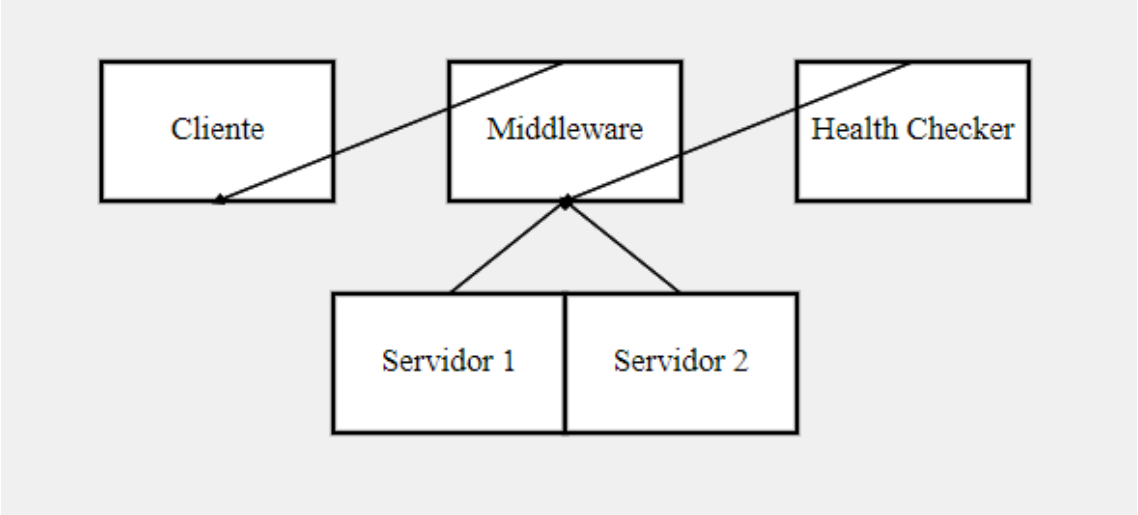
Grupo 7 - Fraga - Monitoramento de Servidores

Sistemas Distribuídos 2024-02
Jarison Vinho: jarisonvinho@discente.ufg.br
Mateus Torres: mateustorres@discente.ufg.br
Matheus Yosimura: matheusyosimura@discente.ufg.br

Descrição
Este sistema distribuído de monitoramento garante a saúde de seus servidores, fornecendo insights sobre o desempenho e alertas sobre eventos críticos. Ele coleta dados de vários servidores e os analisa para detectar problemas e gerar relatórios, mantendo sua infraestrutura funcionando perfeitamente.

Arquitetura do Sistema



O sistema adota uma arquitetura cliente-servidor com um componente intermediário (Middleware) e um monitor de saúde (Health Checker). Ele se caracteriza por:

Middleware: O Middleware atua como um ponto intermediador de comunicação, gerenciando o registro dos servidores e o encaminhamento das requisições dos clientes.

Monitoramento de Saúde: O Health Checker verifica periodicamente o status dos servidores e informa ao Middleware quais estão online, garantindo que os clientes se conectem apenas a servidores funcionais.

Escalabilidade A arquitetura permite a fácil adição de novos servidores sem impactar o cliente ou o Health Checker.

Regras de Negócio

Middleware:
- Recebe conexões de clientes.
- Mantém uma lista de servidores registrados (IP e porta).
- Quando solicitado pelo cliente, envia a lista de servidores.

- Aceita a seleção do cliente e cria uma ligação com o servidor selecionado.
- Encaminha as solicitações do cliente para o servidor.
- Retorna as respostas do servidor para o cliente.

Health Checker
- Verifica periodicamente o status dos servidores.
- Mantém uma conexão TCP persistente com o Middleware.
- Recebe do Middleware a lista de servidores registrados (IP e porta).
- Tenta conectar a cada servidor na lista.
- Usa timeout para evitar bloqueios.
- Determina o status online/offline de cada servidor.
- Envia mensagem de "ping" para verificação mais robusta.
- Envia ao Middleware a lista de servidores online.
- Lida com erros de conexão e evita interrupções.
- O Middleware usa as informações do Health Checker para fornecer aos clientes apenas servidores online.

Servidor:
- Inicia em uma porta livre.
- Anuncia seu endereço (IP e porta) para o middleware.
- Aguarda conexões de clientes (via middleware).
- Coleta informações de sistema (CPU, memória, disco, rede, uptime).
- Envia as informações coletadas de volta para o cliente (via middleware).

Cliente:
- Conecta-se ao middleware.
- Solicita e recebe a lista de servidores disponíveis.
- Escolhe um servidor da lista.
- Envia solicitações de status para o servidor (via middleware).
- Recebe e exibe as informações de status.
- Calcula e exibe médias de uso de recursos.
- Salva o histórico de uso de recursos em um arquivo.

Diagrama de Classe
1. Middleware:
Atributos:
- list_servers: Uma lista que armazena as informações dos servidores registrados (IP e porta).
- SERVER_LIST_IP: O endereço IP em que o middleware escuta por conexões de servidores para registro e de clientes para obter a lista de servidores.
- SERVER_LIST_PORT: A porta em que o middleware escuta por conexões de servidores para registro e de clientes para obter a lista de servidores.

