ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
UNIVERSIDADE DE SÃO PAULO
Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

Relatório do EP - Remote Procedure Calls (RPC)

OBJETIVO
O objetivo deste Exercício Programa é avaliar e comparar o desempenho de implementações diferentes de mecanismos de chamadas de procedimentos remotos (mais conhecidas pelo acrônimo em inglês RPC, de Remote Procedure Call).

Como base para comparação, a dupla utilizou o gRPC, sistema de chamadas de procedimentos remotos de código livre mantido pela Cloud Native Computing Foundation. Além desse, foi adotado o JSON-RPC, protocolo popularmente conhecido que permite a comunicação entre aplicativos cliente-servidor através da transferência de dados estruturados no formato JSON.

1. INTRODUÇÃO
Esta seção contempla a visão geral dos sistemas analisados e descreve todos os recursos necessários para execução dos RPCs. Será abordado o ambiente que envolve a plataforma de hardware, sistema operacional, assim como as bibliotecas importadas.

1.1. Plataforma de hardware e S.O
Todos os experimentos foram testados nas máquinas da dupla. A primeira é um notebook Samsung Book X30 cuja configuração dispõe de um processador Intel Core i5 de 10ª geração com clock de 1.6 Ghz, 8GB de RAM e roda o Windows 11. A segunda, um notebook HP EliteBook 830 G8, conta com Intel Core i7 de 11ª geração com clock de 3Ghz, 16GB de RAM e roda o Windows 11. As duas foram configuradas da mesma maneira para rodar os protocolos.

1.2. Descrição dos RPCs
Os sistemas de RPC possuem particularidades e diferentes objetivos.

1.2.1 O gRPC
Este framework utiliza a Protocol Buffers (protobuf), uma linguagem de descrição de interface (IDL), como formato de serialização de dados. Toda a comunicação é feita pelo protocolo HTTP e é efetuada serialização no envio e deserialização no recebimento das mensagens.

O funcionamento do gRPC se inicia na definição de um arquivo *protobuf* e no desenvolvimento de código para o cliente e servidor em alguma linguagem de programação. O servidor então é iniciado e passa a aguardar mensagens do cliente.

1.2.2 O JSON-RPC
Este protocolo é popular por sua eficiência e facilidade de implementação, utilizando o formato JSON na transmissão dos dados.

O JSON-RPC se comunica de modo semelhante ao gRPC, com a diferença da estrutura de dados adotada. Um objeto JSON-RPC, conta com alguns campos que permitem todo o fluxo de informação.

No cliente, é indicada a versão do RPC, o nome do método invocado, os parâmetros e um *id* para a solicitação, sendo os dois últimos campos opcionais. Já para o servidor, é registrada a versão, a resposta da requisição, a descrição de erro e o id, onde os dois últimos campos também são opcionais.

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
UNIVERSIDADE DE SÃO PAULO
Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

1.3. Implementação da interface, métodos e bibliotecas

A implementação de ambos os RPCs foi feita na linguagem Python 3.10.8 e foram importados módulos nativos para auxiliar no manejo do código e na medição do tempo (*random*, *time*, *logging* e *string*), assim como bibliotecas matemáticas para cálculos e construção de gráficos (*numpy* 1.23.3 e *matplotlib* 3.7.1).

Todo o esquema de comunicação cliente-servidor é realizado da seguinte forma:

1. São definidas a porta e o IP adequados para a conexão no cliente e no servidor;

2. Um menu inicial é criado no cliente, de modo a guiar o usuário nos possíveis serviços, e o stub, recurso necessário para enviar as mensagens ao servidor;

3. Ainda no cliente, são feitos os métodos que atenderão a cada serviço. Cada um deles se encarrega de codificar os parâmetros e os dados para estrutura correspondente do RPC;

4. No servidor, são definidos métodos assíncronos que responderão as requisições de cada método do cliente. O método desencodifica a estrutura de dados, resolve a requisição, codifica a resposta e a envia para o cliente;

No gRPC, os parâmetros podem assumir tipos primitivos ou compostos, os quais são definidos no arquivo *.proto* do projeto. Durante a configuração, foi necessário importar as bibliotecas *grpcio* 1.54.2 e *protobuf* 3.19.5.

No JSON-RPC, os tipos são os próprios do Python. Houve a importação de bibliotecas para configuração e codificação (*jsonpickle* 3.0.1, *jsonrpcserver* 5.0.9 e *jsonrpclib* 0.2.1).

As operações escolhidas para análise foram seis métodos, com diferentes tipos de parâmetros e processamentos no servidor:

- ***Método 1:*** cliente envia nenhum argumento e servidor retorna *void*;
- ***Método 2:*** cliente envia dado de tipo *int64* (*long*) e servidor retorno seu dobro (*long*);
- ***Método 3***: cliente envia lista com oito dados de tipo *int64* (*long*) e servidor retorna a soma desses valores (*long*);
- ***Método 4:*** cliente envia dado de tipo string com tamanho n, servidor identifica o tamanho da string e calcula log2(n); assim, retorna outra string comunicando qual a próxima potência de dois desse valor (2^log2(n));
- ***Método 5:*** cliente envia um objeto de tipo carro (definido pela dupla) e servidor retorna outro objeto carro com o atributo cor alterado;
- ***Método 6:*** cliente envia duas matrizes quadradas de mesma dimesão e servidor retorna outra matriz, resultante da multiplicação das duas primeiras.

