

Sistemas Distribuídos

Sistemas de Informação e Ciência da Computação

Lásaro Camargos

Faculdade de Computação
Universidade Federal de Uberlândia

3 de Abril de 2019

Introdução
Projeto
Objetivos e
Desafios
Da pedra ao
smartphone
Tipos e
Arquiteturas
Projeto

Parte I

Introdução

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

1 Introdução

Motivação

Motivação 1: Diversão

"I think that it's extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun. Above all, I hope we don't become missionaries. Don't feel as if you're Bible salesmen. The world has too many of those already. What you know about computing other people will learn. Don't feel as if the key to successful computing is only in your hands. What's in your hands, I think and hope, is intelligence: the ability to see the machine as more than when you were first led up to it, that you can make it more."

Alan J. Perlis.

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

O que **você** gosta na computação?

Motivação 2: Dinheiro

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Proficiência em sistemas distribuídos é forma de destaque.
- Quora
- Glassdoor
- Data science/engineer
- Facebook
-

Motivação 3: Prática

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Chance de fazer um projeto com os pés na realidade.
O que te faz um bom “computeiro” é uma visão geral, uma capacidade de usar múltiplas ferramentas, não ser um especialista em um paradigma e o detentor de segredos (opinião minha).
Neste sentido, este curso é generalista, apresentando diversos problemas, abordagens, um pouco de teoria, um pouco de prática.

Motivação 4: Obrigação

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- É uma disciplina obrigatória

Objetivo I

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- programar processos que se comuniquem via redes de computadores;
- conhecer arquiteturas clássicas de sistemas distribuídos (e.g., cliente/servidor, p2p e híbrida), seus usos e limitações;
- escrever programas *multithreaded* simples e a entender como o uso de *multithreading* afeta os componentes de um sistema distribuído;
- entender a problemática da coordenação e do controle de concorrência em sistemas distribuídos;
- entender o uso de sistemas de nomeação em sistemas distribuídos bem como diversas formas de se implementar tais sistemas de nomeação;

Objetivo II

- entender os conceitos básicos de replicação e tolerância a falhas;
- entender as implicações da dessincronização de relógios na coordenação, replicação e tolerância a falhas;
- projetar sistemas com componentes geograficamente distantes, fracamente acoplados;
- entender onde os diversos *middleware* podem ser usados para acoplar tais componentes;
- conhecer várias técnicas que controle de concorrência controlar o acesso a um recurso compartilhado;

Introdução
Motivação
Logística
Computação
Distribuída

Projeto
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

1 Introdução

■ Logística

Logística

Aulas

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Expositivas, mas interativas
- Laboratórios **tão frequentemente quanto possível**
- Frequência controlada todos os dias

Logística

Avaliação

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Laboratórios (0%)
- 3 trabalhos **incrementais** (75%)
- 1 seminário (25%)
- Código entregue via repositório **git** (github) na data especificada
- Cada dia de atraso implica 3% de desconto no valor da entrega; atraso máximo de 1 semana.
- Código deve compilar usando (**maven**)
- Testes devem demonstrar atendimento dos requisitos, usando maven (`mvn test`)

Logística

Atendimento

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Piazza: piazza.com/ufu.br/semester12019/gbc074gsi028
- TODA COMUNICAÇÃO deve ser feita via Piazza.
- Atendimento presencial
 - Agendado
 - Sala 1b149
 - Seg. 10:30 e 19:00

Introdução
Motivação
Logística
Computação
Distribuída

Projeto
Objetivos e
Desafios
Da pedra ao
smartphone
Tipos e
Arquiteturas
Projeto

Dúvidas?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

1 Introdução

■ Computação Distribuída

Programar

Na faculdade

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto



Programar

Indústria

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto



Programar

Distribuído

Introdução

Motivação

Logística

Computação

Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto



Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Se computação distribuída é tão complicado, por quê fazê-lo?

Por quê fazê-lo?

- Computadores individuais tem capacidade limitada.
- Você precisa agregar recursos (processamento e armazenamento)
- e Tolerância a falhas.

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

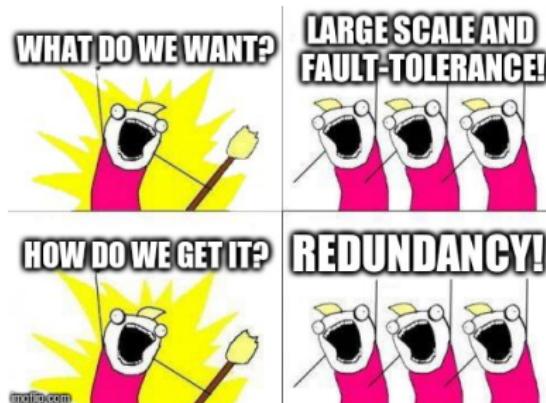
Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Por quê fazê-lo?

- Computadores individuais tem capacidade limitada.
- Você precisa agregar recursos (processamento e armazenamento)
- e Tolerância a falhas.



Por quê fazê-lo?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A máquina em que a computação roda

- está longe do cliente.

Por quê fazê-lo?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A máquina em que a computação roda

- está longe do cliente.
- não pode ficar ao alcance do cliente.

Por quê fazê-lo?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A máquina em que a computação roda

- está longe do cliente.
- não pode ficar ao alcance do cliente.
- precisa atender a diversos clientes, simultaneamente.

Por quê fazê-lo?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A máquina em que a computação roda

- está longe do cliente.
- não pode ficar ao alcance do cliente.
- precisa atender a diversos clientes, simultaneamente.
- tem poder computacional insuficiente.

Por quê fazê-lo?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A máquina em que a computação roda

- está longe do cliente.
- não pode ficar ao alcance do cliente.
- precisa atender a diversos clientes, simultaneamente.
- tem poder computacional insuficiente.
- não pode ser um ponto único de falha, isto é, não pode levar o serviço a ficar indisponível em caso de falhas.

O que é distribuir?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas

O que é distribuir?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas
- e coordenar suas ações para que resolvam a tarefa em questão de forma eficiente

O que é distribuir?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas
- e coordenar suas ações para que resolvam a tarefa em questão de forma eficiente
- e usando de redundância para garantir que o serviço continua sendo provido a despeito de falhas.

