Middleware Distribuido

Arquitetura Cliente-Servidor

João Victor Ribas Coelho
Luciano Costa Vianna Neto
Carine Sá de Morais Aquino
Gabriel Crispim Valentino de Siqueira

Proposta

Projeto:

Implementação de um middleware simples para aplicar os conceitos de sistemas distribuidos

Objetivo:

Demonstrar como um middleware abstrai a comunicação e realiza o gerenciamento de falhas em um ambiente distribuído, usando uma arquitetura cliente-servidor.

Linguagem:

Python

Motivação e Cenário

Problema:

Em sistemas distribuidos, a comunicação direta entre os clientes e servidores pode ser complexa e suscetível a falhas, como falhas de comunicação, localização de recursos e gerenciamento de estado.

Solução:

Implementar um middleware que abstraia essas complexidades e forneça uma comunicação robusta e escalável, aplicando os principais conceitos de sistemas distribuídos.

Conceitos Aplicados

RCP (Remote Procedure Call):

- O middleware usará RCP para permitir que o cliente invoque funções no servidor como se estivessem no mesmo processo.
- Implementação de gRCP ou XML-RCP para abstrair a comunicação.

Serviço de Descoberta:

- O middleware aplicará um serviço de descoberta, permitindo que o cliente se conecte ao servidor sem saber sua localização física.
- Implementação de DNS dinâmico ou uso de ferramentas como Consul para mapear servidores disponíveis

Fallback e Heartbeats:

- Implementação de fallback para redirecionar solicitações em caso de falha do servidor principal.
- Implementação de heartbeats para a monitoração contínua dos servidores com mensagens de “heartbeat” para garantir que estão ativos
- Lógica de retry e fallback para redirecionar requisições a um servidor secundário caso o principal falhe.

Proxy Reverso (Load Balancing):

- O middleware utilizará um proxy reverso para balanceamento de carga, distribuindo as requisições entre diferentes servidores disponíveis.
- Uso de ferramentas como Nginx ou HAProxy para distribuir o tráfego.

Demonstração de Funcionamento

Passo 1:

- O Cliente faz uma requisição simples, por exemplo “Hello”.

Passo 2:

- O middleware processa a requisição
- Abstrai a localização do servidor com o serviço de descoberta
- Faz a chamada via RCP ao servidor
- Monitora a saúde do servidor com heartbeats
- Se o servidor falhar, aplica o fallback para redirecionar a requisição a outro nó
- Utiliza o proxy reverso para distribuir as requisições entre múltiplos servidores

Passo 3:

- Servidor retorna a resposta da requisição com “World”.

Conclusão

Aplicação:

O middleware proporcionará uma comunicação mais eficiente e robusta entre cliente e servidor.

Prevenção:

Com o Fallback e heartbeats, o sistema será tolerante a falhas.

Escalabilidade:

Com o proxy reverso (Load balancing), o middleware será capaz de escalar para suportar mais clientes simultâneos.

Obrigado!

Alguma dúvida?