Introdução aos Sistemas Distribuídos

O que é um sistema distribuído?

Conjunto de processos que se coordenada para executar uma tarefa comum

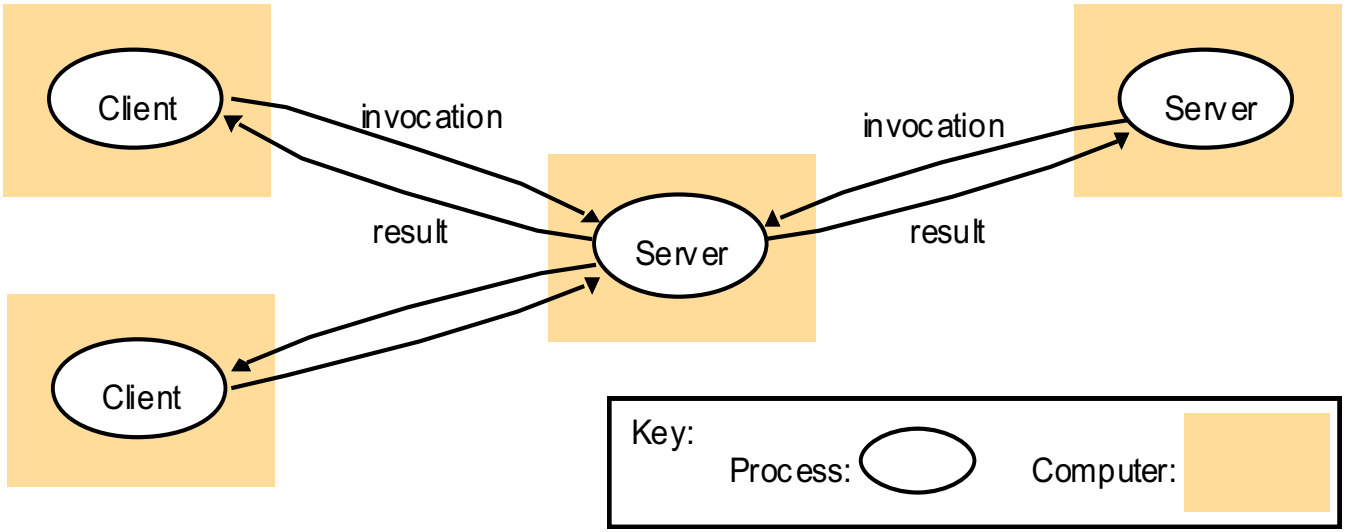
Como programar um SD?

Desafio#1: simplificar a programação de um SD

- Para criar uma aplicação distribuída é necessário definir uma **arquitetura distribuída** e um escolher um **modelo de programação**

- *Já programaram aplicações distribuídas?*

Arquitetura cliente-servidor



Projeto (hipotético) para resolvermos hoje

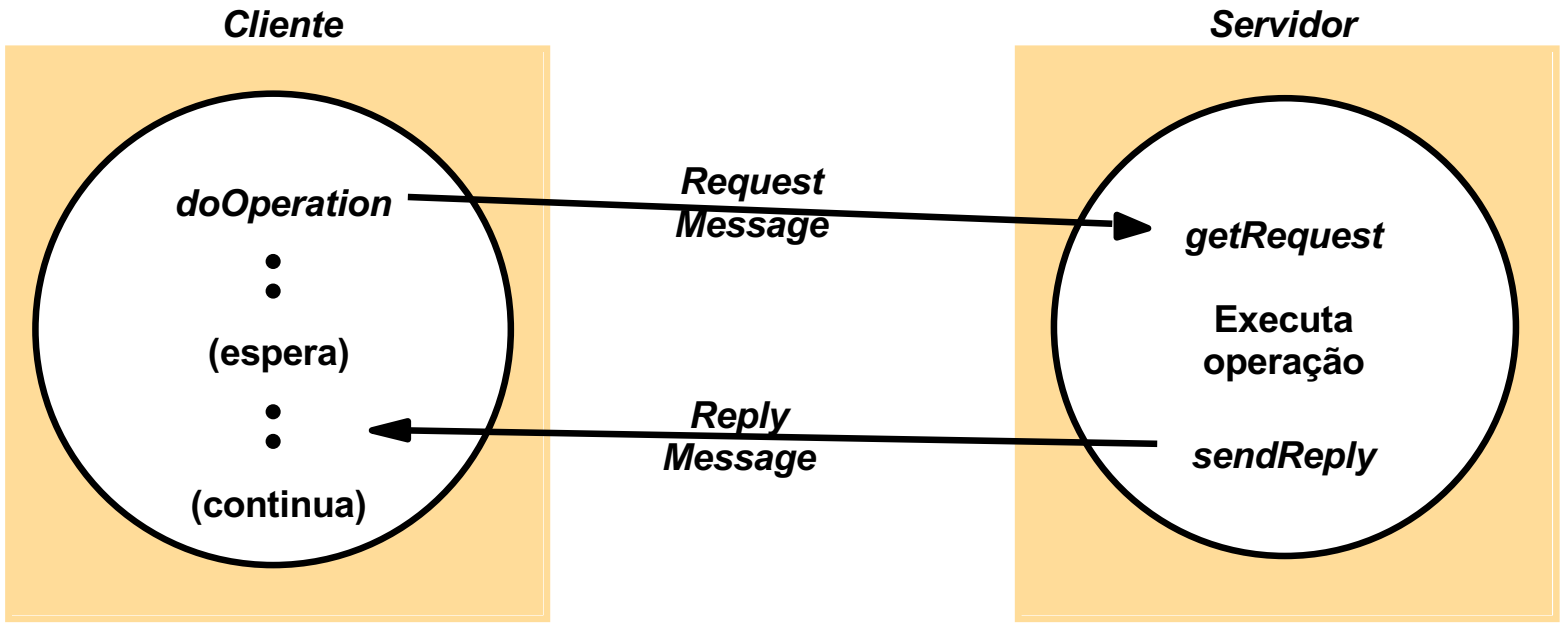
Projeto hipotético para resolvermos hoje