O que é distribuir?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Isto é, distribuir é fazer com que diversas máquinas colaborem na execução de um tarefa. Alguns exemplos são:

Colaboração

- Entregue este email para fulano@knowhere.uni.
- Envie o item X para este endereço, após cobrança de Y dinheiros da conta Z.
- Em um ambiente de simulação de batalhas em 3D, simule o disparo de um projétil nesta direção e sentido, com velocidade v, enquanto movimenta o avatar A para a esquerda.
- Autorize a transferência de X dinheiros da conta C para a conta C'
- Movimente o braço mecânico que está segurando um bisturi, 3cm à direita, então abaixe-o 3mm, e movimente-o 4cm para a esquerda
- Inclua o comentário “LOL!!!” na lista de comentários do item XYZ, com marca de tempo T
- Leia o valor do sensor de temperatura S e, caso seu valor supere V, emita alarme luminoso vermelho intermitente e alarme sonoro

Colaborar por quê?

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Para ganhar acesso a “recursos” como

- memória
- CPU
- impressora
- canal de comunicação
- localização

de outros dispositivos.

Definição I

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Sistema Distribuído

Coleção de sistemas computacionais (software ou hardware) independentes que colaboram na execução de alguma tarefa.

Definição II

Introdução
Motivação
Logística
Computação
Distribuída

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable

Leslie Lamport

O projeto

Visão Geral

Introdução

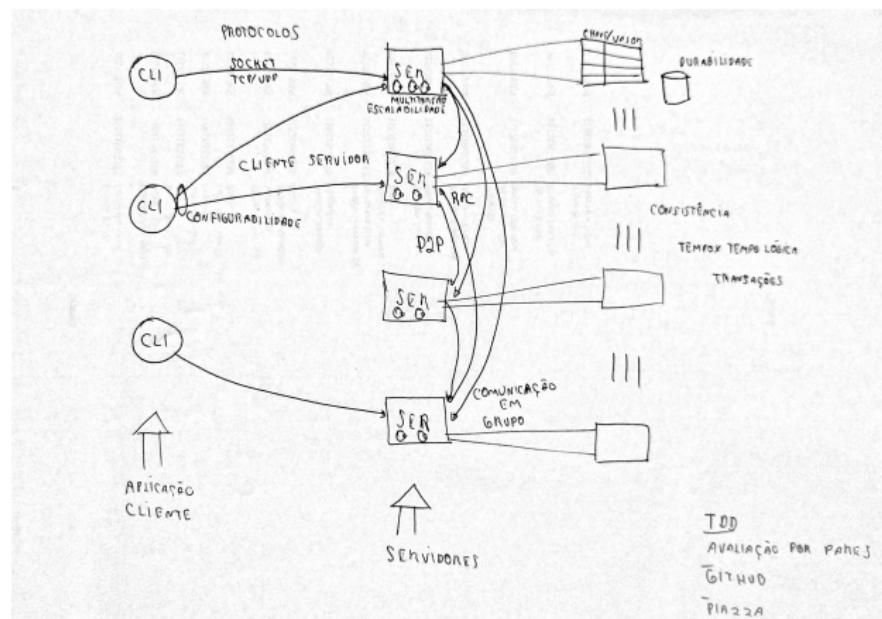
Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto



Banco de dados chave/valor distribuído, particionado e replicado. Comunicação por meio de sockets e RPC (gRPC). Servidores e clientes multithread. Java.

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Fim da aula 1

O quê é distribuir?

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas

O quê é distribuir?

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas
- e coordenar suas ações para que resolvam a tarefa em questão de forma eficiente

O quê é distribuir?

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Dividir a computação/armazenamento em diversas máquinas
- e coordenar suas ações para que resolvam a tarefa em questão de forma eficiente
- e usando de redundância para garantir que o serviço continua sendo provido a despeito de falhas.

Comunicação

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

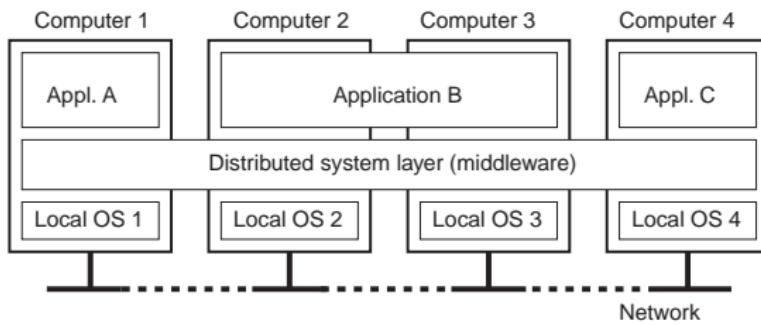
Tipos e
Arquiteturas

Projeto

- Mensagens
- Streams
- Invocação remota de procedimentos
- Mais e mais abstrações

Visão Geral

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto



- Sistemas independentes
- Middleware

Sistemas Independentes: dificuldades

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Falhas independentes
- Relógios dessincronizados
- Linguagens de desenvolvimento diferentes
- Sistemas operacionais diferentes
- Arquiteturas diferentes
- Desenvolvedores diferentes

Middleware

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

The software layer that lies between the operating system and applications on each side of a distributed computing system in a network.

Sacha Krakowiak

Middleware

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Os middleware facilitam a conexão entre nós e permitem o uso de protocolos mais abstratos que “mandar um monte de bytes” para lá e para cá, escondendo a complexidade da coordenação de sistemas independentes.

Middleware

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Desenvolver sistemas distribuídos sem usar um middleware é como desenvolver um aplicativo qualquer, sem usar bibliotecas: possível, mas complicado, e estará certamente reinventando a roda.

Transparência

Coleção de sistemas computacionais (software ou hardware) independentes que se apresentam para o usuário *como um sistema único*. Pense no browser e na WWW.

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Transparência

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

Coleção de sistemas computacionais (software ou hardware) independentes que se apresentam para o usuário *como um sistema único*. Pense no browser e na WWW.

- Acesso: Representação de dados e mecanismos de invocação (arquitetura, formatos, linguagens...)
- Localização: Onde está o objeto?
- Relocação: Cliente não percebe se objeto for movimentado.
- Migração: Objeto não percebe se for movimentado.
- Replicação: Objeto é replicado (tem várias cópias).
- Concorrência: Objeto é acessado por múltiplos clientes (sem interferência).
- Falha: Falha e recuperação não são percebidas (usando-se cópias)

A realidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Do ponto de vista do usuário

- Usuários podem estar espalhados pelo mundo e perceberão latências diferentes.

Do ponto de vista do desenvolvedor

- Impossível distinguir um computador lento de um falho.

De forma geral

- Aumentar transparência custa desempenho.

Transparência de acesso

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Usar padrões abertos e interfaces bem definidas.

- Sistemas bem comportados e previsíveis (RPC/ASN.1)
- Que interajam bem com outros via interfaces bem definidas (REST)
- Suportem aplicações diferentes do mesmo jeito (API)

Transparência Localização e Relocação

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- rede de baixa latência
- Serviços de nome
- Múltiplas cópias
- Cópias temporárias

Transparência Migração

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

■ Virtualização

Transparência de Replicação

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

- Middleware de comunicação em grupo.

