UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE INFORMÁTICA E MATEMATICA APLICADA
Programação Distribuída

Descrição do Primeiro Trabalho

Objetivo

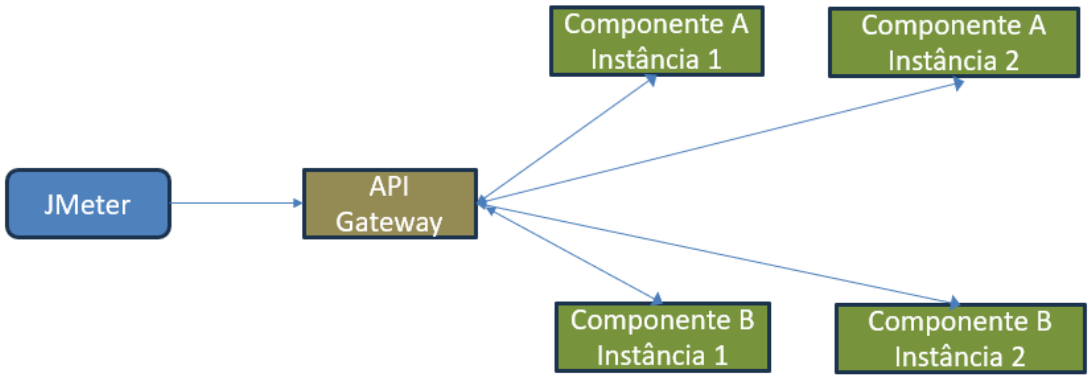
Desenvolver um sistema distribuído tolerante a falhas de rede/componentes, implementado por meio de um ou mais Padrões de Sistemas Distribuídos descritos no livro "Patterns of Distributed Systems" (Addison-Wesley Signature Series, 2024), de Unmesh Joshi.

Requisitos

1. Deve-se implementar uma aplicação simples que esteja relacionada com o Padrão de Sistema Distribuído escolhido. Por exemplo:
    - Leader and Followers: pode-se implementar um **simples e hipotético** banco de dados distribuído com replicação de dados
    - Generation Clock: pode-se implementar um **simples e hipotético** Sistemas de Controle de Versão (Git) ou Whatsapp.
    - Etc.
2. Desenvolvimento Individual
    - Cada aluno(a) deve implementar a plataforma de middleware de forma individual.
    - É expressamente proibido o uso ou a cópia de soluções desenvolvidas por outros(as) alunos(as) em semestres anteriores.
    - É expressamente proibido o uso de modelos de linguagem (LLMs), como ChatGPT, Copilot, ou similares, para a geração parcial ou total do código da aplicação.
    - Em qualquer um dos casos citados acima, caso seja identificada a infração, o(a) aluno(a) receberá nota zero neste trabalho.
3. Uso de Padrões de Projeto
    - O sistema deve ser construído seguindo os Padrões de Projeto (GoF) descritos no livro "Design Patterns: Elements of Reusable Object-Oriented Software" (1995), de Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides.

4. Componentes do Sistema
    - O sistema distribuído deve conter pelo menos três componentes.
    - Um dos componentes obrigatoriamente deve ser um **API Gateway**.
    - Os demais componentes do sistema devem ter no mínimo duas instâncias se forem *stateless.*
    - Se um dos componentes do sistema for *stateful*, pode ter apenas uma instância, a menos que o padrão escolhido implemente replicação de dados.
5. O Sistema distribuído proposto deve ser resiliente a falhas de rede/componentes
6. Protocolos de Comunicação Suportados
    - Transporte: UDP e TCP.
    - Aplicação: HTTP e gRPC.
    - Em um único projeto de código, todos os padrões devem ser suportados. A definição de qual protocolo utilizar deve ser feita no **startup** do componente.
7. Testes de Carga com JMeter
    - O aluno deve utilizar a ferramenta JMeter para realizar testes de carga do sistema.
    - No dia da apresentação do projeto, todos os testes devem estar preparados e configurados antecipadamente.
8. Deve-se investigar a capacidade do sistema, e conhecer seu **Knee capacity** e **Usable Capacity**

Abaixo, segue uma figura hipotética mostrando uma configuração mínima do sistema distribuído a ser apresentado.



Fluxo de Execução da Arquitetura

Nesta arquitetura, o **JMeter** será responsável por enviar requisições para o **API Gateway**. Essas requisições devem ser feitas utilizando os samplers **UDP, TCP, HTTP e gRPC** disponíveis no JMeter.

O **API Gateway** é um padrão arquitetural utilizado em arquiteturas de microserviços, que será estudado em mais detalhes na **terceira unidade** do curso. No entanto, para este trabalho, é essencial compreender que sua principal função na arquitetura é:

1. **Receber requisições dos clientes** (neste caso, o JMeter).
2. **Roteá-las** para os diferentes componentes internos da aplicação (Componentes A e B, conforme a figura).

Para desempenhar essa função, o **API Gateway deve ser capaz de descobrir dinamicamente os componentes internos**. Isso significa que, toda vez que um novo **Componente A ou B** for iniciado, ele deve enviar uma mensagem ao API Gateway contendo seu **endereço IP e porta**.

Portanto, ao iniciar um **Componente A ou B**, é **obrigatório informar o endereço IP/porta do API Gateway**, para que ele possa armazenar essa informação e saber para onde encaminhar as requisições.

Além disso, o API Gateway deve **monitorar a disponibilidade dos componentes internos** usando o padrão **Heartbeat**. Esse monitoramento permite que o API Gateway mantenha uma tabela atualizada com todos os componentes **ativos** e **prontos para receber requisições**. Dessa forma, quando o **JMeter enviar uma requisição**, o API Gateway garantirá que ela seja encaminhada **apenas para componentes disponíveis no momento**.

Por fim, para simplificar a arquitetura, **toda comunicação interna entre os Componentes A e B também deve ser feita via o API Gateway**.

Critérios de Avaliação:

Implementação dos protocolos (Vale 6,00):

UDP: 1,50

TCP com HTTP: 1,50

gRPC: 3,00

Implementação dos padrões (Vale 1,00)

Padrões de Sistemas Distribuídos

Execução com Tolerância a Falhas (Vale 3,0)

Todos os componentes da aplicação devem ser executados utilizando o protocolo definido abaixo no momento da apresentação.

1. No **JMeter**, deve-se configurar um número de usuários simultâneos **maior que 5 e menor que o Knee Capacity**.

2. Quando a aplicação estiver em funcionamento, o **Summary Report** deve indicar **zero erros**.
3. Durante a apresentação, será solicitado que algumas instâncias da aplicação sejam desligadas. Nesse momento, espera-se que a **taxa de erro aumente**.
4. Em seguida, será solicitado que novas instâncias do componente desligado sejam criadas. Quando isso ocorrer, a **taxa de erro deve diminuir**, demonstrando a recuperação do sistema.

O aluno deve ser capaz de **explicar qualquer parte do seu código** durante a apresentação. Problemas na execução da aplicação ou dificuldades em **explicar o código** afetarão a pontuação, não apenas neste critério de avaliação, mas também nos demais critérios do projeto.

Datas para Apresentação

As apresentações deste trabalho ocorrerão no período de 16/10/2025 a 28/10/2025. A ordem de apresentação será definida por sorteio. O(a) aluno(a) deverá estar devidamente preparado(a) para realizar sua apresentação quando for chamado(a). Caso não o faça, perderá a vez, podendo apresentar somente ao final, se houver tempo disponível. Na inexistência de tempo hábil, será atribuída nota zero para este trabalho.