2. AVALIAÇÃO DO GRPC

Foram realizados vinte experimentos de chamadas para cada método mencionado anteriormente, e levaram-se em consideração dez testes no caso onde o cliente e servidor estavam na mesma máquina e outros dez para máquinas diferentes. A avaliação destrincha cada método e apresenta as diferenças de performance em ambos os cenários. Por conta do tempo adicional para estabelecer conexão com outra máquina, os experimentos de duas máquinas apresentam valores maiores de média e desvio padrão.

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
UNIVERSIDADE DE SÃO PAULO
Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

2.1. Método 1

Por ser um método simples que não envia e retorna dados, é um dos métodos de menor variação de tempo de resposta. Quando executado mais vezes, tende a ser o método de menor tempo de resposta, como será mostrado no capítulo quatro. Para a mesma máquina, o tempo de resposta médio achado foi de 1.78 milissegundos, com desvio padrão de 0.44 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 8.42 ms e 1.60 ms de desvio padrão.

Figura 1: Método 1 do gRPC rodando em uma máquina

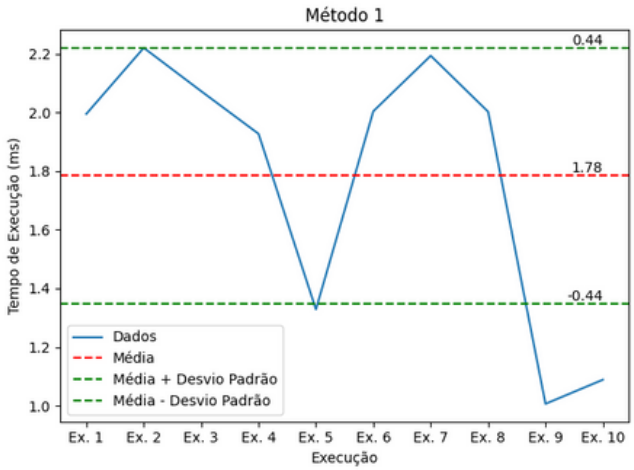
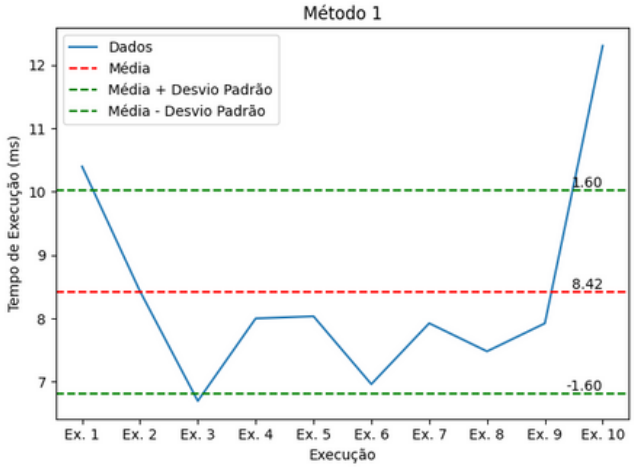


Figura 2: Método 1 do gRPC rodando em duas máquinas



2.2. Método 2

O resultado também se demonstrou baixo em comparação com os métodos seguintes. Para a mesma máquina, o tempo de resposta médio achado foi de 1.24 milissegundos, com desvio padrão de 0.64 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 8.43 ms e 3.10 ms de desvio padrão.

Figura 3: Método 2 do gRPC rodando em uma máquina

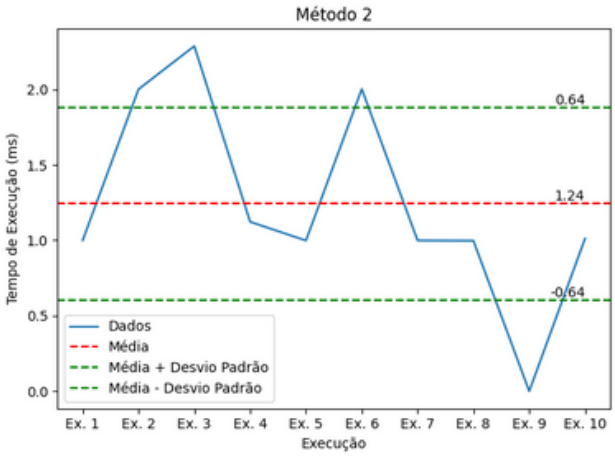
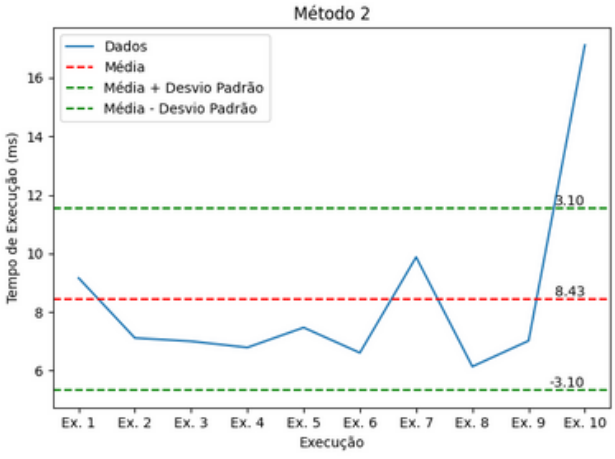


Figura 4: Método 2 do gRPC rodando em duas máquinas



Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

2.3. Método 3

Mesmo com o envio de oito argumentos, os valores sofreram pouca alteração. Para a mesma máquina, o tempo de resposta médio achado foi de 1.28 milissegundos, com desvio padrão de 0.38 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 9.78 ms e 2.13 ms de desvio padrão.