Transparência Concorrência

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- controle de concorrência adequado.
- mecanismos para alcançar
escalabilidade(particionamento/sharding)

Transparência a Falhas

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- mecanismos de tolerância a falhas (replicação)

Dificuldade? A Rede

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

Diversos dos problemas que enfrentamos no desenvolvimento de SD estão ligados às redes de computadores.

Armadilhas

A rede

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- A latência é zero.
- A largura de banda é infinita.
- A rede é confiável.
- A rede é segura.
- A rede é homogênea.
- A rede é estática.
- A rede é de graça.
- A rede é administrada por você ou alguém acessível.

Dificuldade? Escalabilidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

O quê quer dizer um sistema ser escalável?

Tipos

Escalabilidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Tamanho: Número de usuários que suporta.
- Geográfica: Região que cobre.
- Administrativa: Número de domínios administrativos.

Há várias possibilidades: seja específico e exija especificidade.

Como? I

Escalabilidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Esconda latência.

- Comunicação assíncrona: tarefas paralelas.
E.g., em vez de validar formulário após preenchimento de cada campo, valide em paralelo enquanto usuário preenche campo seguinte.
- Use callbacks.
E.g., gatilhe tratamento de erro no formulário em função separada.

Como? II

Escalabilidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Distribuição de tarefas.

- Delege computação aos clientes
E.g., JavaScript e Applets Java.
- Particione dados entre servidores
E.g., Domain Name Service e World Wide Web.

Como? III

Escalabilidade

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Aproxime dados dos clientes

- Mantenha cópias de dados em múltiplos lugares.
- Atualize dados de acordo com necessidade.
E.g., cache do navegador.

Dificuldade? Tolerar falhas

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Replicação e cache.

- Mantenha cópias de dados em múltiplos lugares.
- Atualize dados de acordo com necessidade.

Replicação x Inconsistências

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Múltiplas cópias → em sincronização → custos

- Dado precisa ser consistente entre réplicas (mesmo valor em todo lugar)
- Protocolos de invalidação de cache.
- Muita largura de banda.
- Baixa latência.

Replicação x Inconsistências

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Múltiplas cópias → em sincronização → custos

- Dado precisa ser consistente entre réplicas (mesmo valor em todo lugar)
- Protocolos de invalidação de cache.
- Muita largura de banda.
- Baixa latência.

Algumas aplicações toleram inconsistências, como carrinhos de compra.

Replicação x Inconsistências

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Múltiplas cópias → em sincronização → custos

- Dado precisa ser consistente entre réplicas (mesmo valor em todo lugar)
- Protocolos de invalidação de cache.
- Muita largura de banda.
- Baixa latência.

Algumas aplicações toleram inconsistências, como carrinhos de compra.

Mas não todas, como sistemas de bancos.

Um pouco de história I

Introdução

Projeto

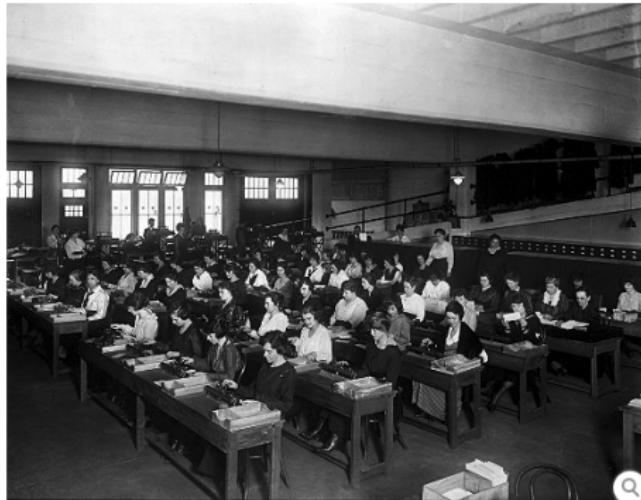
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

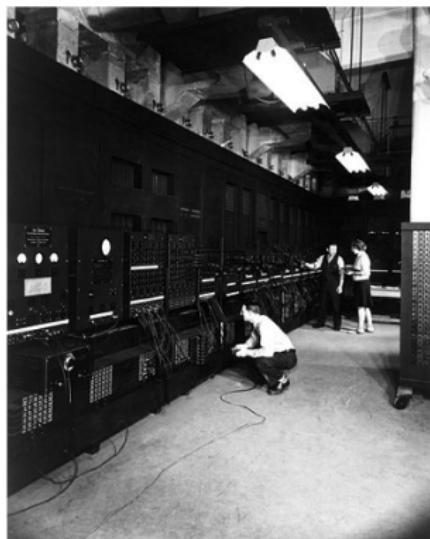
- 1930 – As pessoas eram os computadores.



Um pouco de história II

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

■ 1942 – ENIAC



Um pouco de história III

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- 1948 - A invenção do software – Mark I
- Os primeiros sistemas distribuídos? Reservas de passagens...

Um pouco de história IV

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

- 1964 - IBM 360



- Scale up!

Um pouco de história V

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Minicomputadores
- Morte do mainframe

Um pouco de história VI

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Supercomputadores
- 1994 - Beowulf



- Scale out!

Um pouco de história VII

Introdução

Projeto

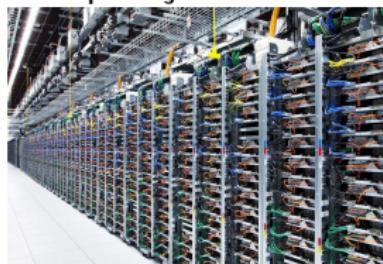
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Computação utilitária.



- Datacenter em 360

Um pouco de história VIII

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

■ Computadores portáteis



Um pouco de história IX

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

Visitem o sítio do **Computer History Museum**

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

A expansão do uso e a necessidade de crescimento dos computadores levou ao surgimento da computação distribuída...

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto

A expansão do uso e a necessidade de crescimento dos computadores levou ao surgimento da computação distribuída... e com ela a necessidade de comunicação entre os componentes...

A expansão do uso e a necessidade de crescimento dos computadores levou ao surgimento da computação distribuída... e com ela a necessidade de comunicação entre os componentes... mas a comunicação é uma velha conhecida nossa.

Comunicação I

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Mensagens – comunicação ponto a ponto.
- Diferentes larguras de banda – correndo, cavalgando, de trem, de avião.

Comunicação II

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

■ Perda de mensagens

2350BC: Rei Sargon de Akkad (Mesopotamia)

- Mensageiros carregavam pombos correio.
- Ao serem atacados, liberavam o pombo para avisar do ataque:
NACK!
- A falta de uma mensagem é boa notícia.

