KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

CAMPUS GROUP T LEUVEN

Analysis of cloud-based deployment with IaaS

A Case Study on Microsoft Azure's Infrastructure and Services

Chieh Fei HSIUNG r0819932

Geng LUO r0878899

Ivan IVANOV r0817605

Jiaao LIU r0880437

Academic Year 2023 - 2024

1 INTRODUCTION

The advent of cloud-based deployment models has significantly transformed how organizations approach IT infrastructure, through the utilization of Infrastructure as a Service (IaaS), which offers a flexible, scalable, and cost-effective alternative to traditional on-premise solutions. This case study focuses on deploying services on Microsoft Azure, and evaluating the SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) services using the HATEOAS (Hypertext As The Engine Of Application State) approach. By leveraging Azure's global infrastructure, this research aims to understand the performance, scalability, and reliability of cloud-based deployments, highlighting the benefits and challenges of adopting IaaS in a cloud-centric world.

Link to the service:

SOAP: http://jiaao@20.11.32.88:8081/ws

RESTful: http://jiaao@20.11.32.88:8082/rest/meals

2 TESTING STRATEGY

2.1 Test Environment

A baseline test with 100 users located at the same region as the server is performed to serve as a reference for both SOAP and REST services. This indicates the optimal performance achievable if clients are in the proximity of the server.

Moreover, to assess the performance and scalability of cloud-deployed services and evaluate the impact of geographical distribution on service efficacy, the server is set up in East Australia, while client simulations were conducted from diverse global regions, including North Europe, Canada, and Japan. This setup enabled a nuanced understanding of how location influences the responsiveness and reliability of cloud services.

2.2 Scenarios

The experiment involved a series of load tests, detailed in Table 2.1, designed to simulate varying levels of user engagement from different geographical regions to understand the performance under different load conditions.

In conducting the load tests, tests from Japan and Canada commenced two minutes before the North Europe test, creating a time shift in the resulting analysis diagrams.

Table 2.1: Load Test Plans Configuration

Parameters	Baseline	Test 1	Test 2	Test 3
Client Regions	1	3	3	3
Concurrent Users	100	1k	2k	5k
Ramp-up time	5 min	5 min	5 min	5 min
Duration	30 min	30 min	20 min	20 min
Client Region	AUS	EU, CA, JP	EU, CA, JP	EU, CA, JP
Server Region	AUS	AUS	AUS	AUS

2.3 Tools, Techniques and Metrics

We employed Apache JMeter and Azure Load Testing for simulation and orchestration of load tests, respectively.

Key performance indicators-including client-side response times, requests per second, errors, and server-side CPU utilization-were recorded and analyzed, which provided a comprehensive view of the scalability, reliability, and efficiency of the deployments on Azure. Notably, load engine health metrics such as CPU percentage, memory usage, and network bandwidth were monitored and remained within optimal ranges throughout testing, confirming that our testing infrastructure was not a limiting factor.

3 EMPIRICAL RESULTS

3.1 Client-Side Metrics

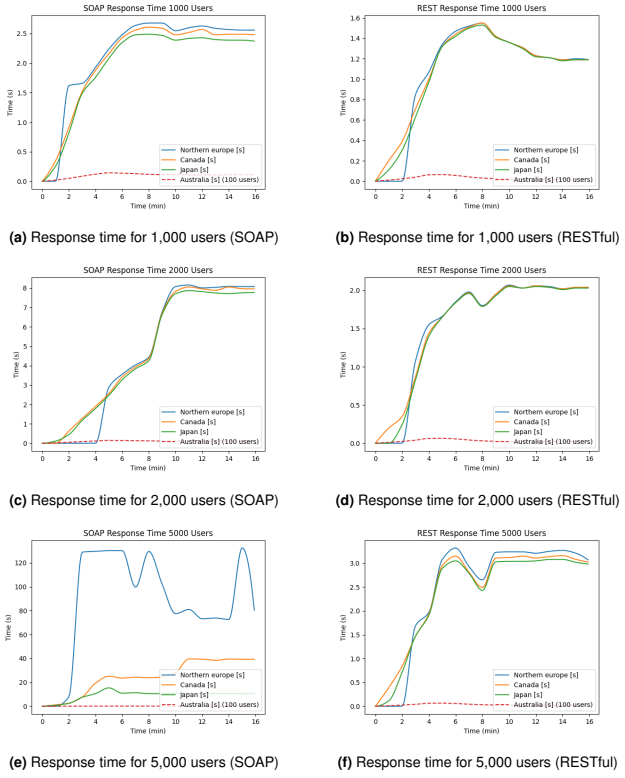
The quality of service performance is evaluated based on indicators of response time, throughput, and errors.