Figura 5: Método 3 do gRPC rodando em uma máquina

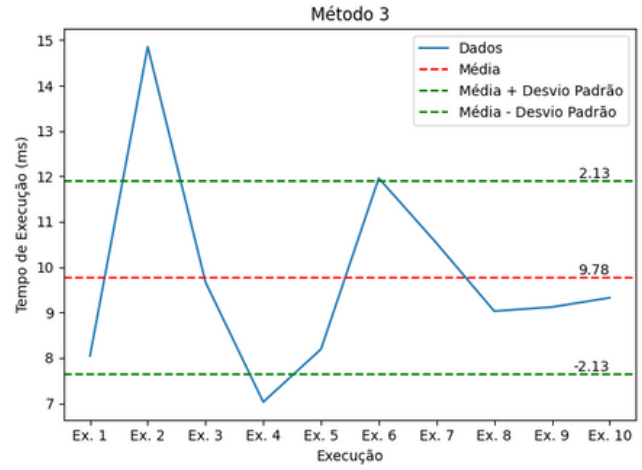
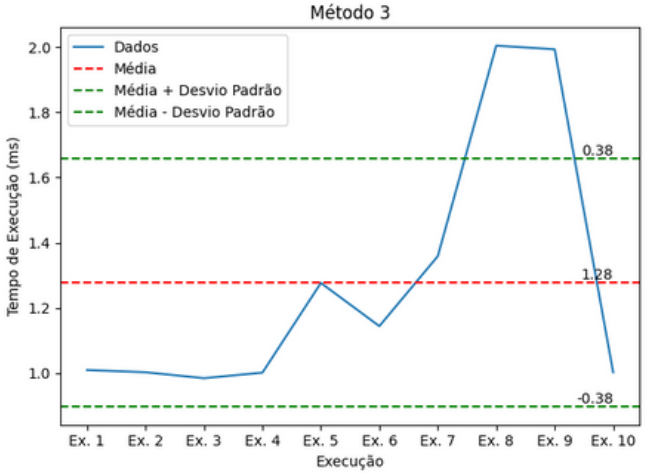


Figura 6: Método 3 do gRPC rodando em duas máquinas



2.4. Método 4

Por se tratar de manipulação de strings de tamanhos aleatórios, esse método obteve grande variação em comparação com os demais. Para a mesma máquina, o tempo de resposta médio achado foi de 3.18 milissegundos, com desvio padrão de 2.42 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 9.71 ms e 2.53 ms de desvio padrão.

Figura 7: Método 4 do gRPC rodando em uma máquina

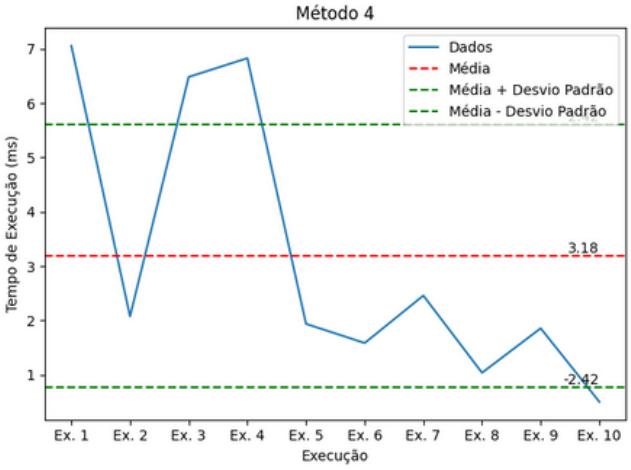
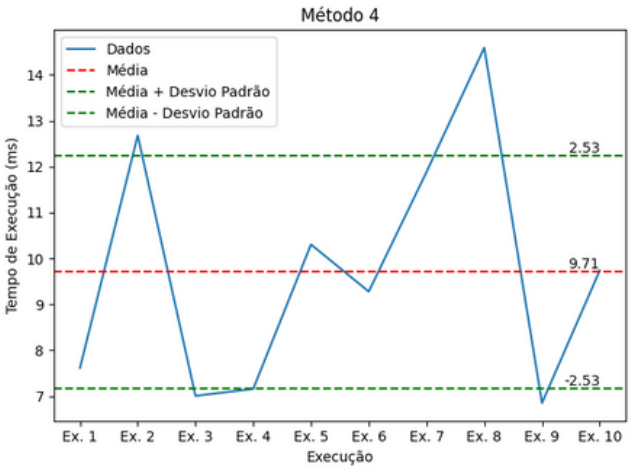


Figura 8: Método 4 do gRPC rodando em duas máquinas



Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

2.5. Método 5

Método que não contou com aleatoriedade ou processamento complexo, entregou valores semelhantes aos métodos mais simples. Para a mesma máquina, o tempo de resposta médio achado foi de 1.23 milissegundos, com desvio padrão de 0.57 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 8.78 ms e 2.44 ms de desvio padrão.