- Implementar um servidor de contagem que mantém um contador e oferece estas operações aos clientes:
  - ***Limpa***: coloca contador a zero
  - ***Incrementa***: incrementa o contador x unidades
  - ***Consulta***: devolve valor atual do contador

Projeto hipotético para resolvermos hoje

- Requisitos adicionais:
  - Rede não é fiável
    - Mensagens podem perder-se e chegar fora de ordem
  - Sistema heterogéneo
    - Servidor e clientes representam inteiros de forma diferente

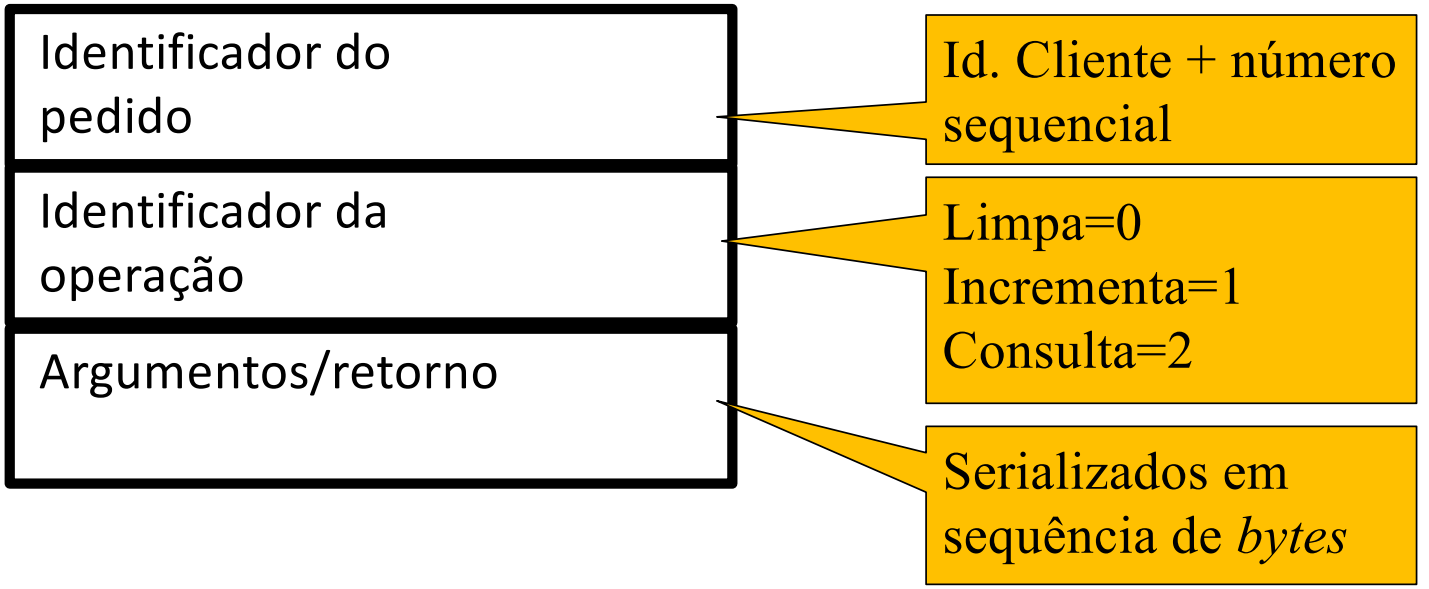
Protocolo RR (Request, Reply)



TCP ou UDP?