Métodos:
- start_middleware(): Inicia o middleware, configura a escuta de conexões e possivelmente outras inicializações.
- handle_client(): Gerencia a interação com os clientes, incluindo o envio da lista de servidores e o encaminhamento de solicitações para o servidor escolhido.
- server_list(): Retorna a lista de servidores registrados para o cliente.

- client_server_select(): Recebe a escolha do cliente e estabelece a conexão com o servidor selecionado.

2. Server:
Atributos:
- host: O endereço IP do servidor.
- port: A porta em que o servidor escuta por conexões.
- running: Um indicador (booleano ou similar) que indica se o servidor está em execução.

Métodos:
- start_server(): Inicia o servidor, incluindo a escolha de uma porta livre e o anúncio para o middleware.
- handle_client(): Gerencia a comunicação com o cliente (via middleware), processa as solicitações e envia as respostas.
- announcement_server(): Anuncia o endereço (IP e porta) do servidor para o middleware para que ele possa ser adicionado à lista de servidores disponíveis.
- pick_free_port_number(): Tenta encontrar uma porta livre para o servidor escutar.
- port_test(): Testa se uma porta específica está disponível.

3. Client:
Atributos:
- cpu_sum: Armazena a soma dos valores de CPU para calcular a média.
- memory_sum: Armazena a soma dos valores de memória para calcular a média.
- ...: Outros atributos para armazenar somas e informações relevantes para o cliente (disco, rede, etc.).

Métodos:
- request_server_status(): Envia uma solicitação de status para o servidor selecionado (via middleware).
- start_connection(): Inicia a conexão com o middleware.
- save_history(): Salva o histórico de uso de recursos em um arquivo.
- load_history(): Carrega o histórico de uso de recursos de um arquivo.
- parse_status(): Analisa a resposta do servidor contendo as informações de status.

4. HealthChecker:
Atributos:
HEALTH_CHECKER_IP: Endereço IP do Health Checker.
HEALTH_CHECKER_PORT: Porta do Health Checker.

Métodos:
start_health_checker(): Inicia o Health Checker.
check_server_health(servers): Verifica a saúde dos servidores recebidos como parâmetro.

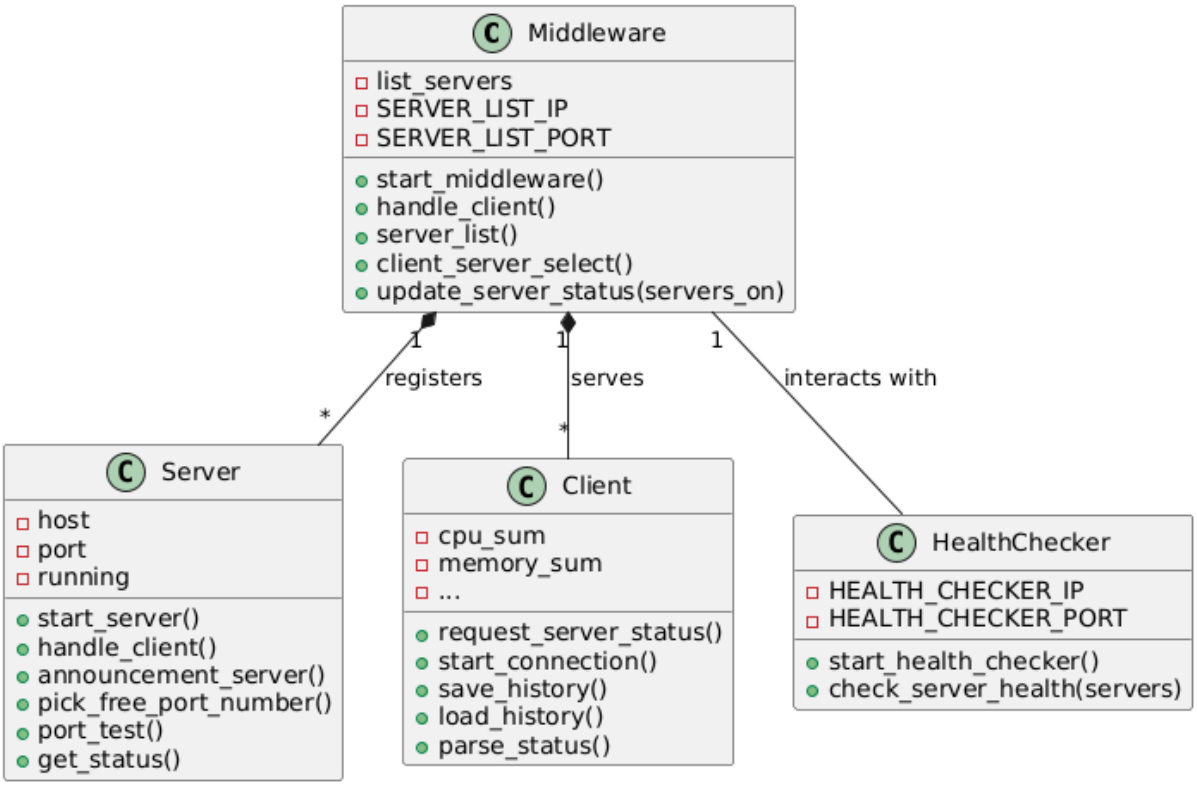


Diagrama de Sequência