GCC129 - Sistemas Distribuídos

Relatório Técnico 2

Modelos de Arquitetura para Aplicações Distribuídas

Nome: Pedro Antônio de Souza
Turma: 14A

1. Introdução

Modelo de arquitetura é uma caracterização conceitual das funcionalidades e propriedades de um sistema e seus componentes.

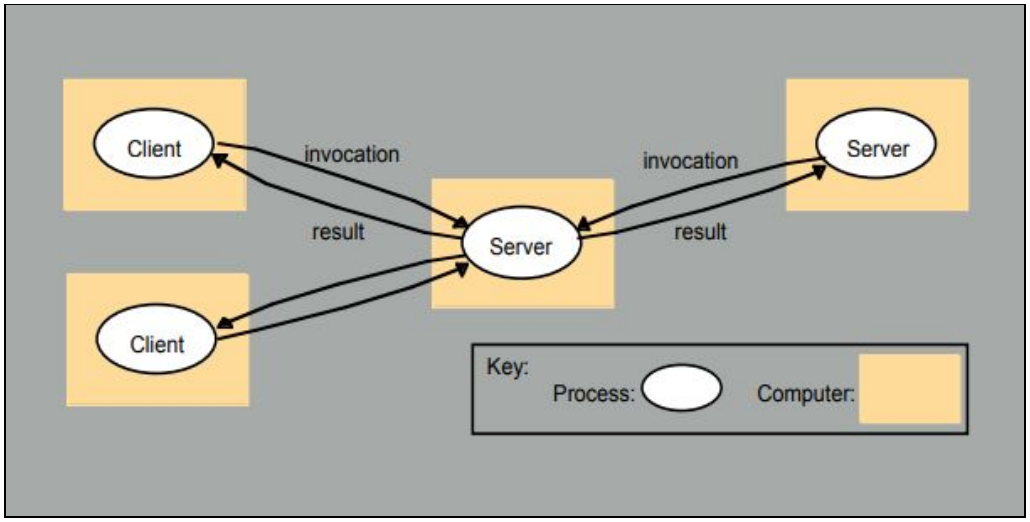
Sistemas distribuídos são compostos por vários processos independentes espalhados em várias máquinas que são interligadas em redes de computadores. Para se obter um desempenho ideal, é importante organizar e gerenciar esses sistemas de maneira satisfatória. Assim, a arquitetura de um sistema distribuído deve definir quais são seus componentes essenciais para que esses sejam dispostos nas máquinas da melhor maneira. Além disso, são descritas as características, funções e importância de cada componente. Também, é fundamental avaliar como os componentes interagem entre si e definir seus papéis e padrões de comunicação.

Os requisitos não-funcionais também são analisados no projeto de arquitetura. Nas aplicações distribuídas, deve-se determinar quais componentes devem ser replicados, o impacto da falha de cada componente, quais estratégias de segurança serão utilizadas, etc.

2. Cliente-Servidor

O modelo **cliente-servidor** é uma arquitetura centralizada que utiliza um protocolo de requisição-resposta. Nesse modelo, existem dois tipos de componentes: os servidores que são encarregados da provisão de informações e os clientes que fazem requisições aos servidores a fim de obter as informações. Contudo, como ilustrado na Figura 1, é possível que servidores atuem como clientes buscando informações em outros servidores.

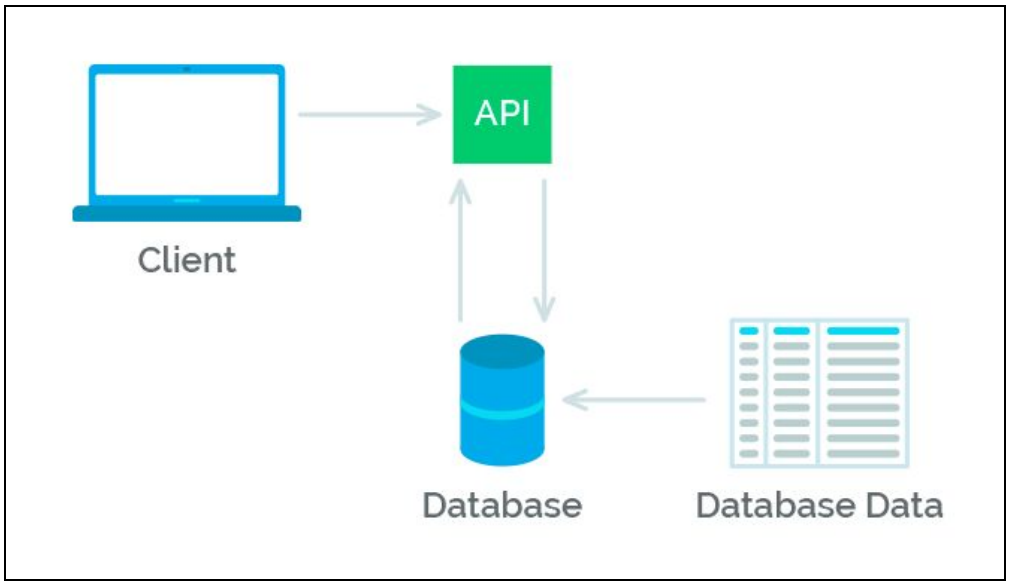
Figura 1 – Representação de arquitetura cliente-servidor.



