r0817511

Belgin Ayvat r0877587

Necessary links:

http://104.210.36.2:8080/rest/meals/

http://104.210.36.2:8081/ws/meals.wsdl

05/04/2025

Introduction

This investigation explores the practical challenges and opportunities presented by deploying remote service protocols in a cloud-based IaaS environment. Specifically, we have deployed remote services such as RMI, SOAP and REST on a set of virtual machines (VM) in data centers provided by Microsoft Azure. Afterwards, we assessed the performance of the services mentioned using Azure Load Testing both in terms of the clients and servers.

Method

To test various remote services deployed across the globe on Azure's VM, we developed several test cases. The tests cover different numbers of parallel clients, sustained load durations, and varying load patterns. These tests aim to simulate real-world load conditions as closely as possible while remaining affordable within the limits of the student credits offered by Azure. Specifically, we decided to test with 10 and 100 parallel clients. These clients sent concurrent requests for 2, 5, and 10 minutes, with each scenario incorporating a 1-minute ramp-up time to gradually reach full load.

In addition to the linear load increase, the 10-minute test scenarios also included a spike load pattern. In this setup, the number of clients doubled from 5 to 10 and from 50 to 100 clients over a 5-minute period, simulating a sudden surge in traffic toward the web services. This scenario mirrors real-life events such as time-limited promotions (e.g., Black Friday) that generate a temporary spike in usage on top of regular traffic. The various test scenarios are summarized in Table 1.

Number of parallel clients	Test Duration (min)	Load Type
10	2	Linear
	5	Linear
	10	Linear
	10	Spike
100	2	Linear
	5	Linear
	10	Linear
	10	Spike

Table 1: Test scenarios for the remote SOAP, REST and RMI services

To test the SOAP and REST services, Azure Load Testing was used. Test agents were deployed in three different locations (East Australia, Canada, and West Europe) to simulate client requests from different parts of the world. These agents targeted four virtual machines hosting the remote services (SOAP, REST, RMI), which were also distributed globally—in East US, West US, West Europe, and Japan.

For each test scenario, two operations were performed for both SOAP and REST services: requesting the largest meal and placing an order. To keep testing costs minimal, we only tested the RPC-style REST service. This decision is justified as performance differences between RPC-style and HATEOAS-style REST are generally negligible.

In the case of RMI, since it does not use HTTP for communication, the testing was carried out using JMeter scripts that spawned RMI clients on our local virtual machines. All clients executed the same request body to book a room, allowing consistent and repeatable test conditions across all runs.

Results

- ### Client Side Metrics

To describe the service performance from the client side, we assessed the metrics using load, error percentage, throughput and response time.

Load refers to the number of requests generated to simulate the traffic for each test. This provides context when the system's performance is being evaluated. The error percentage refers to the portion of the total requests that resulted in a failure or error response during the test.

SOAP and REST generate identical number of requests relative to one another (i.e. 20000 requests in *linear* pattern and up to 60000 requests in *spike* pattern). In comparison, RMI sent out around 30000 requests with linear pattern and with spike pattern.

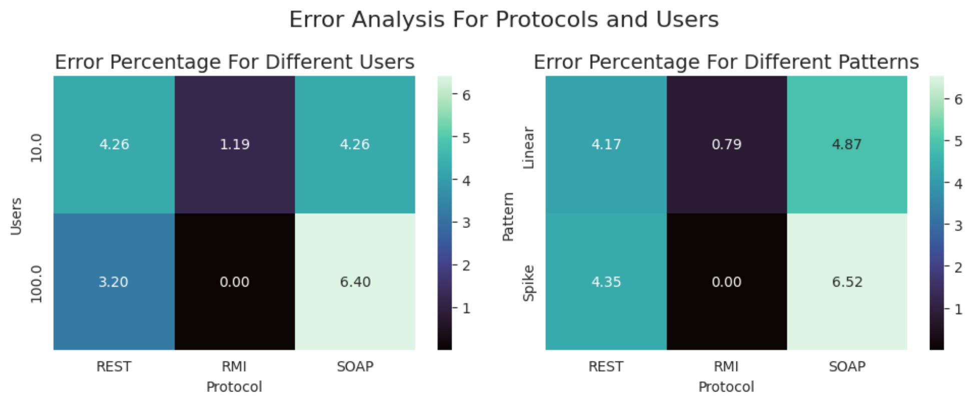


Figure 1

Figure 1 describes the error percentage found for each protocol and shows that on average, RMI protocol produces the least error, showing only 0-1% error. RMI also shows less errors when dealing less users and the *spike* load pattern. Requests sent through REST protocols experience the same trend, while the SOAP requests exhibit a higher error percentage with more users simulated and the spike pattern. SOAP requests also resulted in the highest error, up to 6.52%.

