Lista 3 - Sistemas Distribuídos

Thiago Guimarães - DRE: 118053123

2021.1

1

A arquitetura de cliente-servidor consiste em uma arquitetura onde cada componente assume um dos dois papéis: cliente ou servidor. Enquanto o servidor oferece um serviço, o cliente demanda e consome este serviço. Neste tipo de arquitetura, os papéis dos componentes são bem distintos. Com mais detalhes, o servidor aguarda a conexão de um cliente, recebe e processa um pedido e retorna o resultado para ele. Já o cliente se conecta a um servidor, envia um pedido e aguarda o retorno do servidor. Um exemplo é a Web, onde temos servidores web, por exemplo, retornando documentos HTML para clientes distintos. Em contrapartida, a arquitetura P2P consiste numa arquitetura onde os componentes realizam papéis semelhantes. A ideia central é que o cliente também é um servidor. Assim, demandas geradas por clientes passam a poder ser atendidas por outros clientes. Aqui, temos como exemplo o BitTorrent, onde múltiplos clientes compartilham arquivos entre si, todos atuando como servidor (realizando upload de arquivos) como cliente (realizando download de arquivos).

2

Na forma iterativa, o cliente realiza as requisições para cada servidor etapa a etapa, ou seja, se ele pede para um servidor que não conhece um endereço, este servidor retorna um endereço de outro servidor para que o cliente pergunte para ele. Já na forma recursiva, o próprio servidor inicial requisita para o segundo servidor o nome requisitado para o cliente. Assim, isto ocorre de forma recursiva até um servidor encontrar o nome e retornar em cascata para o cliente. Uma vantagem da forma recursiva é que o cliente não precisa realizar N requisições para encontrar o endereço IP de host. O cliente realiza apenas uma requisição e todo o processo fica abstraído.

3

O servidor de nome que precisa conhecer o endereço IP do servidor de nome autoritativo para lab.cos.ufrj.br é o cos.ufrj.br.