3.1.1 Response Time

The empirical analysis of SOAP and RESTful service response times elucidates distinct response behaviors under varying user load conditions. The examination revealed two salient phenomena influencing performance: the geographical proximity between client and server nodes, and the inherent server request handling algorithms. Figure 3.1 delineates the server's temporal response across different implementation scenarios.

SOAP Response Time

A macroscopic synthesis of the results, as portrayed in Figure 3.1, demonstrates a direct correlation between the number of concurrent user requests and response latency. Specifically, the data indicated a maximum average response time of 2.55 seconds for a load of 1,000 users,



(a) Response time for 1,000 users (SOAP)

(b) Response time for 1,000 users (RESTful)

(c) Response time for 2,000 users (SOAP)

(d) Response time for 2,000 users (RESTful)

(e) Response time for 5,000 users (SOAP)

(f) Response time for 5,000 users (RESTful)

Figure 3.1: Response times for different numbers of concurrent users

which is captured in Figure 3.1a. As the load escalates to 2,000 users, the response latency peaks at 8 seconds, as documented in Figure 3.1c. The trend culminates for the 5,000 user scenario, where the response times for Japan, Canada, and Northern Europe reach 20, 40, and 120 seconds respectively, as shown in Figure 3.1e. Notably, this represents an exponential increase in response time for Northern Europe, and a linear scale-up by factors of 16 for Canada and 8 for Japan, underscoring the varying scalability across regions.

A detailed microscopic analysis, particularly highlighted in Figure 3.1a, reveals the dynamics of response time variations for Japan and Canada. Initially, at the one-minute mark-coinciding with the influx of traffic from Northern Europe-the slope of the response time increases from approximately 0.4166 seconds per minute to 0.7 seconds per minute by the two-minute mark. This increment signifies a notable escalation in the response burden on the system, attributable to the compounded load.

Furthermore, geographical distance manifests as a critical determinant of performance, especially at higher request loads. The 5000-user scenario starkly illustrates the pronounced impact of distance, with Australian clients proximal to the server experiencing the lowest initial response times. However, these times are interspersed with sporadic peaks suggesting potential volatility in infrastructural resilience or network stability at high loads. Conversely, clients in Northern Europe, Canada, and Japan encounter a gradual, more predictable ascension in response times.

The critical load threshold is vividly demonstrated by the service's response for 5,000 users, revealing the system's operational limits. It becomes evident that while the service sustains up to 2,000 users with nominal distance-related degradation, the performance deterioration at 5,000 users is marked and indicative of the infrastructural constraints. Such observations underscore the imperative for strategic server placement and load distribution to optimize the cloud-based service delivery, ensuring efficient scalability and robust user experience across global client regions.

RESTful Response Time

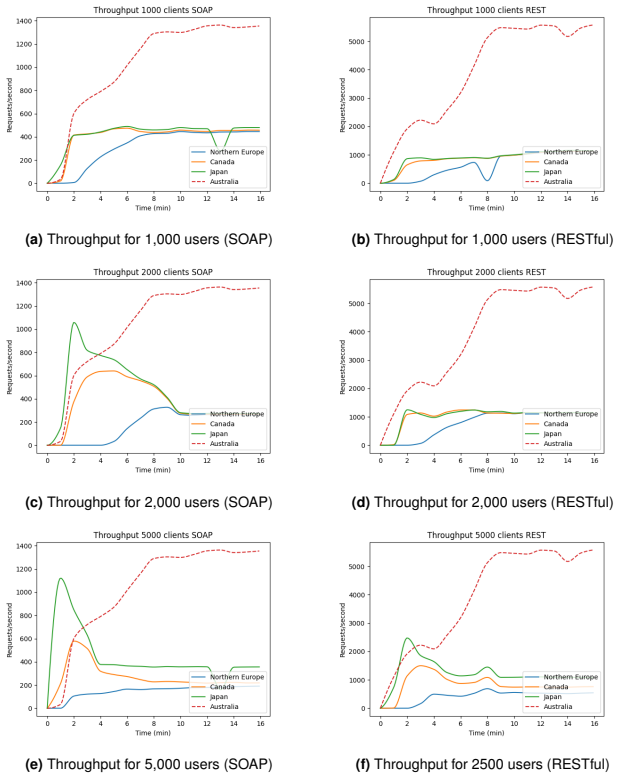
The investigation into RESTful service response times reveals discernible patterns that echo the influence of geographical proximity and concurrent user requests, similar to the observations made in SOAP services. Notably, an analysis across varying load scenarios underscores Japan's geographical leverage over Canada, and, by extension, Canada's over Northern Europe, as particularly elucidated in Figure 3.1f.