To identify bottleneck areas, *load versus error percentage* graphs is produced. To demonstrate the effect of distance between server and client, two such graphs are produced across different combinations of server (VM) and client locations. The graphs demonstrate how the error percentage changes with the number of requests in the case of the nearest and farthest distance between the server and the client. In this case, the nearest distance is between the Europe West client to a Europe West server (Figure 2) and the farthest is between the Australia East client to a US West server (Figure 3).

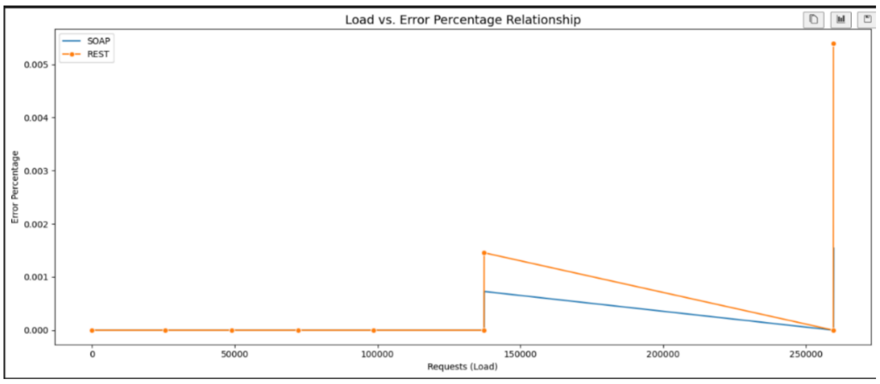


Figure 2

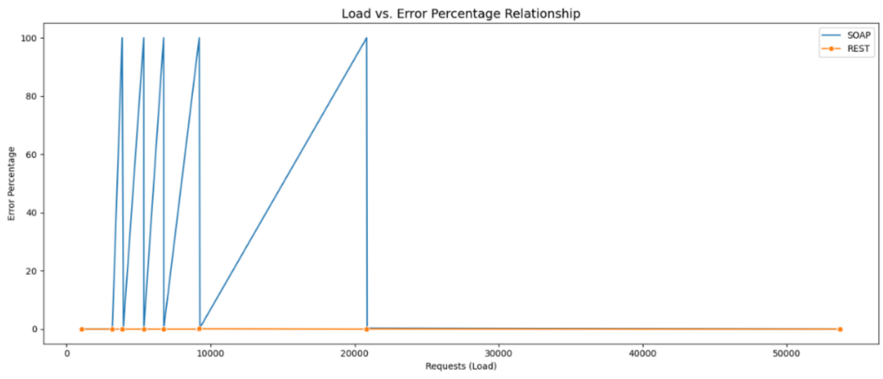


Figure 3

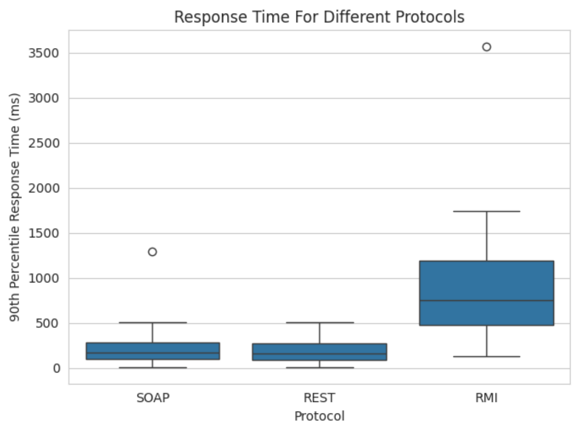
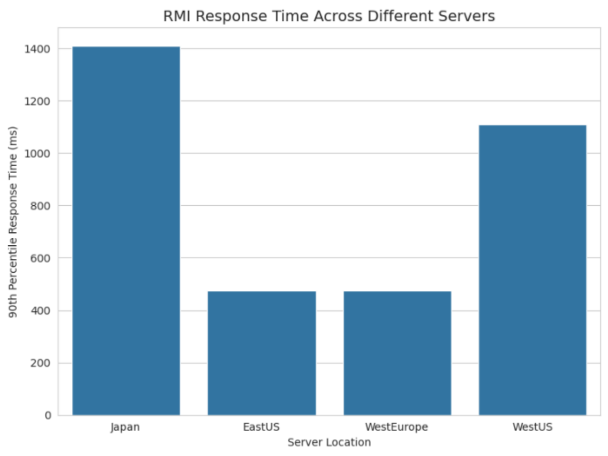