Comunicação III

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

■ Repetidores



Fonte: O Senhor dos Anéis: o Retorno do Rei

Comunicação IV

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- **Roteamento
Estações de trem.**

Comunicação V

Introdução

Projeto

Objetivos e
Desafios

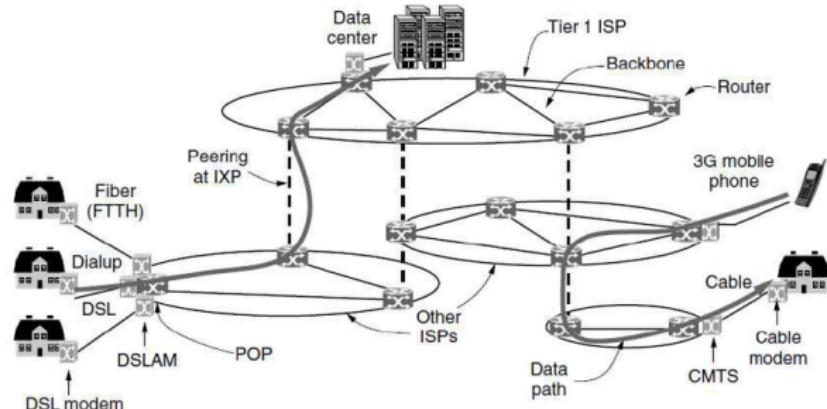
Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Broadcasts – sinais de fumaça
- Relógio d'água
- Telescópio
- Controle de erro

A Internet



Fonte

A Internet

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- Provedores e Autonomous Systems
- Redundância de caminhos
- Roteamento: Packet Switching x Circuit Switching
- Uso eficiente dos canais
- Store and forward overhead

Por quê é importante?

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

- A rede é o substrato do sistema distribuído
- Não é simples
- Diversas nuances
- Muito o que aprender

Introdução

Projeto

Objetivos e
Desafios

**Da pedra ao
smartphone**

Tipos e
Arquiteturas

Projeto

Fim da aula 2

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos
Arquiteturas
Projeto

5 Tipos e Arquiteturas

■ Tipos

High Performance Computing I

Sistemas de Computação

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



High Performance Computing II

Sistemas de Computação

Introdução

Projeto

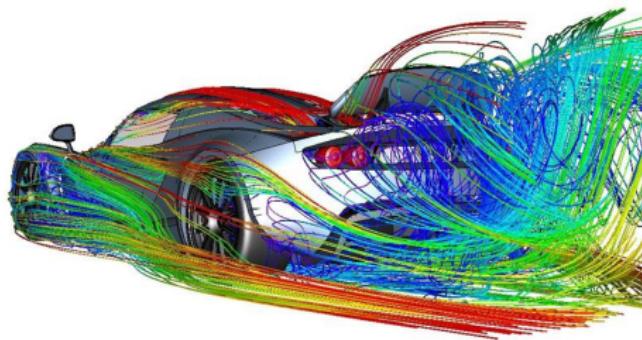
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



High Performance Computing III

Sistemas de Computação

Introdução

Projeto

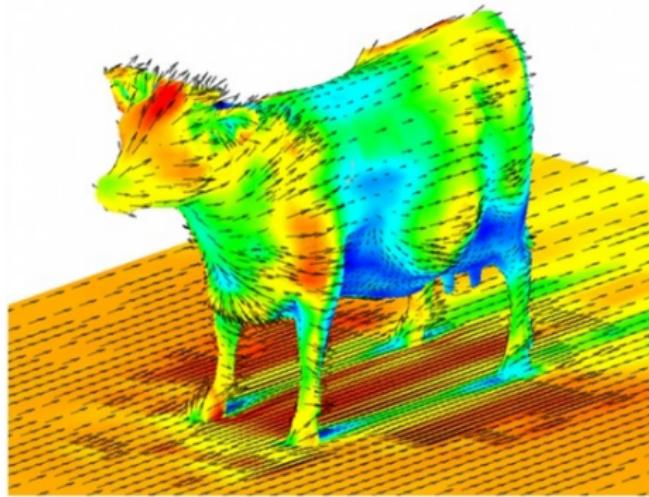
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Bancos de dados I

Sistemas de Informação Distribuído

Introdução

Projeto

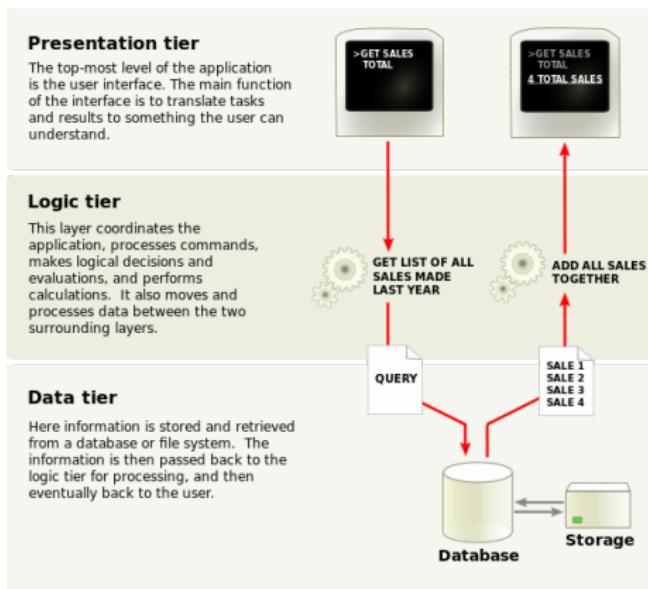
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Fonte

Bancos de dados II

Sistemas de Informação Distribuído

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos

Arquiteturas

Projeto

ACID

- Atomicidade: transações são tratadas de forma indivisível, isto é, ou tudo ou nada.
- Consistência: transações levam banco de um estado consistente a outro
E.g., $x == 2^y$
- Isolamento: transações não vêem dados não comitados umas das outras.
- Os efeitos de uma transação comitada devem persistir no sistema a despeito de falhas.

Bancos de dados III

Sistemas de Informação Distribuído

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos

Arquiteturas

Projeto

Transação

```
1: a = SELECT X
2: c = a * 2
3: b = c + 10
4: SET X=c
5: SET Y=b
```

Transações T_1 e T_2 , intercaladas.

