12th International Conference DaMSS
December 2 – 4, 2021, Druskininkai, Lithuania

Benchmarking of Communication Patterns in Microservice Architecture

Dalius Mažeika, Justas Kazanavičius
Vilnius Gediminas Technical University, Vilnius, Lithuania

Summary: Microservice architecture is applied for cloud-native applications to decompose it into small functional units. A microservices-based application is a distributed low coupling system running on multiple processes or services, and therefore proper communication patterns between microservices must be defined. Communication has a significant impact on the application's performance and must be adapted depending on the application architecture, exchange data, deployment approach, and service topology. The study aimed to perform benchmarks of different communication technologies and patterns between microservices and determine use cases for their application. The application-oriented criteria were introduced to evaluate communication patterns, and a microservice-based application was developed to perform experiments. Communication technologies based on remote procedure invocation and messaging protocols have been analyzed, and corresponding advantages and disadvantages have been identified.

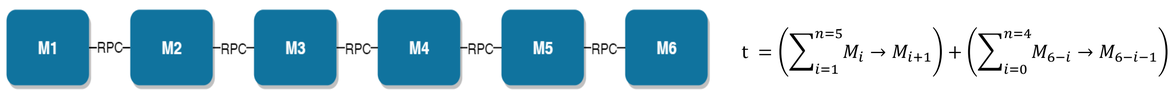
COMMUNICATION TECHNOLOGIES

Four technologies were chosen for analysis, i.e., HTTP (Rest API), messaging (Rabbit MQ), gRPC, and GraphQL. Rest API and Rabbit MQ represent synchronous and asynchronous communication styles, respectively, and are the most used technologies in a microservice architecture. GraphQL and gRPC have been chosen for the investigation because of the rapidly growing popularity.

RESEARCH METHODOLOGY

EXPERIMENT DESIGN

A set of six microservices were created and connected via RPC technique in a line topology to evaluate and compare communication technologies.



$$t = \left(\sum_{i=1}^{n=5} M_i \rightarrow M_{i+1}\right) + \left(\sum_{i=0}^{n=4} M_{6-i} \rightarrow M_{6-i-1}\right)$$

Fig. 1. The topology of microservices and formula used for the experiment.

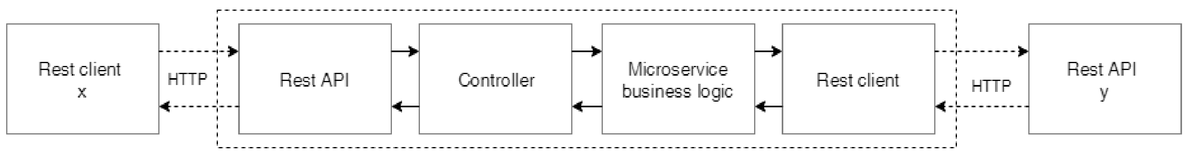
CRITERIA

These criteria were chosen to compare each communication technology: *Architecture*: to highlight the specific impact of each technology regarding app--lication architecture. *Topology*: technology impact to the topology of microservices. *Performance*: technology performance is measured and analyzed by time in milliseconds since the request was sent till the response was received. *Messages size*: to determine the potential technology impact on network load requests and response size in bytes were measured. *Memory size*: to evaluate how much memory is needed to run an application with each communication technology, application memory usage in bytes was measured. *Storage size*: to evaluate how much storage is needed to store an application with each communication technology, storage usage in bytes was measured. *Boot time*: application boot time in seconds was measured to determine how much time is needed to start the application. *Used applications and libraries*: to analyze the availability of the particular library.

TOOLS

All microservices were written using C# and .Net Framework 4.8 or .Net Core. All coding and testing were done using Microsoft Visual Studio IDE. All libra--ries used in the research were downloaded from NuGet gallery. Rest API was tested using Postman. All experiments were performed on a computer with the following specification: CPU - Core i5 6300U, memory – 12 GB RAM, storage – 512 GB SSD, and OS - Windows 10 Enterprise. All applications were run in computer, no external devices or networks were used.

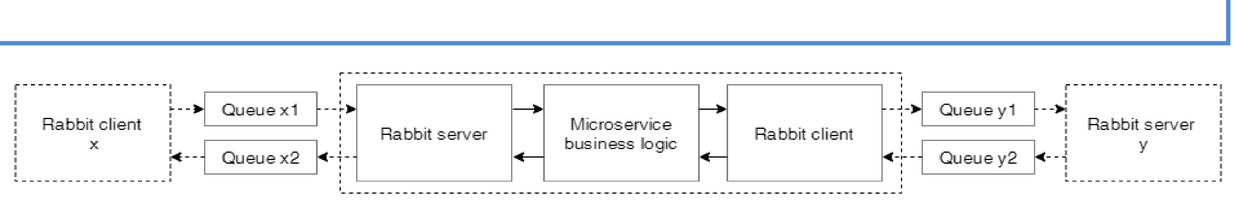
RESULTS OF EXPERIMENT



Number of Messages	Message size in characters			
	10 Char	1000 Char	100000 Char	1000000 Char
100	3.7 ms	3.8 ms	1.8 ms	7.9 ms
1000	3.3 ms	3.9 ms	1.6 ms	7.1 ms
10000	3.2 ms	3.7 ms	1.5 ms	7.0 ms

Metric	Result
Request/Response size	211 B/200 B (payload 27 B)
Microservice application size	976 KB (empty 24.1 KB)
Memory usage size	54 MB (empty 7 MB)
Boot time	2 seconds

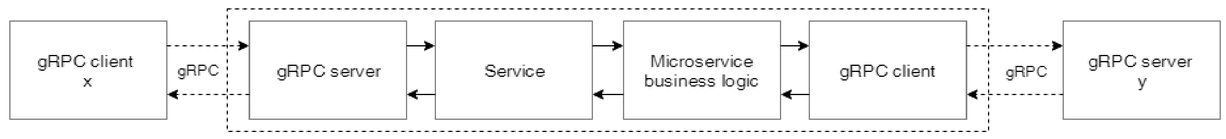
Fig. 2. The architecture and results of experiment of Rest API.