- Vantagens do **TCP**
  - Oferece canal fiável sobre rede não fiável
- Mas por vezes é demasiado pesado para o que precisamos
  - Para cada invocação remota passamos a precisar de mais 2 pares de mensagens
    - SYN, ACK + FIN, ACK
  - Gestão de fluxo é redundante para as invocações simples do nosso sistema
  - Confirmações (ACKs) nos pedidos são desnecessárias
    - A resposta ao pedido serve de ACK
- Vamos assumir por isso que se usa **UDP**

Conteúdo das mensagens de pedido/resposta



Modelo de faltas

- Usando UDP para enviar mensagens, estas podem:
  - Perder-se
  - Chegar repetidas
  - Chegar fora de ordem
- E os processos podem falhar silenciosamente (por *crash*)
- Como lidar com isto?

Timeout no cliente

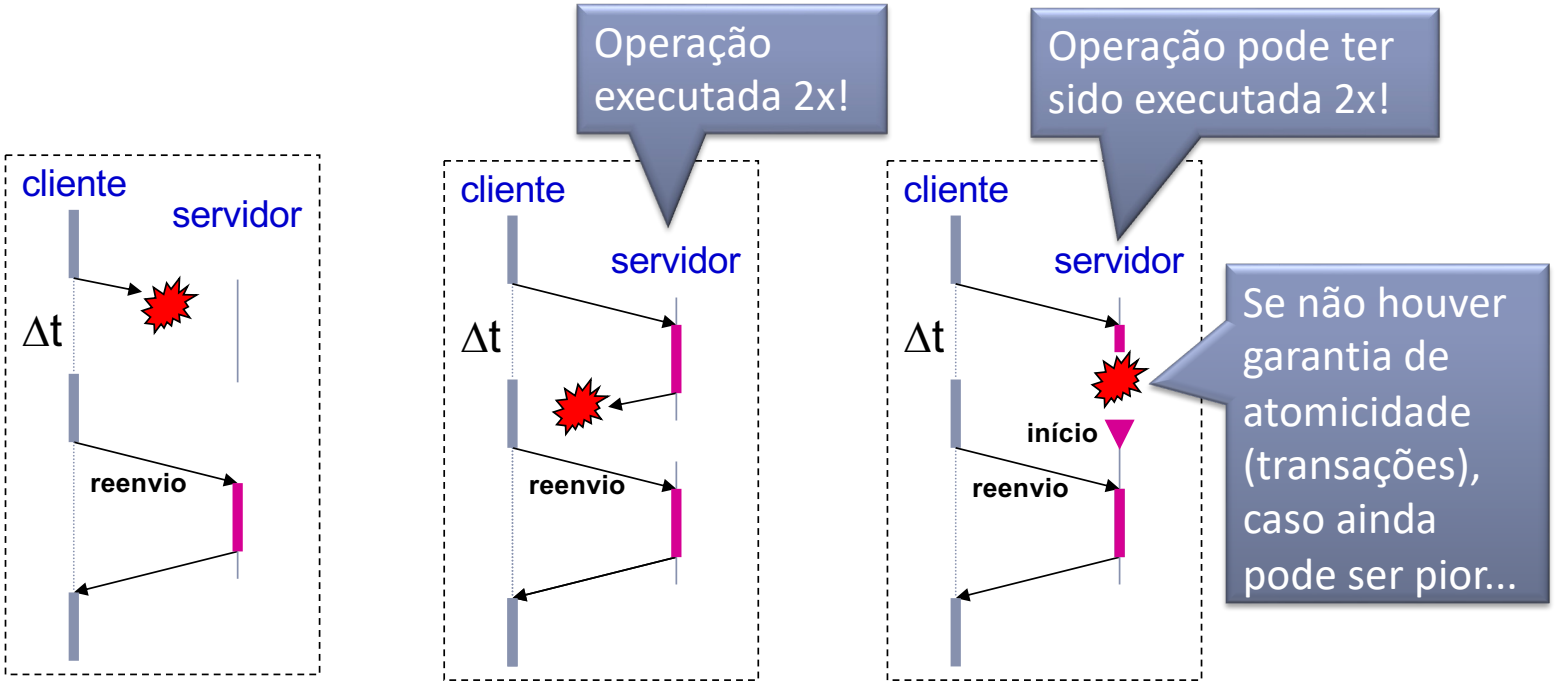
- Situação: cliente enviou pedido mas resposta não chega ao fim do *timeout*
- O que deve o cliente fazer?