Figura 9: Método 5 do gRPC rodando em uma máquina

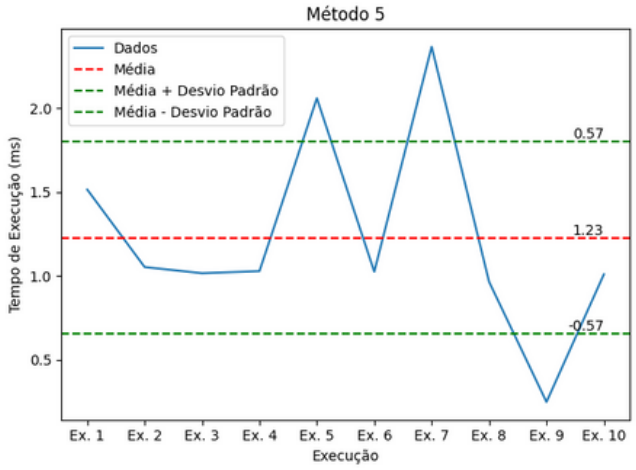
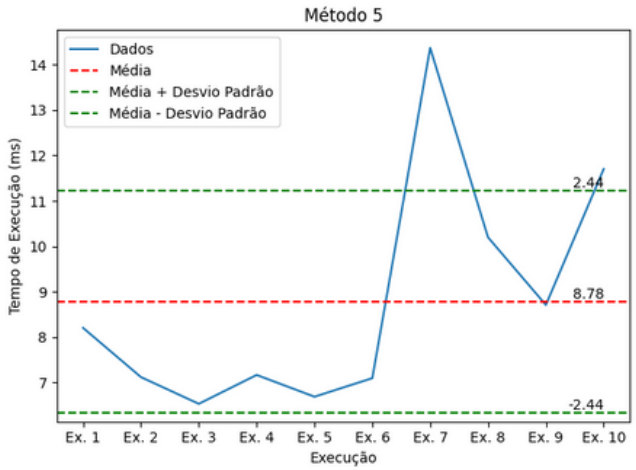


Figura 10: Método 5 do gRPC rodando em duas máquinas



2.6. Método 6

O cálculo da multiplicação de matrizes resultou em um tempo médio maior nesse experimento. Para a mesma máquina, o tempo de resposta médio achado foi de 3.26 milissegundos, com desvio padrão de 2.55 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 15.76 ms e 6.29 ms de desvio padrão.

Figura 11: Método 6 do gRPC rodando em uma máquina

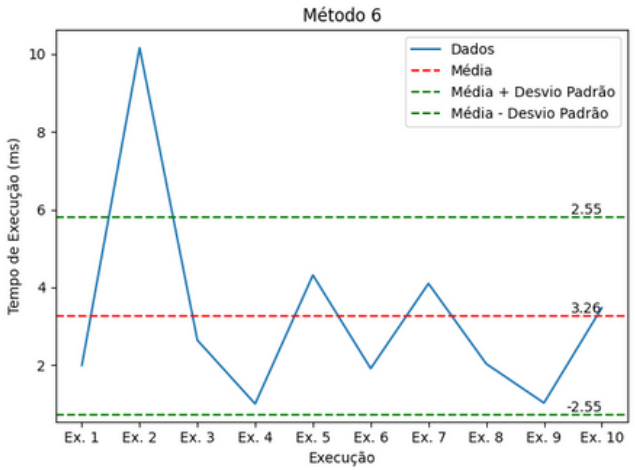
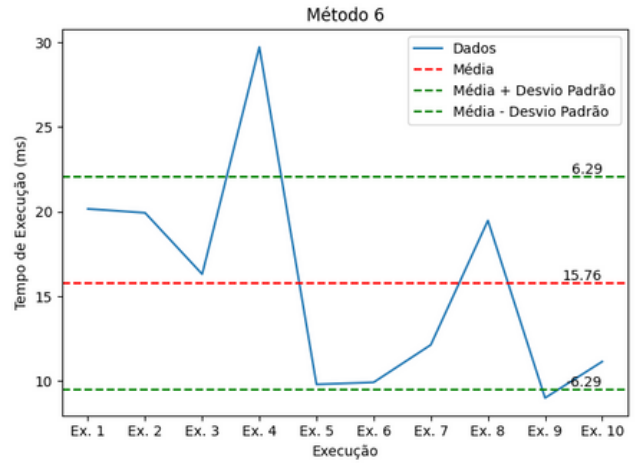


Figura 12: Método 6 do gRPC rodando em duas máquinas



Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

3. AVALIAÇÃO DO JSON-RPC

Assim como na avaliação do gRPC, foram realizados vinte experimentos de chamadas para cada método, dez para cada cenário. Uma particularidade ocorrida no JSON é o conflito de recursos durante a execução do cliente e servidor na mesma máquina. Essa adversidade, que é otimizada e não ocorre no gRPC, faz o tempo de execução em uma única máquina ser maior do que o tempo médio em duas máquinas, ainda que exista o tempo de conexão entre elas. Portanto, os graficos de máquina única devem possuir valores mais altos nos experimentos a seguir.