Distinctively, the RESTful architecture showcases a nuanced response behavior under high user loads. While SOAP analysis (3.1.1) highlighted exponential response time increases for Northern Europe in the 5,000 user scenario (Figure 3.1e), RESTful services exhibit a less pronounced disparity. The increment rates for Northern Europe in the RESTful context closely align with those of Japan and Canada, albeit with slight disadvantages, indicating a more homogeneous scalability across regions.

Further examination of user request increments over time reveals a parallel with SOAP findings, with a notable distinction in the impact of user load and geographical distance on RESTful services. At a 5000-user load, where SOAP response latency observed significant escalations, RESTful services maintained more stable latencies: 1.6 seconds for a 1000-user load (Figure 3.1b), 2 seconds for a 2000-user load (Figure 3.1d), and 3.2 (Figure 3.1f) seconds for a 5000-user load. This data implies a relative mitigation of load and geographical influences in RESTful services, suggesting inherent efficiencies in handling concurrent requests and geographic dispersion.

3.1.2 Throughput

Figure 3.2 shows three distinct sets of loading tests conducted on the SOAP and RESTful service.



(a) Throughput for 1,000 users (SOAP)

(b) Throughput for 1,000 users (RESTful)

(c) Throughput for 2,000 users (SOAP)

(d) Throughput for 2,000 users (RESTful)

(e) Throughput for 5,000 users (SOAP)

(f) Throughput for 2500 users (RESTful)

Figure 3.2: SOAP Throughput for different numbers of concurrent users

SOAP Throughput

According to 3.2a, despite the concurrent onset of testing by client terminals in Japan and Canada, the throughput increasing rate observed for Japan was more rapid, which can be attributed to the server's location in Australia, rendering the Japanese clients more closely situated and thereby facilitating quicker throughput acceleration. However, post the initial two minutes, notwithstanding the non-termination of ramp-up intervals for Japan and Canada, a stabilization of throughput at approximately 400 was witnessed. This stabilization elucidates that under the simultaneous request influx from three disparate client bases, the throughput was getting close to its bottleneck. Concurrently, an observation was made regarding the throughput increment rate for Northern Europe, which began its ascent at a conspicuously slower pace post the two-minute mark, owing to its maximal geographic remoteness from the server, consequently exhibiting the slowest rate of increase. Subsequently, around the seventh minute, a simultaneous and continuous request emission was initiated by all clients across the three regions, leading to a stabilization of throughput for each scenario at around 400.

In case of scenario 3.2c, it was observed that Japan's throughput peaked at the two-minute mark; however, due to the commencement of requests from Canadian clients, a rapid decline in Japan's throughput was evident after the second minute. Approximately at the fourth minute, clients from Northern Europe also initiated their requests, elucidating the rationale behind the second rapid decline in throughput for Japan and Canada around the fifth minute. Subsequently, the throughput for all three regions gradually stabilized at an approximate value of 300, which indicates that the server's maximum sustainable throughput is approximately 900, signifying a noticeable reduction in comparison to the previous scenario 3.2a.

In scenario 3.2e, it was observed that, with 5,000 clients in each location, the rate of throughput growth for Japan significantly outpaced that of Canada, due to its closer proximity to the server. However, as the volume of requests increased, a decline in Japan's throughput was also noted. By the second minute, as clients from Northern Europe began submitting requests, a decline in Canadian throughput ensued as well, eventually leading to a gradual stabilization of throughput for Japan, Canada, and Northern Europe at approximately 400, 200, and 190, respectively. With a total of 15,000 clients, the maximum sustainable throughput is approximately 790, marking a discernible decrease from the previous two scenarios.

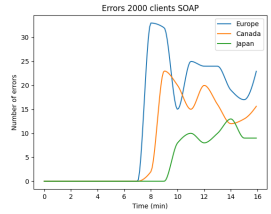
REST Throughput

Figures 3.2b, 3.2d and 3.2f and 3.2 represent the scenario for the tests on the RESTful service.