$T_{11}, T_{12}, T_{13}, T_{14}, T_{21}, T_{22}, T_{23}, T_{24}, T_{25}, T_{15}$

Bancos de dados IV

Sistemas de Informação Distribuído

Introdução

Projeto

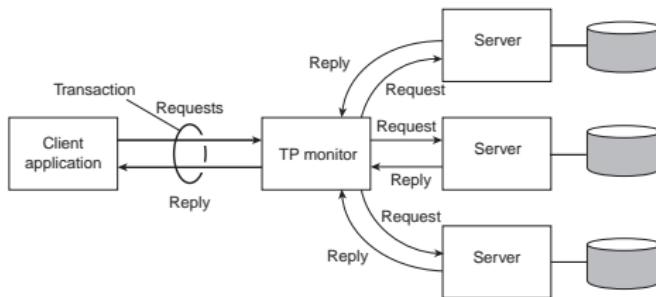
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Bancos de dados V

Sistemas de Informação Distribuído

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

Type	Example
Key-Value Store	 redis
Wide Column Store	 HBASE
Document Store	 mongoDB
Graph Store	 Neo4j  InfiniteGraph The Distributed Graph Database

Fonte

Integração de aplicações I

Sistemas de Informação Distribuído

Introdução

Projeto

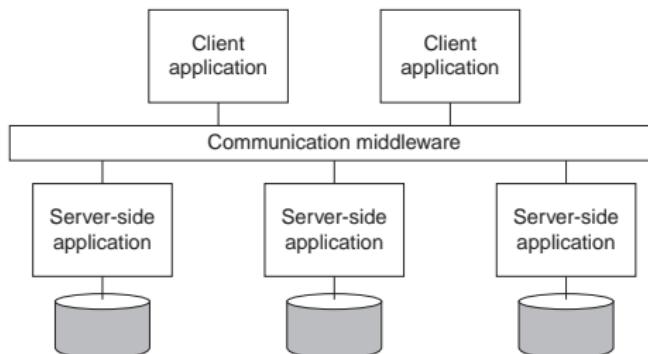
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Integração de aplicações II

Sistemas de Informação Distribuído

Introdução

Projeto

Objetivos e

Desafios

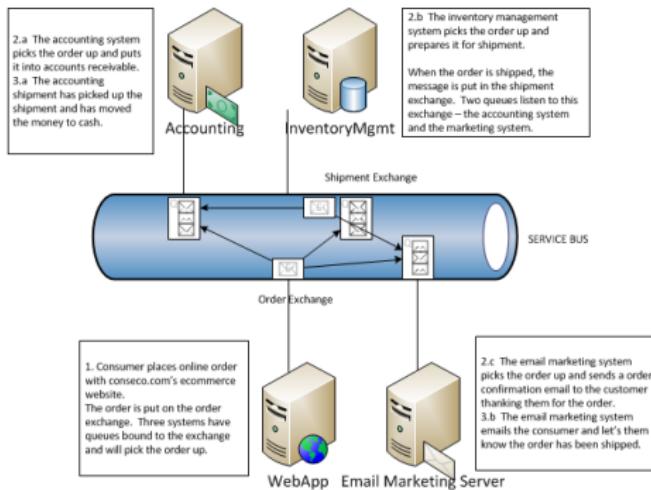
Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos

Arquiteturas

Projeto



Fonte
Mais

Sistemas Pervasivos/Ubíquos I

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos

Arquiteturas

Projeto

"Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user".(Weiser 1993).

Foco na tarefa em vez de na ferramenta.

Sistemas Pervasivos/Ubíquos II

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto

- Detecção de contexto.
- Composição ad-hoc
- Compartilhamento

Smart Life
Amazon Go
Reality check

Sistemas Pervasivos/Ubíquos III

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

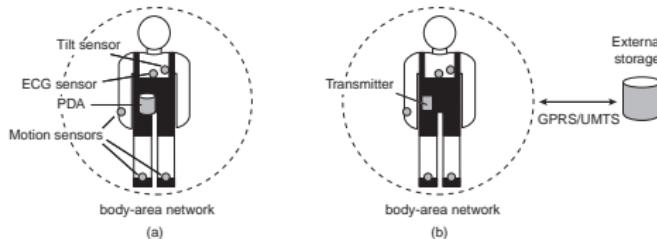
Tipos
Arquiteturas

Projeto

- **Internet das coisas**
- Realidade aumentada
- Smart grid
 - E.g., lavadora que escolhe horário

Sistemas Pervasivos/Ubíquos IV

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto



- Segurança e privacidade
- Ghost in the shell? Snow crash?

Redes de Sensores I

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

- Movimentação de tropas e de fauna
- Índices de poluição
- Abalos sísmicos e predição de avalanches
- Indoors e outdoors

Redes de Sensores II

Introdução

Projeto

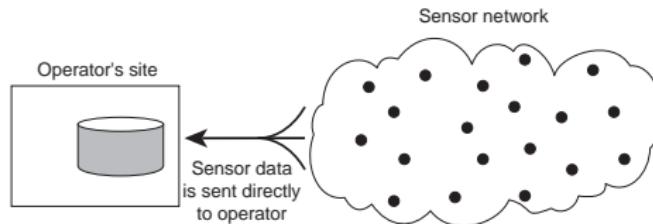
Objetivos e
Desafios

Da pedra ao
smartphone

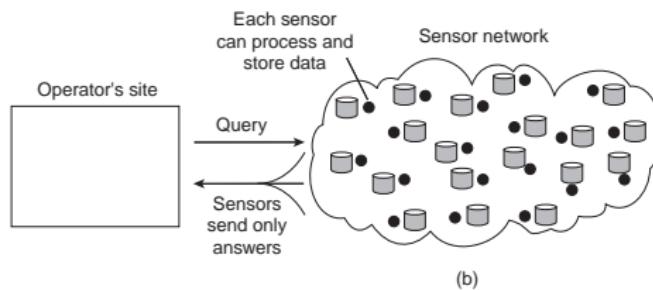
Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



(a)



(b)

Sistemas de Computação

Nuvens

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas
Tipos
Arquiteturas

Projeto

<https://www.google.com/about/datacenters/gallery/#/all>

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto

5 Tipos e Arquiteturas

■ Arquiteturas

Arquitetura I

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

“... an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions (e.g., no cycles). Other constraints—say, having to do with execution semantics—might also be part of the style definition.”

David Garlan and Mary Shaw, January 1994, CMU-CS-94-166,
em “An Introduction to Software Architecture

Arquitetura II

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

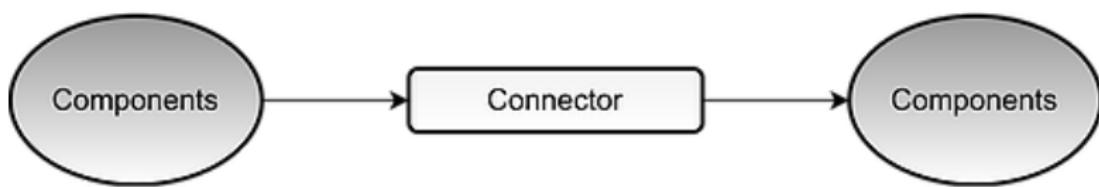
Projeto

Um estilo ou padrão arquitetural é o conjunto de princípios que provê uma infraestrutura abstrata para uma família de sistemas. Promove reuso de projeto ao prover soluções para problemas recorrentes e frequentes.