3.1. Método 1

Para a mesma máquina, o tempo de resposta médio encontrado foi de 2039.67 milissegundos, com desvio padrão de 10.45 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 11.84 ms e 2.65 ms de desvio padrão.

Figura 13: Método 1 do JSON-RPC rodando em uma máquina

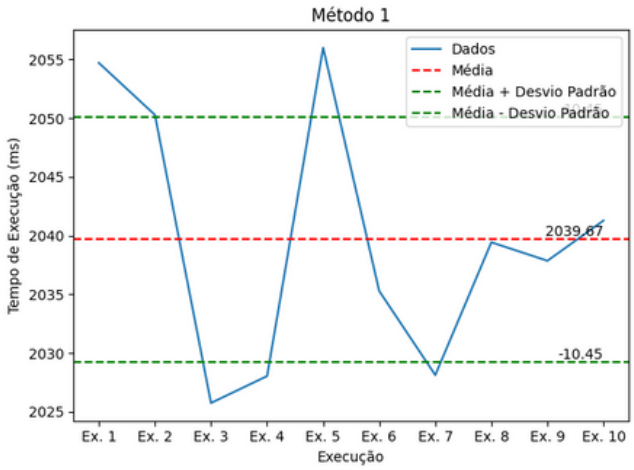
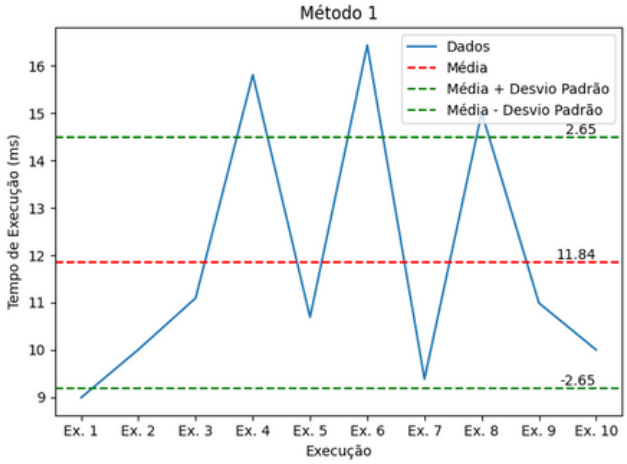


Figura 14: Método 1 do JSON-RPC rodando em duas máquinas



3.2. Método 2

Para a mesma máquina, o tempo de resposta médio encontrado foi de 2043.55 milissegundos, com desvio padrão de 10.22 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 17.18 ms e 3.10 ms de desvio padrão.

Figura 15: Método 2 do JSON-RPC rodando em uma máquina

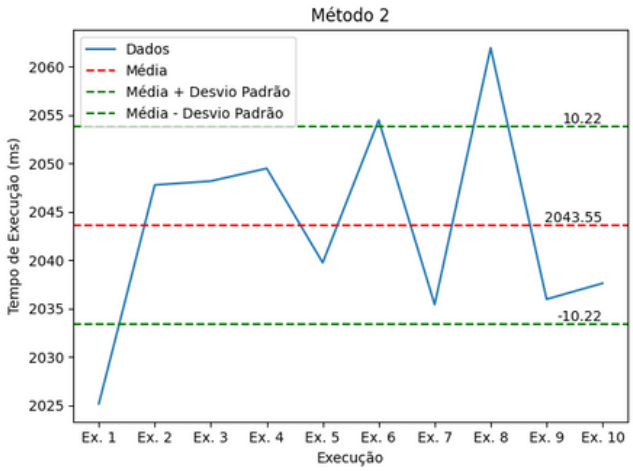
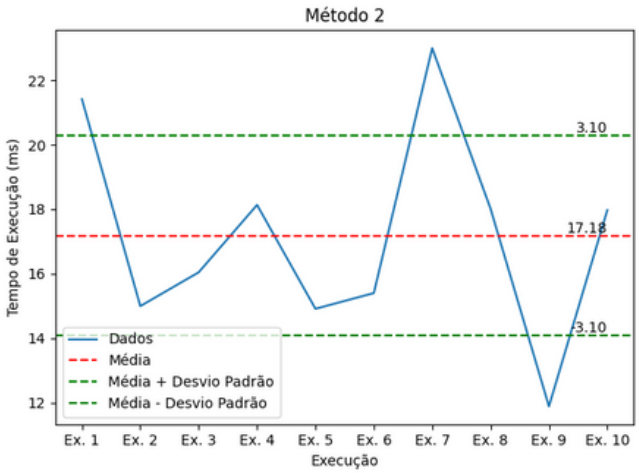


Figura 16: Método 2 do JSON-RPC rodando em duas máquinas



3.3. Método 3

Para a mesma máquina, o tempo de resposta médio encontrado foi de 2041.58 milissegundos, com desvio padrão de 8.85 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 14.44 ms e 3.42 ms de desvio padrão.