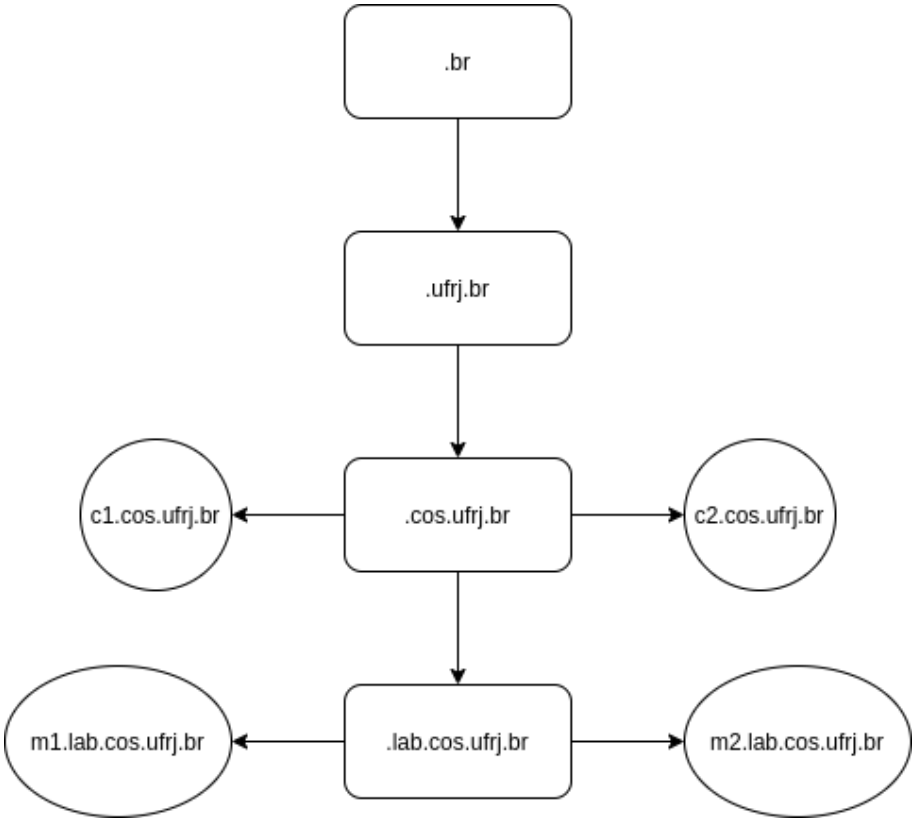

Figure 1: Desenho - DNS

4

A CDN consiste em armazenar e servir conteúdo a partir de N servidores, espalhados pelo mundo. A primeira vantagem é que, utilizando uma CDN, não permitimos a ocorrência de um ponto único de falha. Se tivéssemos um servidor único, se este está com muitas requisições ou algum problema ocorre, todo o serviço quebra. Também eliminamos o problema de distância. Tendo um servidor único, por exemplo, no Brasil, um cliente no Japão estaria prejudicado. Outro problema que é resolvido é a questão da banda. Tendo um servidor único, o envio de conteúdo fica limitado pelo tamanho do enlace conectado ao servidor. Com uma CDN, ficamos muito menos limitados nesse aspecto, dado que cada servidor da CDN está em um enlace diferente.

5

As CDNs utilizam o DNS para fornecer o nome do servidor que servirá o conteúdo para um cliente específico. Uma estratégia é ter um servidor que simplesmente redireciona o usuário para o servidor que sirva o serviço de forma mais otimizada, podendo até mesmo cachear esse par usuário - servidor.

6

Uma desvantagem é que, ao conectar inicialmente, podemos pegar peers com conexões ruins em relação a nós, seja por distância ou qualquer outro fator, tendo um desempenho ruim inicialmente.

7

Sim, é possível. Isto devido ao sistema de optimistic unchoking, onde peers aleatoriamente passam a fazer upload para peers aleatórios. Assim, este peer receberia este benefício. Contudo, este peer nunca seria um dos top 4 providers do peer e eventualmente pararia de receber uploads. Logo, seria possível, porém o sistema funcionaria de forma lenta.

8

Para que o peer entre na DHT, primeiramente, ele aplica a função hash a si. Assim, com esse valor, faz uma busca na DHT para descobrir qual o peer responsável por esse valor. Após isso, entra anterior ao peer encontrado na busca e requisita para ele o sucessor dele, de forma a armazenar seus dois sucessores, entrando assim na DHT, passando a poder responder consultas. Este é o processo de entrada de um peer qualquer na DHT. Como neste caso, temos que o peer já entrou e apenas quer tomar o lugar do seu sucessor, apenas é necessário pegar todas as chaves que passam a ser sua responsabilidade, tomando o lugar do peer mais antigo.

9

Utilizando caching, o peer poderia armazenar endereços onde ele realiza muitas buscas, o que pode melhorar muito o desempenho das buscas no DHT. O que poderia ser feito seria, por exemplo, armazenar os top 5 endereços/peers e quais valores eles são responsáveis.

10

Dois desafios que precisam ser resolvidos na implementação RPC são a garantia da execução da RPC e garantir a representação correta dos dados. O primeiro consiste no problema decorrente do fato de falhas serem muito comuns, dado que as redes e as máquinas envolvidas no processo podem falhar durante a chamada da função específica. Já o segundo advém do fato de que

sistemas diferentes armazenam e manipulam os dados de maneiras diferentes. Estamos lidando com possivelmente linguagens, endereços de memória e Sistemas Operacionais diferentes. Um exemplo disso são as representações little endian e big endian, que são utilizadas dependendo do sistema.

11

O retorno de uma chamada RPC assíncrona consiste no seguinte: Quando a função é chamada, há um retorno inicial que advém da espera do aceite do outro sistema para execução da RPC. Após isso, o programa que fez a chamada inicial continua a execução do programa. Após um delta T advindo da execução da chamada RPC, ocorre um segundo retorno via interrupção do programa - de forma similar ao funcionamento dos signal handlers - que é de fato o retorno final da chamada RPC, retomando a execução do programa a partir do ponto que esperava a interrupção.

12

A semântica "no máximo uma vez", no contexto de RPC, significa que o stub realiza a chamada repetidas vezes até receber uma resposta. O servidor armazena as chamadas e as respostas, de modo que, ao receber chamadas idênticas, não há reprocessamento, dado que estas chamadas já tiveram seus resultados armazenados, sendo retornados automaticamente. O stub implementa isto de forma que, enquanto o programa espera pela resposta, este fica realizando a chamada repetidas vezes, de forma transparente ao programador.