- Hipótese 1: Cliente retorna erro

- Hipótese 2: Cliente reenvia pedido
  - Repete reenvio até receber resposta ou até número razoável de reenvios

Timeout no cliente com reenvio

- Quando a resposta chega após reenvio, o que pode ter acontecido?



Problema: execuções repetidas do mesmo pedido

- Perde-se tempo desnecessário
- Efeitos inesperados se operação não for **idempotente**

Função *limpa* é idempotente?
Função *incrementa* é idempotente?

Operação que, se executada repetidamente, produz o mesmo estado no servidor e resultado devolvido ao cliente do que se só executada 1 vez

Execuções repetidas do mesmo pedido: como evitar?

- Servidor deve ser capaz de verificar se *id.pedido* já foi recebido antes
- Se é a primeira vez, executa!
- Se é pedido repetido?
  - Deve guardar história de respostas de pedidos executados e retornar a resposta correspondente
  - Necessário guardar estado: e.g., tabela com (*id.pedido*, resposta)

Quantos pedidos manter por cliente?

Como escalar para grande número de clientes?

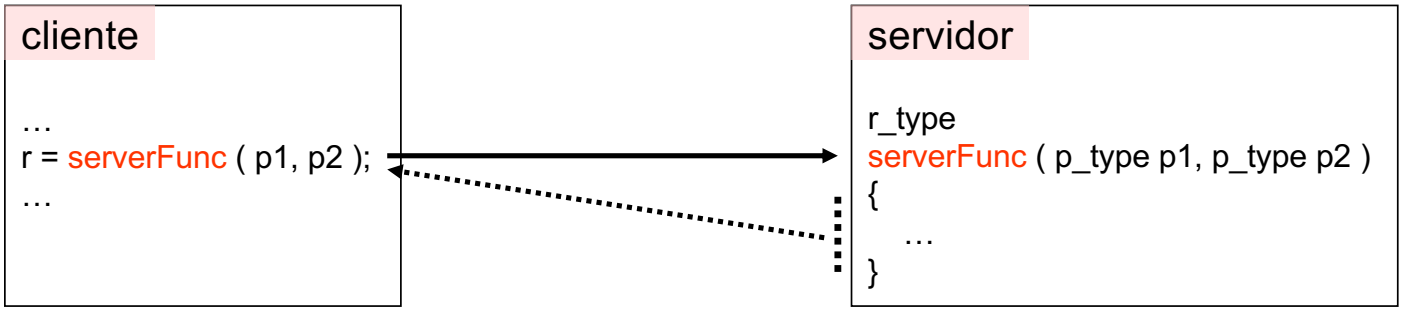
Programar sistemas distribuídos usando sockets é um processo complexo, difícil e muito propenso a erros.