Fonte

Componentes e Conectores

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto



Layers e Objetos

Introdução

Projeto

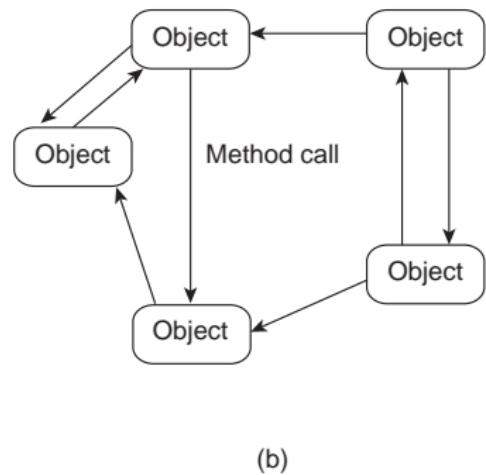
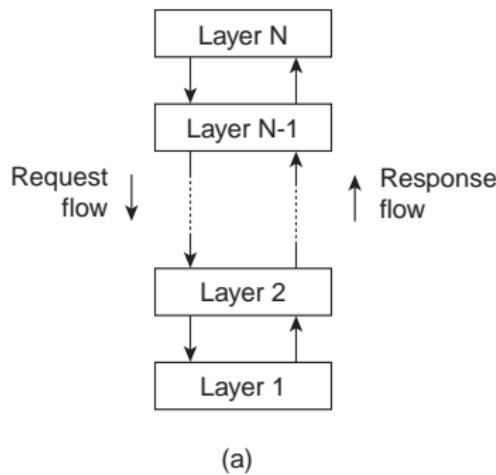
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

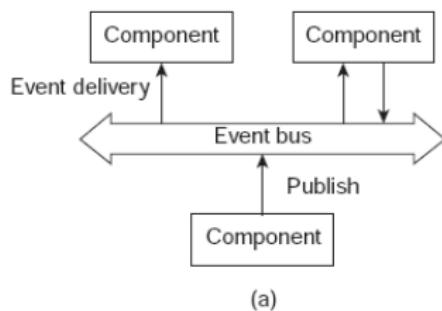


(a)

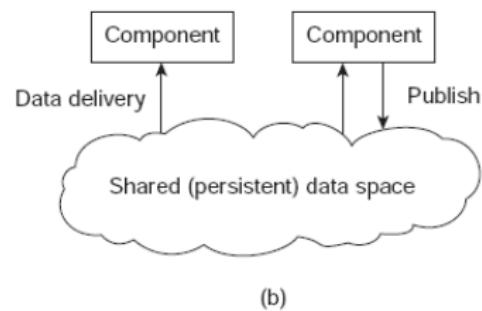
(b)

Desacoplamento Espacial e Temporal

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto



(a)



(b)

Orientação a eventos e a dados.

Introdução
Projeto
Objetivos e
Desafios
Da pedra ao
smartphone
Tipos e
Arquiteturas
Tipos
Arquiteturas
Projeto

Comunicação x Papéis

Cliente/Servidor I

1 x 1 ou n x 1

Introdução

Projeto

Objetivos e
Desafios

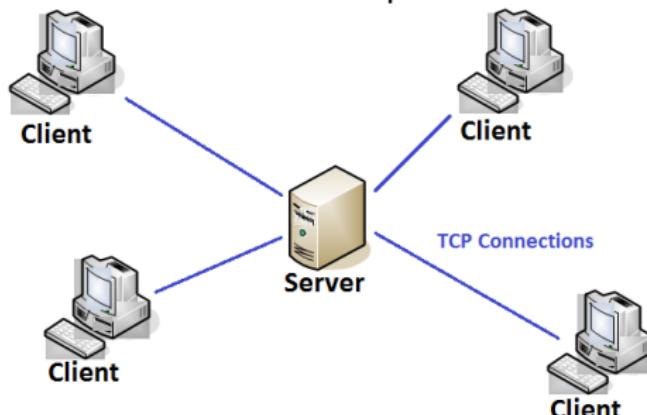
Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

- Servidor: oferece serviço.
- Cliente: usa serviço.
- Possivelmente em máquinas distintas



Fonte

Cliente/Servidor II

1 x 1 ou n x 1

Introdução

Projeto

Objetivos e
Desafios

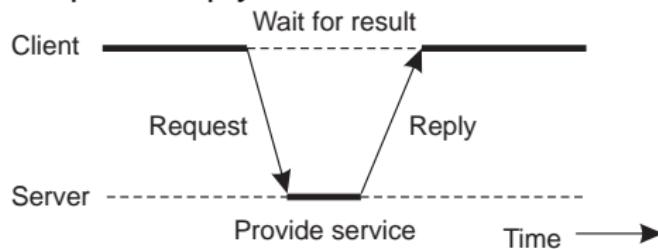
Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

■ Request/Reply



Oportunidade para computação assíncrona!

Cliente/Servidor III

1 x 1 ou n x 1

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas
Tipos
Arquiteturas

Projeto

- Cliente multi-thread:
- Servidor multi-thread: múltiplos clientes
Concorrência, Isolamento, multi-tenancy,

Cliente Servidor

n x n

Introdução

Projeto

Objetivos e
Desafios

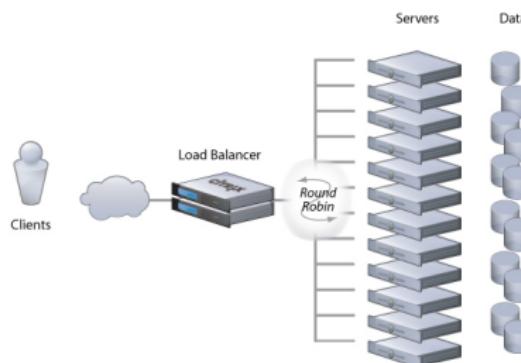
Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

Load Balancing



Fonte

E.g., Sistemas bancários. Netflix.

Peer-2-Peer I

n x n

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

- Colaboradores/Pares, em vez de clientes e servidores.
- E.g., Gnutella, Limewire, Shareaza, Napster...
Todos vão para a cadeia juntos!