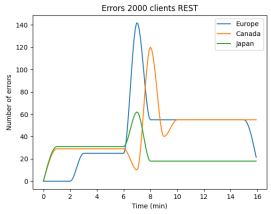
In scenario 3.2b, Canada and Japan initiated their tests concurrently, with Northern Europe starting two minutes later. Based on the 3.2b, it is observed that the throughput for Japan and Canada escalated at nearly identical rates. However, subsequent to the two-minute mark, that is, following the commencement of requests from Northern European clients, a decline in throughput for both Japan and Canada was noted, with the throughput of all three regions gradually stabilizing around 1000. This indicates that, when the total concurrent client requests amount to 3000, the server's maximum sustainable throughput is approximately 3000.

In scenario 3.2d, the circumstances largely mirror those of scenario 3.2b, with the server's maximum sustainable throughput ultimately stabilizing at approximately 3000 as well when the total concurrent client requests amount to 6000.

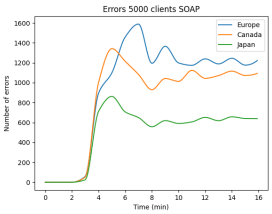
In scenario 3.2f, Canada and Japan initiated their testing concurrently, with Northern Europe starting two minutes later. It was observed that the rate of growth in throughput for Japan was notably higher than that of Canada, a phenomenon attributed to Japan's closer proximity to the server. Additionally, the throughput for both Japan and Canada began to decline following the initiation of



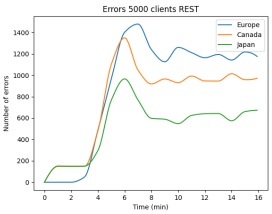
(a) Number of errors for 2,000 users (SOAP)



(b) Number of errors for 2,000 users (RESTful)



(c) Number of errors for 5,000 users (SOAP)



(d) Number of errors for 5,000 users (RESTful)

Figure 3.3: Number of errors for different numbers of concurrent users

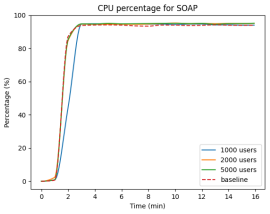
requests from Northern Europe. Ultimately, the throughput for Japan, Canada, and Northern Europe stabilized at approximately 1000, 750, and 550, respectively, reflecting the increasing distance of each region from the server. Concurrently, this scenario indicates that, with a total of 15,000 clients making requests, the server's maximum sustainable throughput is approximately 2300, representing a reduction when compared to the previous two scenarios.

3.1.3 Error

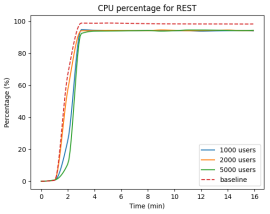
This section evaluates the system's performance and error rates from the client's viewpoint, as illustrated by error occurrence patterns depicted in Figure 3.3.

SOAP Error

Initially, the service exhibited stability, handling 1,000 simultaneous users without errors. However, during the test with 2,000 users, several challenges emerged. A noteworthy observation is that clients from North Europe, the region most distant from the server, encountered the highest number of errors compared to other client regions. Furthermore, there is a clear correlation between the number of concurrent users and the incidence of errors, with both the error count and rate escalating as user numbers increase. Specifically, the occurrence of **Java.net.SocketException** indicates the server is running out of available sockets for new TCP/IP connections. **Java.net.-SocketTimeoutException** and **Org.apache.http.conn.-ConnectTimeoutException** indicates the significant processing and connection establishment delays, suggesting potential server overload or network congestion.



(a) Server's percentage CPU - SOAP



(b) Server's percentage CPU - RESTful

Figure 3.4: Server's CPU Percentage for both SOAP and RESTful

Elevating the load to 5,000 users intensified these issues, introducing **Org.apache.http.NoHttpResponse-Exception**, which showed the server's failure to respond to requests, a clear sign of system overload.

This progression from connection issues to response failures emphasizes the system's scalability limitations under increased demand.

RESTful Error

In the evaluation of the RESTful service, the system encountered the challenges akin to those occured in the SOAP.

This pattern of errors underscores the critical need for advanced resource management strategies and scalability solutions to uphold service reliability and performance integrity in high-demand scenarios.

3.2 Server-Side Metrics

In the following section, the CPU's performance on the server side is analyzed to serves as a fundamental indicator of computational resource utilization within a system.

3.2.1 CPU Load

In the conducted load tests monitoring the server's CPU percentage, a pattern emerged in both SOAP and RESTful services. Despite varying client loads as seen in figure 3.4 a consistent behavior was observed. Specifically, within a minute of initiating the tests, due to the incoming of requests, the CPU utilization surged to more than 90 % across all scenarios and remained constant thereafter. This suggests a critical threshold being reached swiftly, indicating potential resource constraints under heavy loads.