Vamos aumentar um pouco mais o nível de abstração

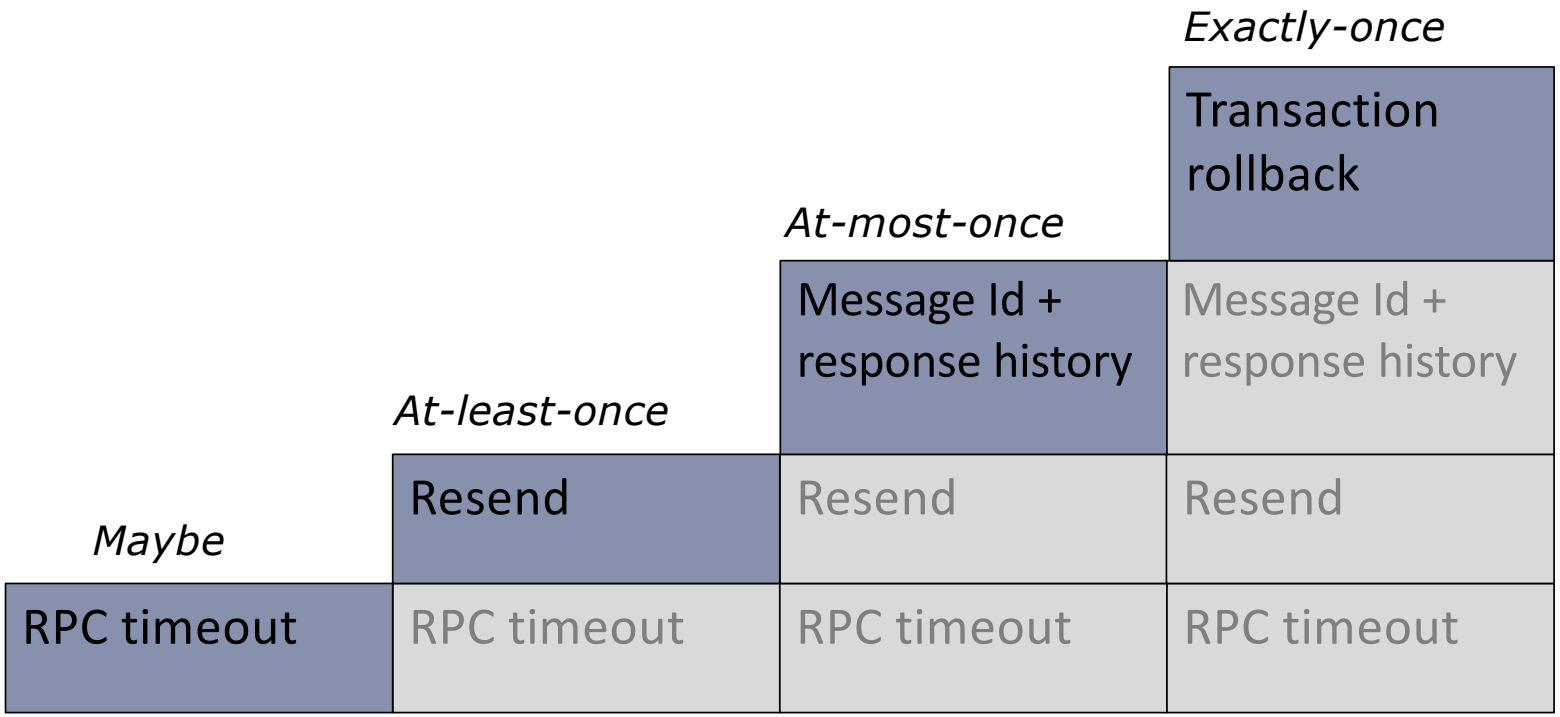
Chamadas de Procedimentos Remotos (RPC)

RPC: visão do programador

- O programador chama uma função (procedimento) aparentemente local
- A função é executada remotamente no servidor
  - Acedendo a dados mantidos no servidor



RPC: semânticas de invocação e mecanismos necessários



Deixem-me mostrar como isto foi resolvido em 1985 no SUN RPC

`[exemplo-sun-rpc-contador.zip]`

GRPC

- Uma tecnologia RPC atual

- Para aprenderem nos laboratórios
- Será o RPC usado no projeto

Mas SD não é apenas troca de mensagens por RPC…

Como implementar um SD?