Figure 4

Next, we explored the performance using response time and throughput metrics. The response time is given as the time between the client sending a request and receiving the first byte of the response. We report that as 90th percentile to understand the distribution of response time more accurately as it is an important indicator of user quality experience. At the same time, we define throughput as the number of requests processed by the system per unit of time, measured in requests per second (RPS). This metric reflects the system's capacity to handle load which gives insight into scalability and resource utilization options.

From Figure 4, the median response times show REST performing the best with a tight distribution between 100ms to 250ms. SOAP protocol exhibits a similar performance with slightly higher values. In contrast, RMI displays significantly higher response times and larger variability ranging from 500ms to 1200ms.
To investigate the cause for such large distribution of RMI response times, we plotted them across different servers in Figure 5. From this, we observe two distinct performance tiers, one with East US and West Europe servers with response times around 450ms, and one with Japan and West US servers with response times of more than 1100ms. The clients' location for each of this server was Belgium and so we observe that more distant locations experience higher latency.



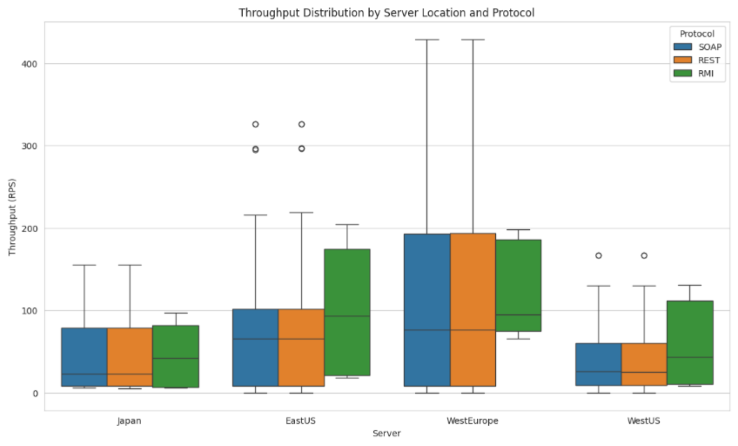


Figure 5 and Figure 6

In Figure 6, we compared throughput performance across different protocols and server locations. REST and SOAP protocols exhibit similar throughput values and distributions across all server locations. However, RMI shows location-dependent performance with a median throughput always being higher than that of the other protocols. Nevertheless, we observe REST and SOAP protocols have a larger variability in throughput values. In other words, for some values those protocols actually perform better than RMI.

When it comes to server location impact on the throughput, we see that the server in West Europe shows the highest throughput variability. It contains the highest peak performance, but it is also most inconsistent. Server in Japan displayed the most consistent performance across all protocols with the tightest distribution of all locations, suggesting that at that location we have the most reliable performance. Servers in East US and West US show moderate variability with outliers.

- **Server Side Metrics**

Finally, we assess the service performance using server-side metrics such as the CPU and memory usage. Specifically, we infer that the CPU usage of RMI is relatively stable, and linear or spike pattern will not cause a big impact. However, the CPU usage of SOAP/REST increases significantly when facing spike pattern, indicating that the demand for server computing resources is higher.

For further analysis of the impact of concurrent users on CPU usage, we plotted Figure 7 which shows that with RMI protocol, CPU usage rate remains almost the same when the number of parallel users increases from 10 to 100, showing good scalability and stability. In SOAP/REST protocol, the CPU usage rate rises when number of parallel users increases, especially in spike pattern.