4 DISCUSSION

The distinct difference between the SOAP and RESTful web services reveals critical insights into the perfor-

mance implications of these two HTTP-based service design paradigms, particularly in the context of cloud computing and distributed systems.

SOAP is designed for complex applications, offering high interoperability and security via XML. However, this results in larger payloads and increased processing overhead, which can lead to longer response times, particularly in geographically distributed applications. The structured protocol of SOAP, while beneficial for certain enterprise applications, imposes scalability and efficiency constraints when fast response times are crucial.

Conversely, RESTful services employ lightweight JSON formats and stateless operations, enabling them to deliver more consistent performance across different regions. Key to REST's efficiency are HTTP methods that facilitate effective caching and optimizations like CDNs and HTTP/2 or HTTP/3 protocols, which reduce latency and data transfer overhead. The stateless nature and lightweight communication protocol of RESTful services allow for better scalability and responsiveness, making them preferable for applications requiring global accessibility and high performance.

During several of our testing iterations, sometimes we observed a notable phenomenon where the CPU percentage of the B1s virtual machine exhibited a sharp increase, followed by an equally sudden decline, which result in a sudden decrease in throughput and an increase in response times. Such fluctuations are characteristic of Azure's B-series burstable VMs, which can build up credits when it is using less than its baseline. While these VMs can handle intermittent high loads by expending accumulated credits, performance is constrained once these credits are exhausted.

5 CONCLUSION

Our study presents an overall evaluation of SOAP and RESTful web services across various geographical locations and under different user load scenarios. The empirical analysis distinctly emphasizes the influence of geographical distance on response times, with services exhibiting varying latency degrees based on the proximity between client and server nodes. Regions closer to the server nodes experienced reduced response times, underscoring the critical need for strategic server placement to enhance performance globally.

Both the response time and throughput analysis highlights the challenges regarding scalability for both SOAP and RESTful services under elevated user loads. However, the impact of these challenges varied between the two service types. RESTful services demonstrated a relative resilience to load and geographical influences, maintaining more consistent latencies under comparable conditions. Conversely, SOAP services were significantly affected, especially in scenarios involving users from distant regions, indicating a pronounced sensitivity to load increases and geographical dispersion.

At last, the study identified a direct correlation between elevated error rates, increased CPU utilization, and user load increments, pointing to the operational boundaries of both SOAP and RESTful architectures. This correlation underscores the imperative for robust scalability solutions to mitigate system stress and maintain service reliability under high demand.

APPENDIX A : STATISTICAL RESULTS OF SOAP AND RESTFUL LOAD TEST

Table A.1: Statistical results of RESTful service

#User	Case	Requests	Error Count	Error Percentage	Response Time(s)	Throughput
100	baseline	3965718	0	0	0.03	4131
1000	North Europe	780674	0	0	1.27	813
1000	Canada	867135	0	0	1.14	903
1000	Japan	899296	0	0	1.12	937
1000	Aggregate	2547105	0	0	1.17	2653
2000	North Europe	921374	730	0.08	1.89	960
2000	Canada	1030305	729	0.07	1.65	1073
2000	Japan	1104702	410	0.04	1.74	1151
2000	Aggregate	3056381	1869	0.06	1.76	1061
5000	North Europe	487280	17469	3.59	2.97	508
5000	Canada	834235	13068	1.57	2.56	869
5000	Japan	1228344	8791	0.72	2.48	1280
5000	Aggregate	2549859	39328	1.54	2.67	885

Table A.2: Statistical results of SOAP service

#User	Case	Requests	Error Count	Error Percentage	Response Time(s)	Throughput
100	baseline	1030784	0	0.00	0.01	1073.73
1000	North Europe	348424	0	0.00	2.41	362.94
1000	Canada	404637	0	0.00	2.16	421.5
1000	Japan	415044	0	0.00	2.06	432.34
1000	Aggregate	1168105	0	0.00	2.21	405.59
2000	North Europe	235447	268	0.11	6.94	245.26
2000	Canada	386606	127	0.03	5.53	402.72
2000	Japan	456811	67	0.01	4.9	475.85
2000	Aggregate	1078864	462	0.04	5.79	375.61
5000	North Europe	157541	18507	11.75	94.39	164.11
5000	Canada	266181	14201	5.34	25.97	277.27
5000	Japan	425360	8530	2.01	9.67	443.08
5000	Aggregate	849082	41238	4.86	43.34	294.82