Number of Messages	Message size in characters			
	10 Char	1000 Char	100000 Char	1000000 Char
100	69.2 ms	68.9 ms	76.5 ms	140.2 ms
1000	71.2 ms	69.3 ms	73.0 ms	136.1 ms
10000	72.1 ms	68.8 ms	75.2 ms	137.3 ms

Metric	Results
Request/Response size	186 B/182 B (payload 7 B)
Microservice application size	2.68 MB (empty 24.1 KB)
Memory usage size	17 MB (empty 7 MB)
Boot time	5 seconds

Fig. 3. The architecture and results of experiment of RabbitMQ.



Number of Messages	Message size in characters			
	10 Char	1000 Char	100000 Char	1000000 Char
100	75.3 ms	74.1 ms	133.7 ms	713.1 ms
1000	73.4 ms	73.6 ms	131.2 ms	763.2 ms
10000	74.0 ms	72.3	131.2 ms	672.9 ms

Metric	Results
Request/Response size	211 B/200 B (payload 27 B)
Microservice application size	976 KB (empty 24.1 KB)
Memory usage size	54 MB (empty 7 MB)
Boot time	2 seconds

Fig. 4. The architecture and results of experiment of gRPC.

Number of Messages	Message size in characters			
	10 Char	1000 Char	100000 Char	1000000 Char
100	59.9 ms	60.9 ms	105.0 ms	362.7 ms
1000	67.5 ms	66.4 ms	99.9 ms	356.9 ms
10000	72.0 ms	56.1 ms	82.4 ms	298.3 ms

Metric	Results
Request/Response size	230 B/732 B (payload 39 B)
Microservice application size	334 KB (empty 24.1 KB)
Memory usage size	80 MB (empty 7 MB)
Boot time	8 seconds

Fig. 5. The architecture and results of experiment of GraphQL.

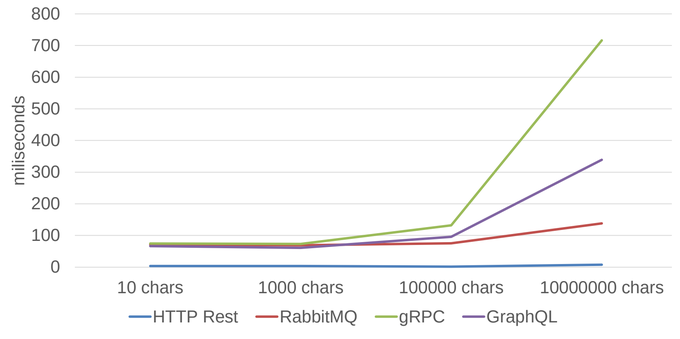


Fig. 6. Comparison of the performance tests.

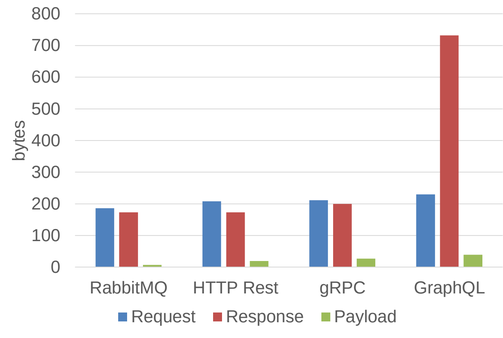


Fig. 7. Request/Response size.

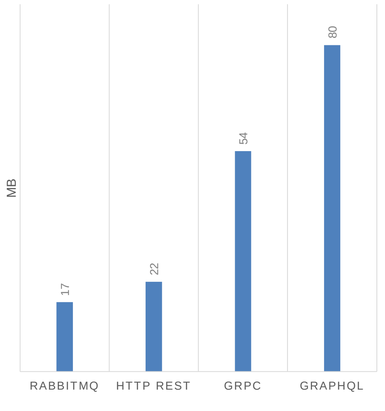


Fig. 8. Memory usage.

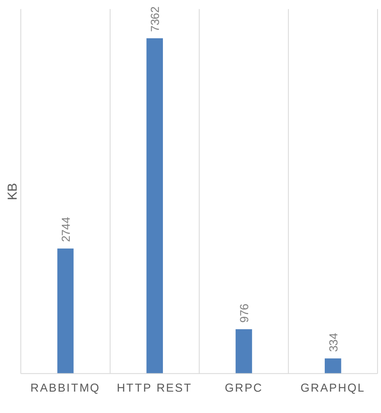


Fig. 9. Application size.

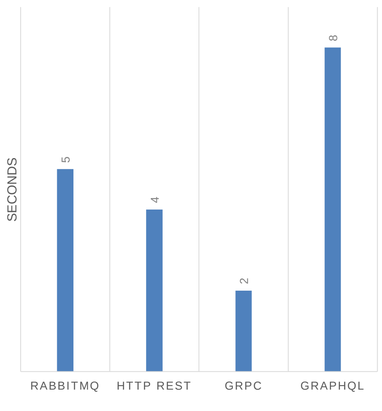


Fig. 10. Boot time.