Figura 17: Método 3 do JSON-RPC rodando em uma máquina

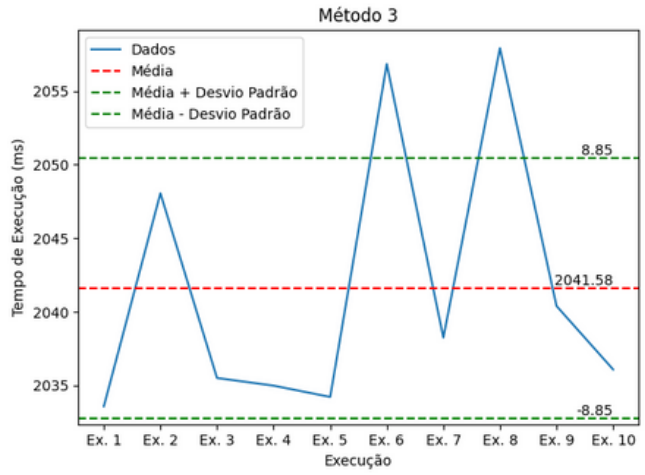
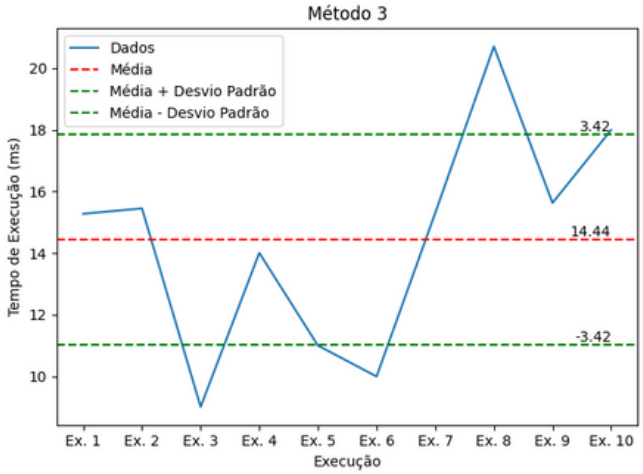


Figura 18: Método 3 do JSON-RPC rodando em duas máquinas



3.4. Método 4

Para a mesma máquina, o tempo de resposta médio encontrado foi de 2052.74 milissegundos, com desvio padrão de 7.12 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 15.46 ms e 5.31 ms de desvio padrão.

Figura 19 Método 4 do JSON-RPC rodando em uma máquina

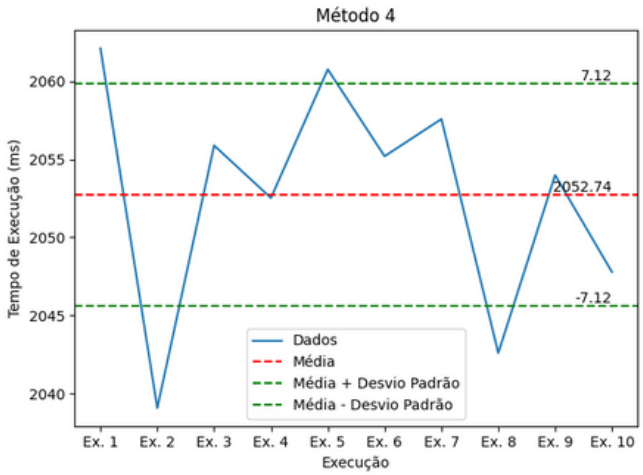


Figura 20: Método 4 do JSON-RPC rodando em duas máquinas

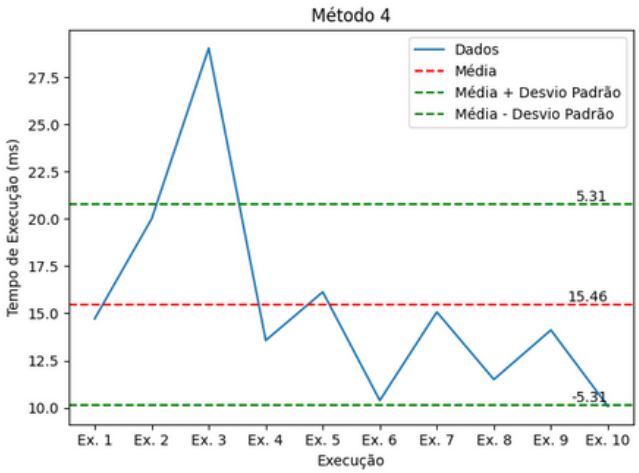
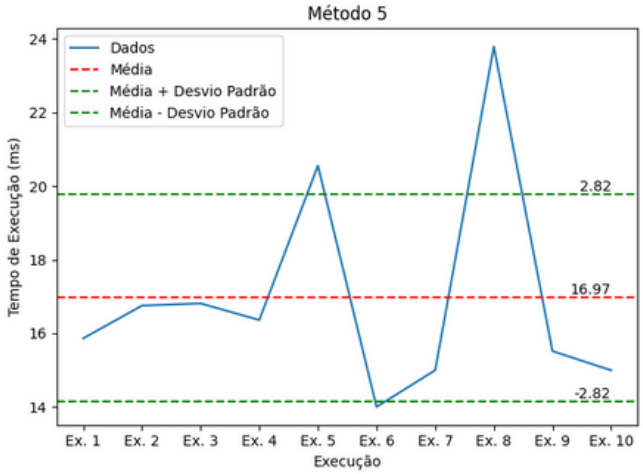