- Troca de mensagens
- Partilha de memória

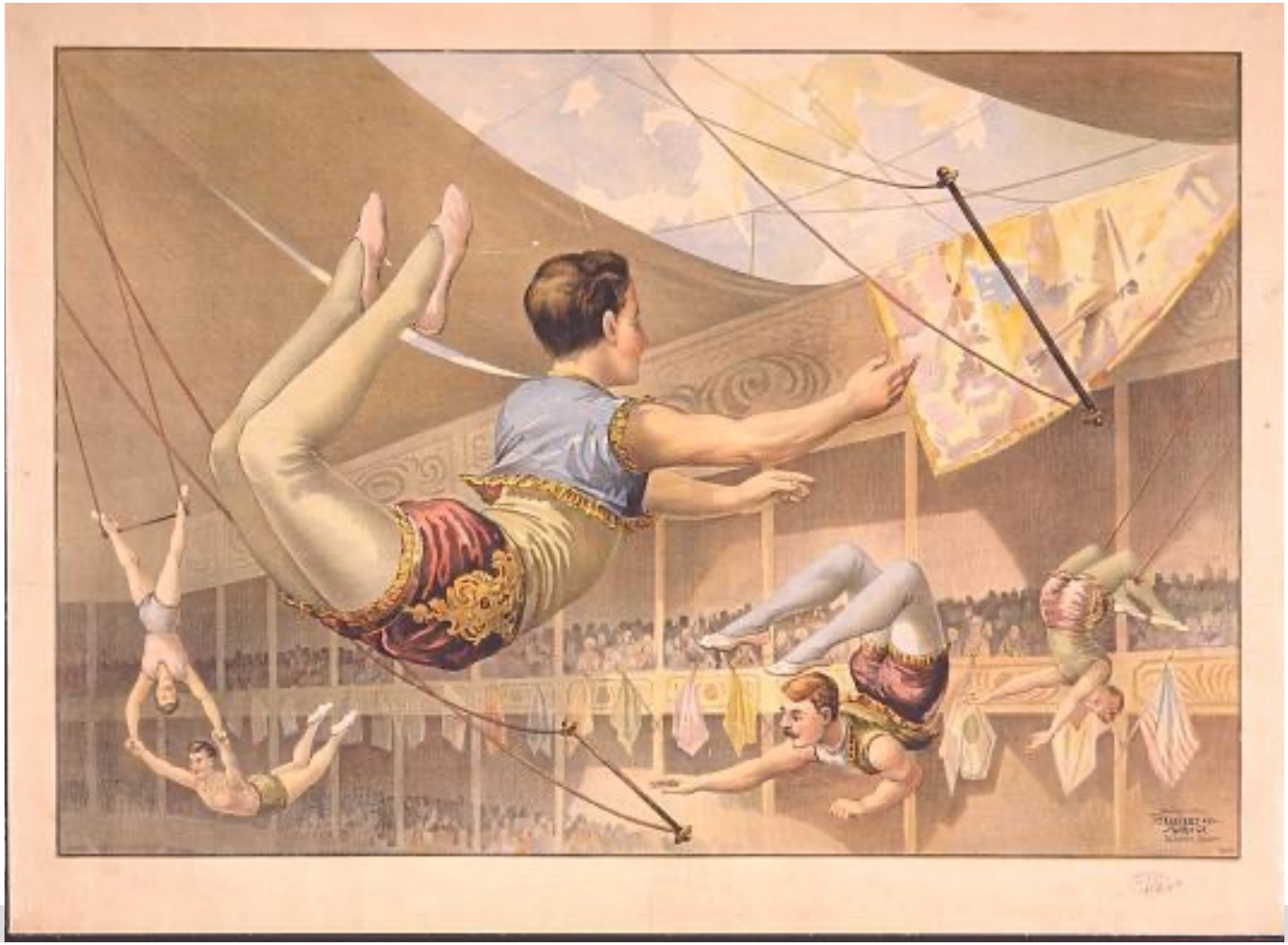
Troca de mensagens

- Chamada a procedimentos remotos
- Difusão em grupo
- Edição-subscrição de eventos

Partilha de memória

- Memória partilhada distribuída
- Espaços de tuplos distribuídos

Os desafios de SD



Quais os desafios?

- Concorrência
- Latência
- Coerência
- Tolerância a faltas
- Capacidade de escala
- Segurança

Concorrência

- Más notícias:
  - Os sistemas distribuídos são concorrentes

Concorrência

- Más notícias:
  - Os sistemas distribuídos são concorrentes
- Boas notícias
  - Os sistemas distribuídos são concorrentes

Concorrência

- Más notícias:
  - Os sistemas distribuídos são concorrentes
- Boas notícias
  - Os sistemas distribuídos são concorrentes
  - **São precisos bons profissionais para fazer bons sistemas distribuídos!**

Latência

- Consta que existe uma constante chamada velocidade da luz
- Isto significa que:
  - Por muito potente que seja a máquina
  - Por muita largura de banda que tenha a rede
  - Um processo em Portugal a coordenar-se com um processo no Japão será sempre uma tarefa lenta.

Coerência

- Quando um processo faz uma alteração ao estado do sistema…
- …esta alteração não fica instaneamente visível nos restantes processos
- Isto pode ser um problema:
  - Que comportamentos podemos considerar correctos?
  - Podemos ter diversas definições de coerência

Tolerância a faltas

- Faltas nos processos
  - Paragem
  - Bizantinas
- Faltas na rede
  - Perda de mensagems
  - Reordenação de mensagens
  - Partições na rede

Detecção de falhas

- Pode ser impossível ter um detector de falhas perfeito!

- Um processo pode ter falhado, pode estar simplemente lento, ou com problemas na rede.