Peer-2-Peer II

n x n

Introdução

Projeto

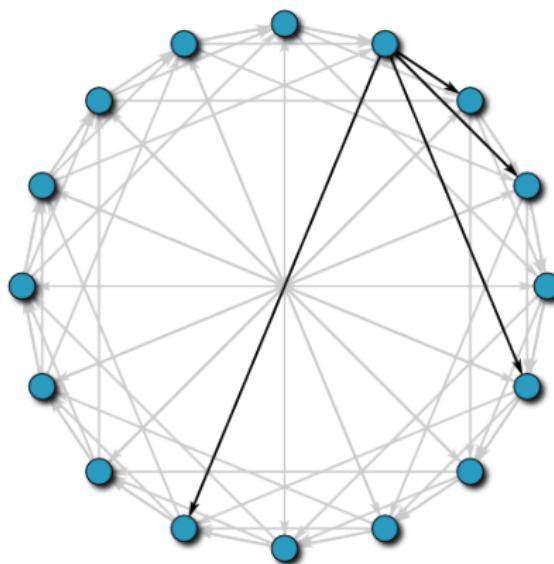
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Híbridos I

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

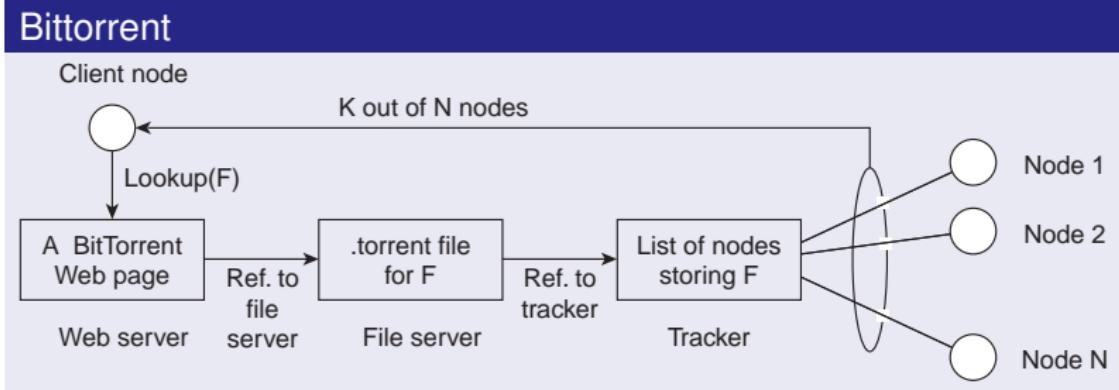
Tipos
Arquiteturas

Projeto

- Clientes, servidores, pares, super-pares, tudo junto e misturado.
- CassandraDB, por exemplo, é um banco de dados P2P, que serve a múltiplos clientes, de diversas aplicações distintas, ao mesmo tempo.
CassandraDB, em quadrinhos!
- Massively Multiplayer Online Games.

Híbridos II

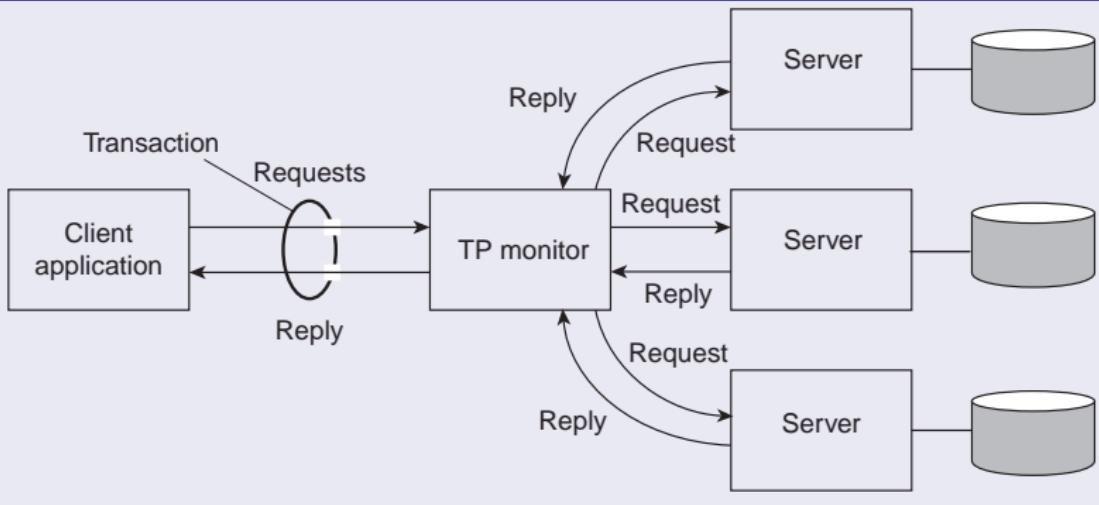
Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto