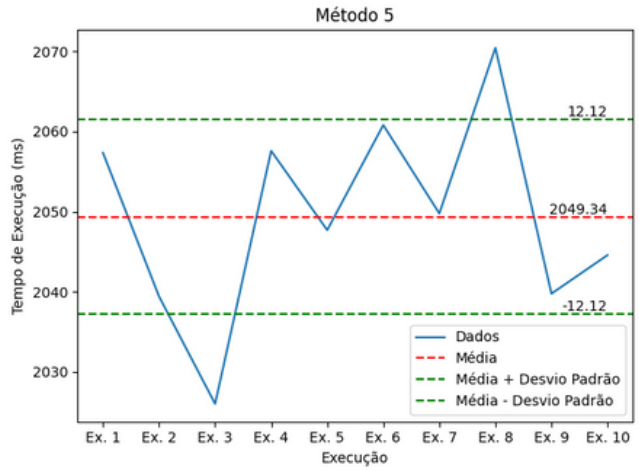
Figura 22: Método 5 do JSON-RPC rodando em duas máquinas



3.5. Método 5

Para a mesma máquina, o tempo de resposta médio encontrado foi de 2049.34 milissegundos, com desvio padrão de 12.12 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 16.97 ms e 2.82 ms de desvio padrão.

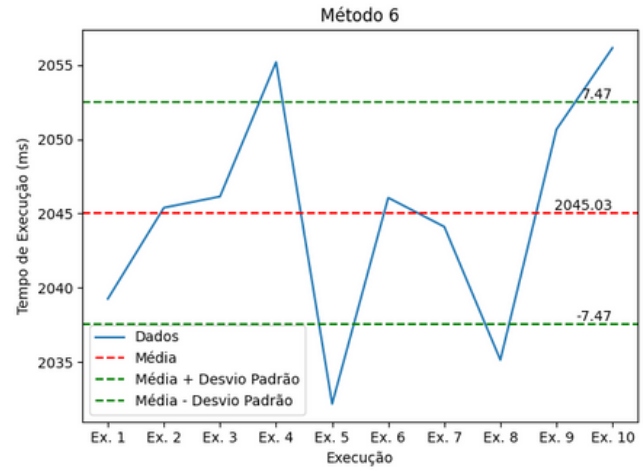
Figura 21: Método 5 do JSON-RPC rodando em uma máquina



3.6. Método 6

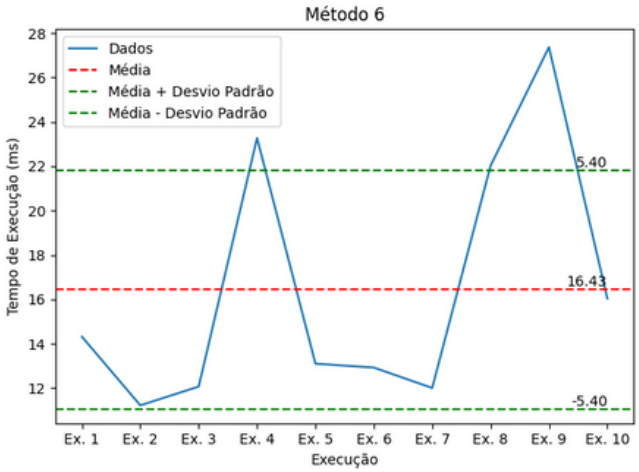
Para a mesma máquina, o tempo de resposta médio encontrado foi de 2045.03 milissegundos, com desvio padrão de 7.47 milissegundos. Já para o cenário de duas máquinas, o tempo médio foi de 16.43 ms e 5.40 ms de desvio padrão.

Figura 23: Método 6 do JSON-RPC rodando em uma máquina



Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

Figura 24: Método 6 do JSON-RPC rodando em duas máquinas



4. COMPARAÇÃO ENTRE GRPC E JSON-RPC

De forma a evidenciar as diferenças entre os sistemas, a dupla optou por comparar os resultados obtidos apenas no cenário de duas máquinas, além de ter aumentado o número de experimentos para trinta execuções, a fim de trabalhar com uma amostra de dados maior.

Serão utilizados dois tipos de gráficos para análise geral dos protocolos: um gráfico de barras, contendo as médias e desvios de cada método; e um gráfico de linhas, últil para representar as variações de tempo entre os métodos.

4.1. Comparação entre os tempos médios e desvios padrão

É possível observar nos gráficos a seguir a melhor performance do gRPC em relação ao seu concorrente. As médias de tempo e desvios são menores em todos os casos, e o ganho de performance, considerando as médias dos tempos médios de cada framework, é de cerca de 56,5%.

Calculando-se a variabilidade relativa do gRPC em relação ao JSON-RPC, encontrou-se o valor 0.28, o qual indica que o gRPC de fato apresenta variabilidade relativa menor que seu competidor, ou seja, há menor dispersão dos dados no primeiro protocolo, como será apresentado nos gráficos de linhas.

Um dos motivos para a vantagem do gRPC está na serialização binária desempenhada pelo Protocol Buffer e pela utilização do protocolo HTTP/2, o qual possui recursos avançados de compressão de cabeçalhos e multiplexação.