Capacidade de escala

- Conceber sistemas que continuam a funcionar à medida que acrescentamos mais processos ao sistema

Segurança

- Não é um problema específico dos sistemas distribuídos
- Mas é amplificado pelos sistemas distribuídos
  - Mais processos = maior exposição

No resto desta aula

- Como simplificar a programação de um SD
- Exemplo de um SD de larga escala (DNS)

DNS

Exemplo de um SD de larga escala

DNS

- Serviços de directório
  - Procurar impressoras a cores que façam frente e verso e que tirem cafés
- Serviços de nomes
  - Dado um nome saber um endereço

Exemplo

DNS: exemplo com nslookup

```
$nslookup www.gsd.inesc-id.pt
Server:     192.168.1.1
Address:    192.168.1.1#53

Non-authoritative answer:
Name: www.gsd.inesc-id.pt
Address: 146.193.41.139
```

DNS: exemplo com nslookup

```
$nslookup -type=ns gsd.inesc-id.pt
Server:     212.113.177.241
Address:    212.113.177.241#53

Server:     192.168.1.1
Address:    192.168.1.1#53

Non-authoritative answer:
gsd.inesc-id.pt   nameserver = ns1.gsd.inesc-id.pt.
gsd.inesc-id.pt   nameserver = ns2.gsd.inesc-id.pt.
gsd.inesc-id.pt   nameserver = inesc-id.inesc-id.pt.
```

DNS: exemplo com nslookup

```
$nslookup www.gsd.inesc-id.pt ns1.gsd.inesc-id.pt
Server:     ns1.gsd.inesc-id.pt
Address:    146.193.41.2#53

Name: www.gsd.inesc-id.pt
Address: 146.193.41.139
```

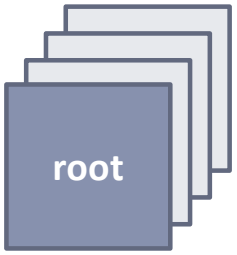
De que maneira o DNS lida com os desafios de SD?

DNS

- Um servidor central

DNS

- Um servidor central replicado

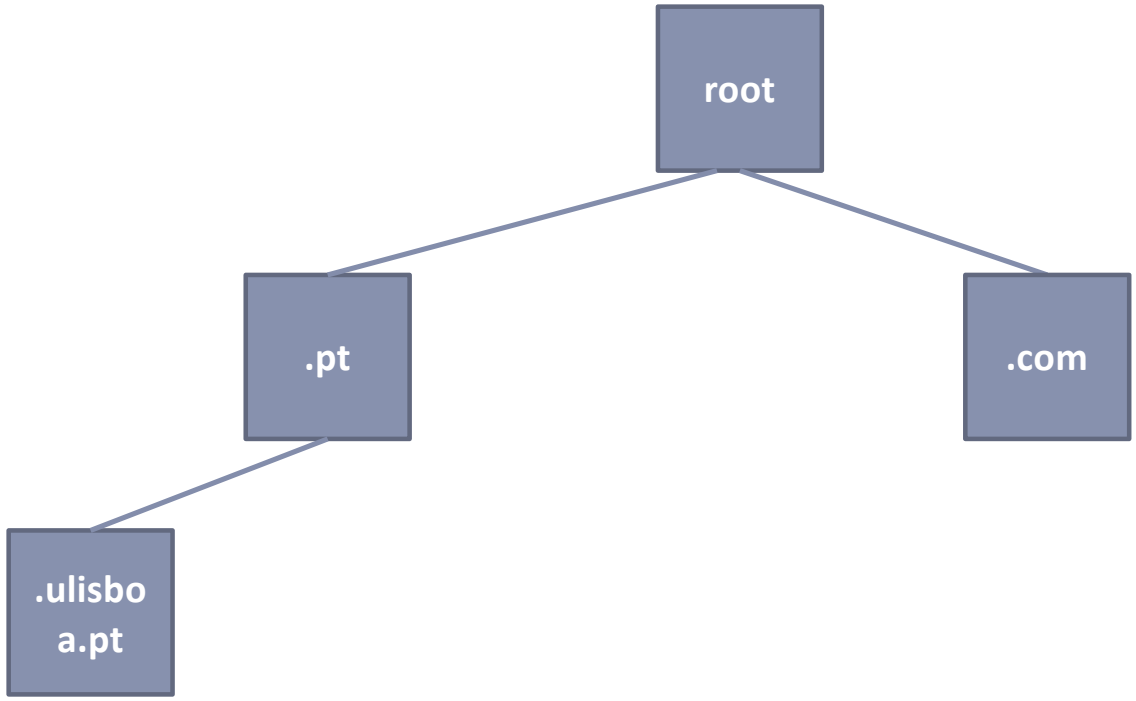


DNS

- Um servidor central replicado e geo-replicado

DNS

- Uma hierarquia de servidores cooperativos



DNS

- Resolução de um nome
  - Através de um componente desigado por “resolver”
- Duas estratégias possíveis
  - Resolução iterativa
  - Resolução recursiva

DNS

- Cache
  - Em cada organização existe um servidor que faz a resolução e guarda os resultados
  - Cache pode gerar problema de coerência!

