

Conexões HTTP: persistentes e não persistentes

1. Introdução

Toda comunicação HTTP depende de uma conexão de transporte, geralmente estabelecida por meio do protocolo TCP, para que requisições e respostas possam trafegar entre cliente e servidor. A forma como essa conexão é gerenciada — se ela é reaberta a cada nova requisição ou se permanece ativa para múltiplas trocas de mensagens — deu origem a dois modelos distintos: as conexões não persistentes e as conexões persistentes. Compreender a diferença entre esses dois modelos é fundamental para entender questões de desempenho, latência e eficiência no carregamento de páginas web.

2. Conexões não persistentes

Em uma conexão não persistente, cada requisição HTTP exige a abertura de uma nova conexão TCP, que é encerrada assim que a resposta correspondente é entregue. Esse era o comportamento padrão na versão HTTP/1.0. Se uma página contém múltiplos objetos — como imagens, arquivos CSS e scripts —, cada um deles exige uma conexão TCP independente, o que gera um custo adicional significativo, já que o processo de estabelecimento de conexão (conhecido como "three-way handshake") precisa ser repetido a cada novo recurso.

Esse modelo apresenta duas desvantagens principais: o aumento da latência total, pois cada conexão exige tempo adicional para ser estabelecida, e o maior consumo de recursos tanto do cliente quanto do servidor, que precisam criar e destruir conexões repetidamente.

3. Conexões persistentes

Nas conexões persistentes, introduzidas como padrão a partir do HTTP/1.1, uma única conexão TCP permanece aberta e pode ser reutilizada para o envio de múltiplas requisições e o recebimento de múltiplas respostas, sem a necessidade de reestabelecer a conexão a cada novo recurso solicitado. A conexão só é encerrada após um período de inatividade (definido por um tempo limite, ou "timeout") ou quando uma das partes decide fechá-la explicitamente.

Dentro das conexões persistentes, existem ainda duas variações quanto à forma de envio das requisições:

- **Sem pipelining** — o cliente envia uma nova requisição somente após receber a resposta da requisição anterior, mas reaproveitando a mesma conexão TCP.

- **Com pipelining** — o cliente pode enviar múltiplas requisições em sequência, sem esperar pela resposta de cada uma, reduzindo ainda mais o tempo total de carregamento.

4. Comparação entre os dois modelos

Aspecto	Não persistente	Persistente
Conexão TCP	Uma nova a cada requisição	Reutilizada para várias requisições
Desempenho	Mais lento (overhead de conexão)	Mais rápido e eficiente
Uso de recursos	Maior consumo de CPU e rede	Menor consumo de recursos
Versão padrão do HTTP	HTTP/1.0	HTTP/1.1 e superiores

Para indicar o uso de conexões persistentes, o cabeçalho HTTP "Connection: keep-alive" pode ser utilizado explicitamente, embora no HTTP/1.1 esse comportamento já seja o padrão, sendo necessário apenas o cabeçalho "Connection: close" quando se deseja encerrar a conexão após a resposta:

```
GET /index.html HTTP/1.1
Host: www.exemplo.com
Connection: keep-alive
```

5. Exemplos de aplicação

A escolha entre conexões persistentes e não persistentes impacta diretamente diversas situações práticas:

- **Carregamento de uma página com muitas imagens** — em um site de e-commerce com dezenas de fotos de produtos, uma conexão persistente permite que todas as imagens sejam baixadas por meio da mesma conexão TCP, reduzindo o tempo total de carregamento da página.
- **Aplicações de streaming de vídeo** — serviços de streaming se beneficiam de conexões persistentes para solicitar continuamente novos trechos (chunks) de vídeo sem o custo de reabrir conexões a cada segmento.
- **APIs consumidas por aplicativos móveis** — aplicativos que realizam múltiplas chamadas a uma mesma API (por exemplo, para atualizar um feed de notícias periodicamente) se beneficiam de conexões persistentes, economizando bateria e dados móveis.
- **Sistemas legados baseados em HTTP/1.0** — em sistemas antigos que ainda utilizam conexões não persistentes, cada requisição a um servidor exige uma nova conexão, o que pode ser perceptível em redes com alta latência, como conexões móveis mais lentas.

- **Monitoramento de rede em servidores de alto tráfego** — administradores de sistemas configuram o tempo limite (timeout) de conexões persistentes para equilibrar desempenho e uso de memória, evitando que conexões ociosas consumam recursos do servidor por tempo excessivo.

6. Considerações finais

A evolução das conexões não persistentes para as conexões persistentes representou um avanço importante na eficiência do protocolo HTTP, reduzindo a latência e o consumo de recursos envolvidos na comunicação entre clientes e servidores. Esse avanço, posteriormente aprimorado por técnicas como o multiplexing introduzido no HTTP/2, evidencia como pequenas mudanças na forma de gerenciar conexões podem impactar significativamente a experiência de navegação na Web.