Essa arquitetura possui a desvantagem do servidor centralizado ser o gargalo, já que ele deve receber as requisições de todos os clientes. Para garantir desempenho, faz-se o balanceamento de carga replicando o servidor e implementando um intermediador para receber as requisições e encaminhá-las ao servidor ideal.

Como consequência do uso do protocolo HTTP pela rede mundial de computadores, a grande maioria das aplicações atuais seguem esse modelo. Outro exemplo são as APIs REST na qual os clientes requisitam as informações utilizando, também, o protocolo HTTP. A API ora faz o papel de servidor provendo informações aos clientes, ora faz o papel de cliente requisitando dados do banco de dados como ilustra a Figura 2.

Figura 2 – Representação de uma API REST.



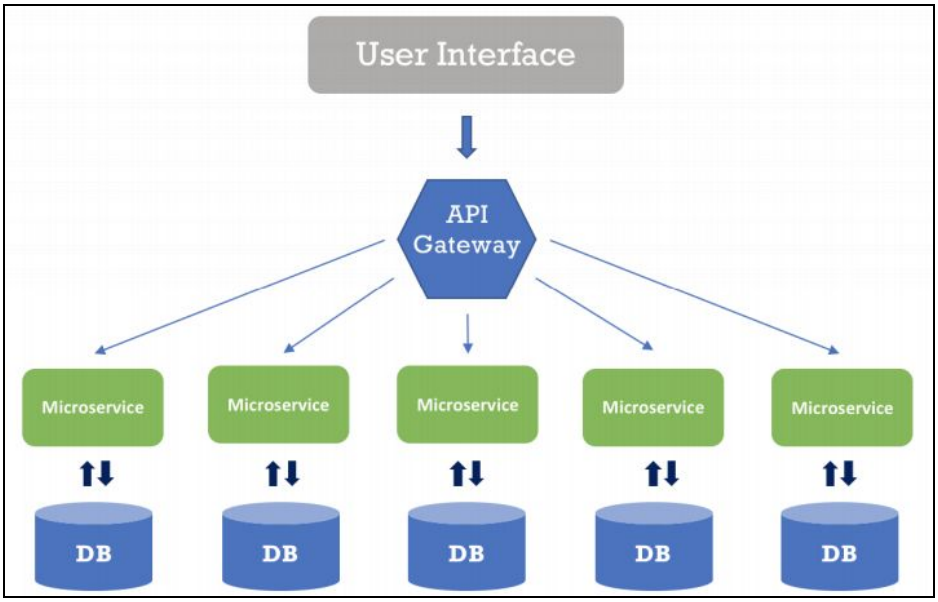
3. Microsserviços

No modelo de **microsserviços** ainda há componentes clientes que fazem requisições a componentes servidores, porém as regras de negócio nesse modelo são fragmentadas em diversos microsserviços. Assim, também é possível dividir o banco de dados em vários outros de acordo com o escopo dos dados armazenados.

Com esse desmembramento, há a descentralização do servidor, eliminando o gargalo existente no modelo cliente-servidor. Possivelmente, ocorre um ganho de disponibilidade do sistema, já que não há mais um ponto único de falha.

Do ponto de vista da manutenção, a independência entre os serviços permite que apenas partes da lógica de negócio sejam replicadas e, também, que um microsserviço seja reutilizado em outros projetos. Porém, como as requisições do cliente devem ser direcionadas aos microsserviços corretos, o desmembramento do servidor implica na fragmentação dos end-points. Para contornar esse problema, é necessário criar um API Gateway. Essa API é responsável por receber as requisições do cliente e redirecioná-la para o microsserviço correto como ilustrado na Figura 3.

Figura 3 – Representação do uso de API Gateway.



Em aplicativos de banco, pode ser interessante o uso de microsserviços para segregar serviços de autenticação do usuário e serviços responsáveis pelas transações financeiras. Dessa forma, é possível que o microsserviço de transações fique inativo até que o usuário vá até um caixa-eletrônico e habilite seu celular para realizar essa atividade.

4. Peer-to-Peer (P2P)

O modelo **peer-to-peer** é uma arquitetura descentralizada na qual todos os componentes do sistema se comportam igualmente, não sendo distinguidos entre clientes ou servidores. Cada componente se comporta, ao mesmo tempo, como cliente e servidor, isto é, qualquer componente pode fazer e receber requisições de qualquer outro componente. Nesse modelo, há uma distribuição horizontal dos componentes, não havendo nenhuma hierarquia entre eles.

Já que cada novo nó que se integra à rede para consumir também irá servir, a capacidade do sistema não é afetada de forma linear com o aumento de componentes consumidores. Apesar da grande escalabilidade, sistemas desse tipo são difíceis de administrar.

O protocolo BitTorrent é um exemplo de sistema P2P utilizado para compartilhamento de arquivos entre usuários. Um detalhe importante desse protocolo é que ele permite que usuários façam download de partes do arquivo de computadores que também ainda não o possuem por completo. A Figura 4 ilustra esse comportamento.

Figura 4 - Compartilhamento de arquivos no modelo P2P.