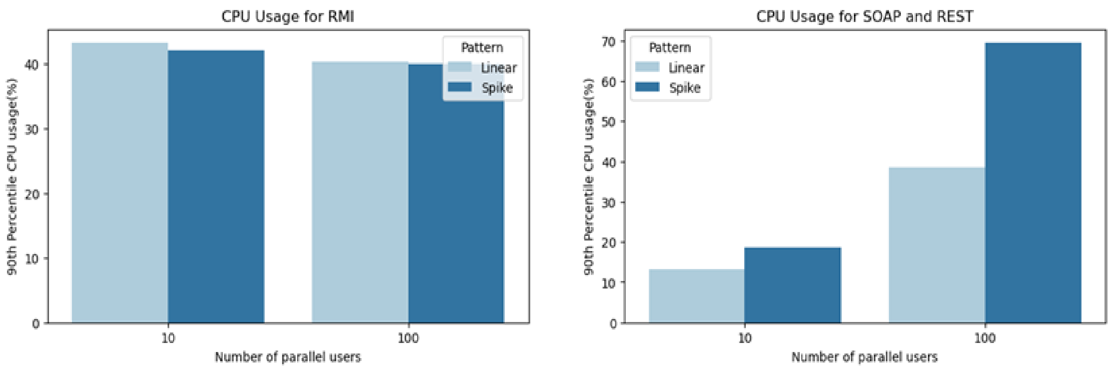


Figure 7

Memory usage in different situations is plotted in Figure 8. Both RMI and SOAP/REST takes almost 100% percent of memory usage, and the use of memory does not change much with number of parallel users or load pattern, implies that the memory requirements are relatively fixed.

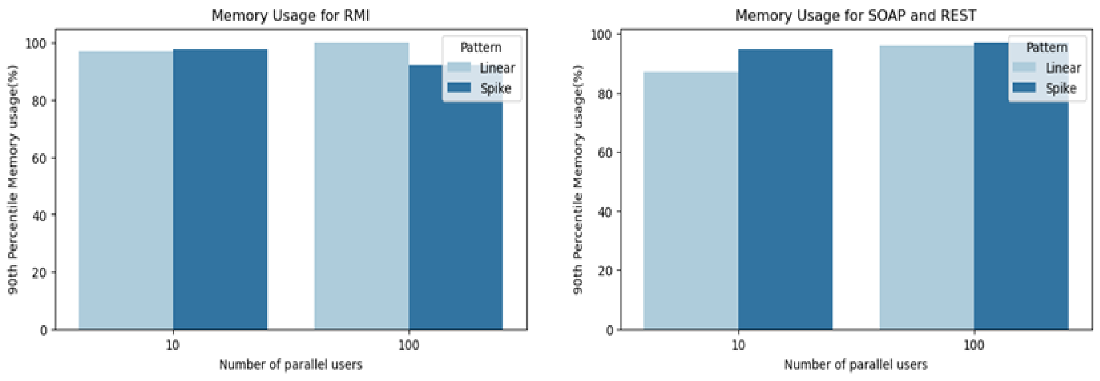


Figure 8

Discussion & Conclusion

The current study found that RMI has the lowest error rate in most cases and performs stably in spike tests while SOAP has the highest error rate, reaching 6.52% especially under high concurrency or heavy traffic. This may be related to SOAP's need to parse XML, which increases its response time and error compared to REST.

Another important finding was that REST has the shortest response time, SOAP is slightly higher, while RMI has the highest response time. However, RMI shows a higher average throughput in most tests, indicating that it has stronger processing power, but is more affected by geographical location.

Also, geographical distance has a significant impact on performance. Servers closer to the client have better response time and throughput, while servers in Japan or West US have significantly higher response times. This shows that network latency is a key factor affecting the performance of remote services, especially for protocols such as RMI that rely on underlying socket communication.

The results also find that RMI protocol shows better stability as both CPU and memory usage are stable under different numbers of concurrent users and load patterns. This makes RMI more suitable for deployment in systems that require high predictability and resource control.

Finally, although SOAP and REST have more modern interfaces, they consume more resources under high concurrency or under spike load pattern, especially the CPU usage would obviously increase, which may cause response delays or system bottlenecks.