Algumas notas

- DNS
  - Os requisitos de coerência do DNS são relativamente fracos

- O DSN usa nomes hierárquicos

Nomes

- Como dar nomes a entidades é um problema interessante por si só
  - Que não abordaremos em pormenor nas aulas teoricas

- `/Users/ler/…`
- `/Volumes/Google Drive/…`
- `C:\DOSTEMP`
- `FA19 B295 C86D 993C 8432 DEFD 11AA 88E6`

Nomes puros

- Como encontrar o endereço de uma entidade a partir de um nome puro?

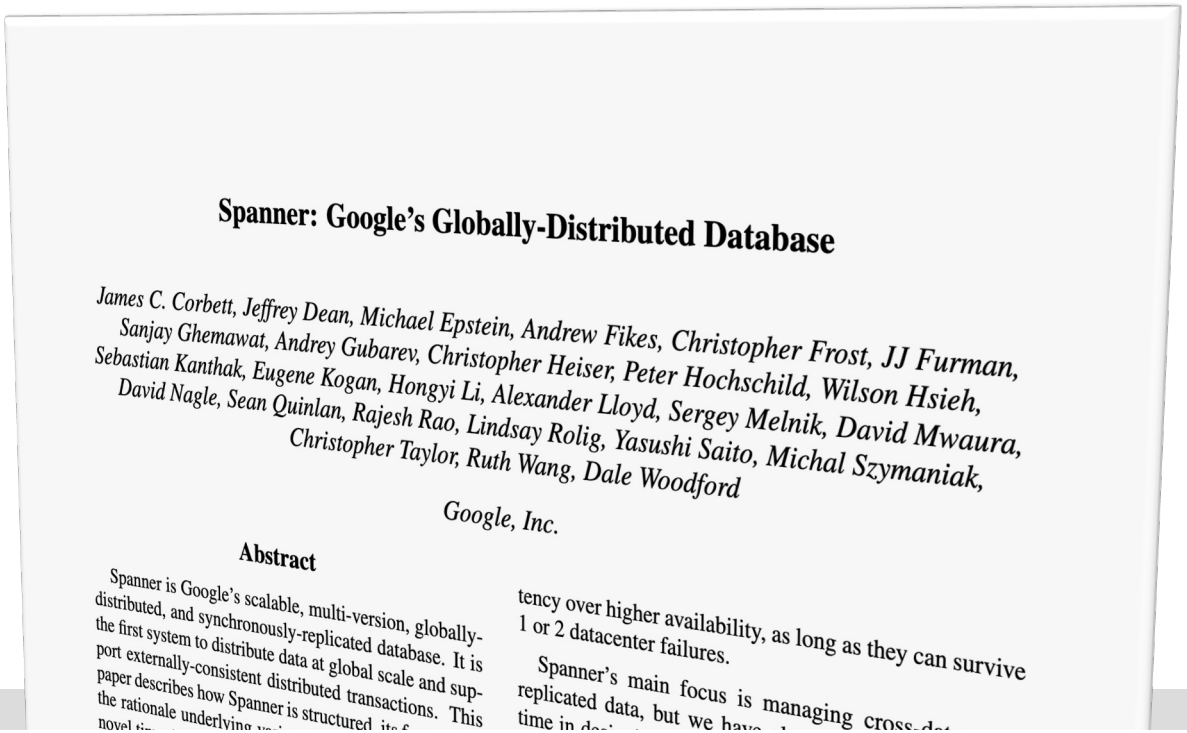
```
FA19 B295 C86D 993C 8432 DEFD 11AA 88E6
```

Outro exemplo de um SD de larga escala

Para verem for a da aula

Spanner: Google's Globally Distributed Database (2012)

- Ler as primeiras páginas do artigo



Spanner: Google's Globally-Distributed Database

James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, Dale Woodford

Google, Inc.

Abstract

Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This paper describes how Spanner is structured, its

tency over higher availability, as long as they can survive 1 or 2 datacenter failures.

Spanner's main focus is managing cross-

replicated data, but we have

time in

Spanner: Google's Globally Distributed Database (2012)

- Ler as primeiras páginas do artigo
- Que técnicas o Spanner usa para lidar com estes desafios?
  - Latência
  - Coerência
  - Tolerância a faltas
  - Capacidade de escala
  - Segurança
- Voltaremos a isto no início da próxima aula

Bibliografia recomendada

- **Secções 1.1-1.5 4.2, 5.1-5.3, 13.1-13.2**