Figura 25: Tempos médios de execução e desvios no gRPC

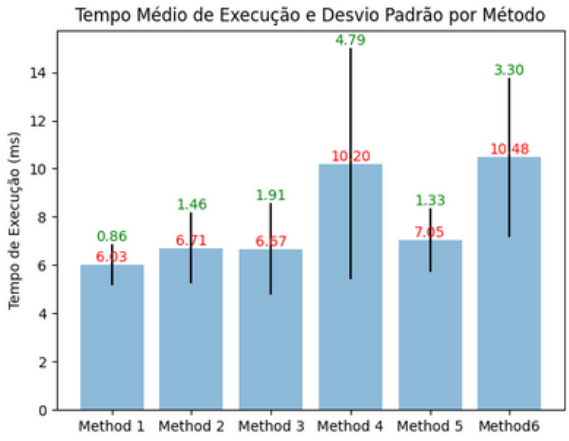
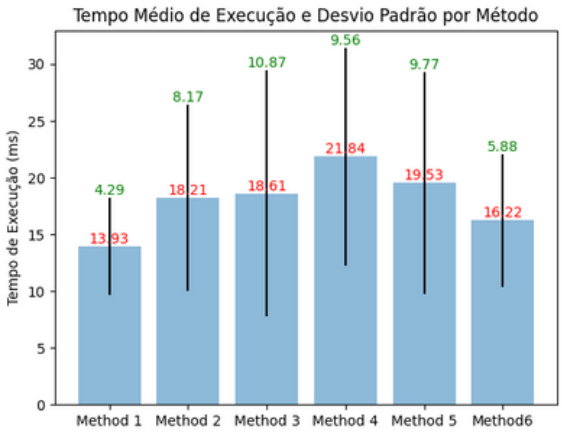


Figura 26: Tempos médios de execução e desvios no JSON-RPC



Lucas S. S. Gigante (12691898) e Pietro Zalla (12717606)

4.2. Comparação gráfica entre as variações de tempo

Como mencionado anteriormente, o gRPC apresenta menor dispersão de dados, e isto pode ser visualizado nos gráficos de linha abaixo.

No gráfico do gRPC, é possível perceber flutuações e valores extremos nos métodos quatro e seis, justamente os que trabalham com aleatoriedade de tamanho da entrada e cálculos matemáticos, então é esperada a alta dispersão.

Por outro lado, o gráfico do JSON-RPC apresenta inconstâncias até mesmo em métodos mais simples, que exigem pouco processamento, o que evidencia a reduzida agilidade do protocolo como um todo.

Figura 27: Representação das flutuações de tempos de execução por método no gRPC

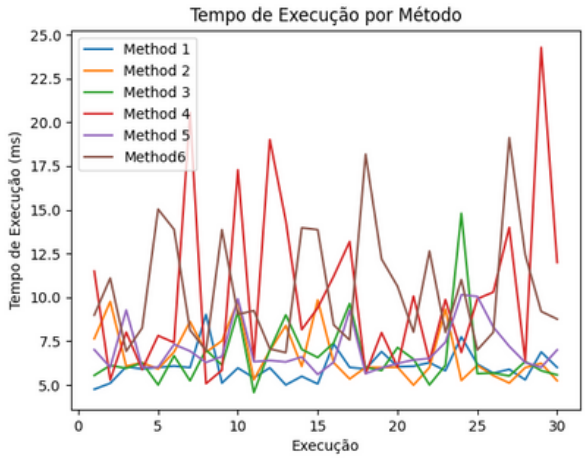
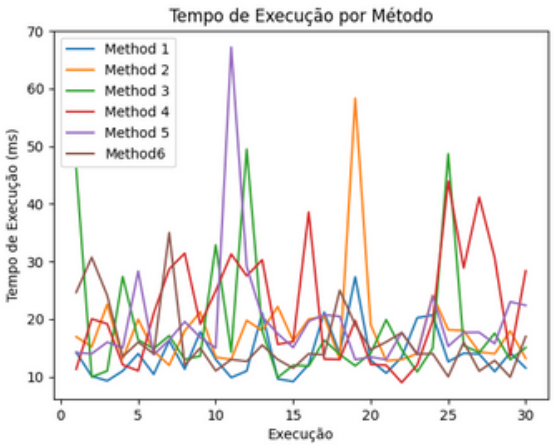


Figura 28: Representação das flutuações de tempos de execução por método no JSON-RPC



CONCLUSÃO

O JSON-RPC é um protocolo mais simples, ao passo que define poucos comandos e tipos de dados, o que torna seu escopo mais limitado e pouco flexível quando comparado a outros protocolos de chamada remota. Além disso, a manipulação de strings é outro fator que contribui para a menor eficiência do sistema.

Esse framework é uma boa escolha para serviços como o envio de notificações ao servidor e em casos da execução de múltiplas chamadas sem ordem específica. Nesse sentido, apresenta boa compreensão humana e é mais fácil de implementar.

Em relação ao gRPC, há maior complexidade no sistema, de maneira a oferecer flexibilidade ao desenvolvedor para implementar os métodos que preferir. A sua eficiência provém, principalmente, do recurso de serialização própria e do uso do HTTP/2, facilitando a definição de serviços, stubs e a manutenção em aplicações de larga escala.