Híbridos III

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto

Sistema de informação



Tiers I

3-Tiers

Presentation tier



Application tier

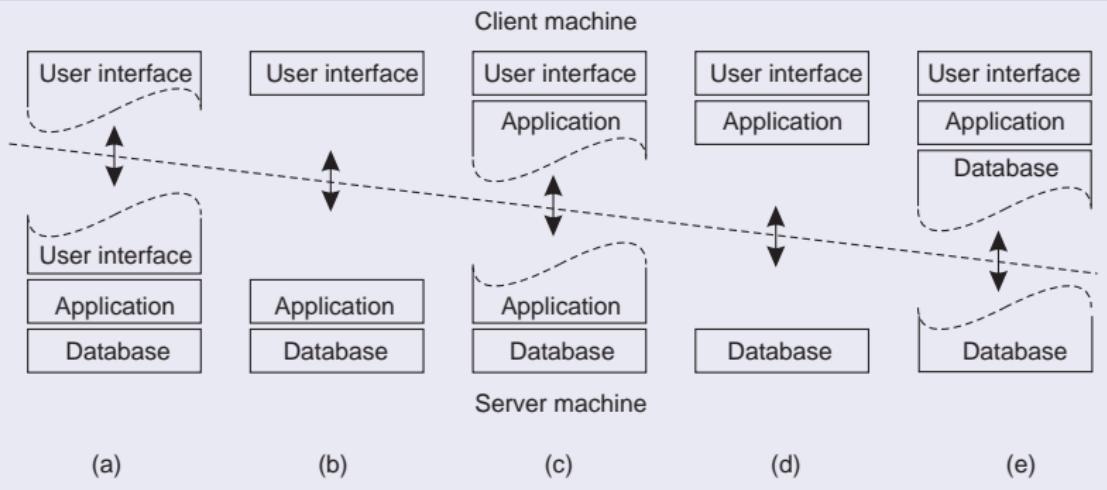
Data tier

Fonte

Tiers II

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto

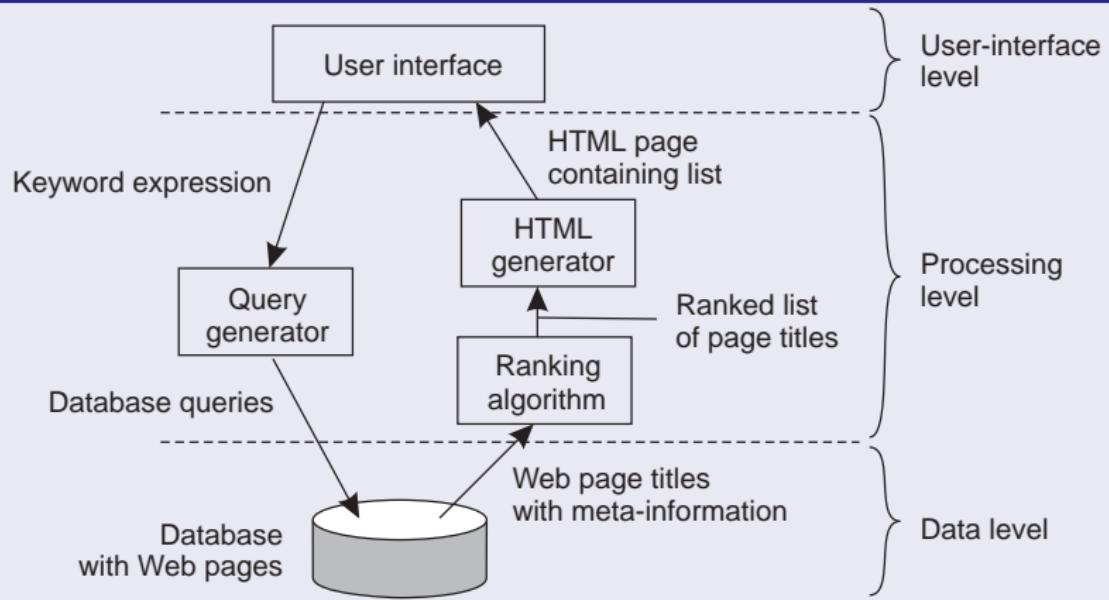
2-Tiers



Tiers III

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos Arquiteturas
Projeto

Múltiplos tiers



Tiers IV

Microserviços

Introdução

Projeto

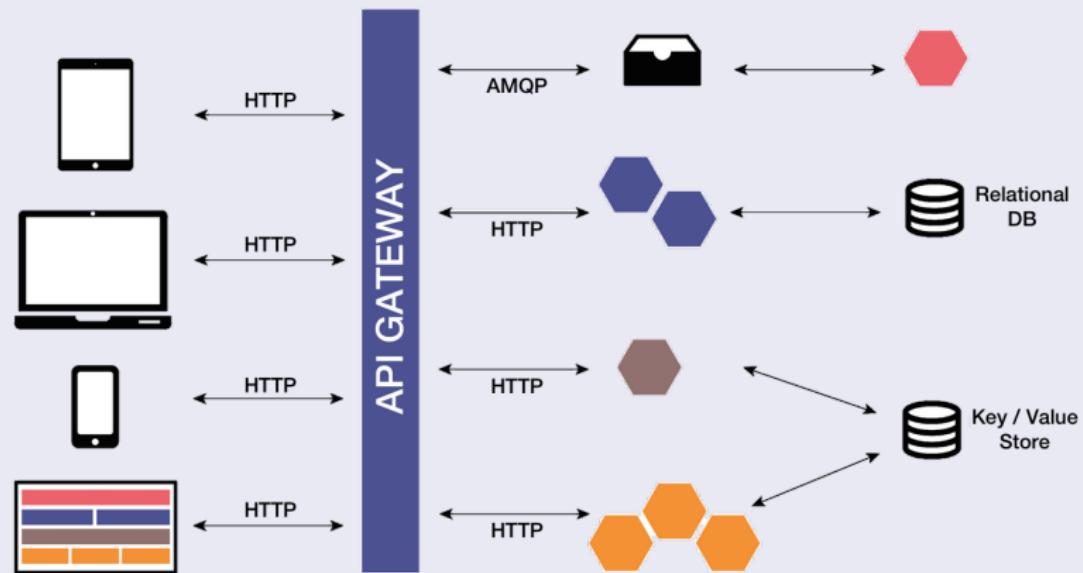
Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto



Tiers V

Microserviços

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

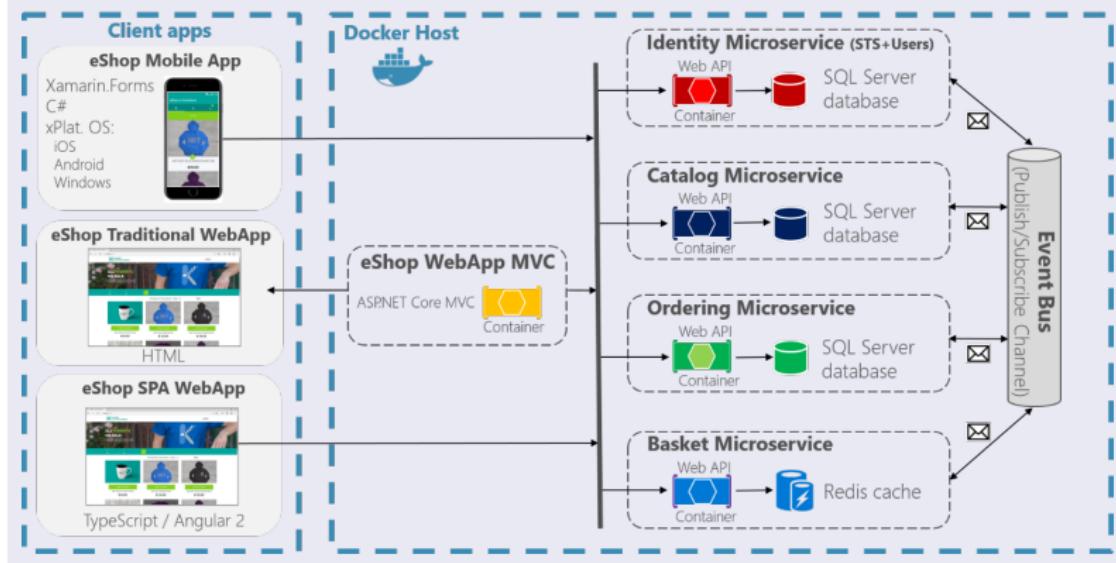
Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

“eShopOnContainers” Reference Application

Microservices Architecture



Fonte: desconhecido.

Tiers VI

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos

Arquiteturas

Projeto

Leia mais

Microserviços

A rede por trás

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Tipos
Arquiteturas

Projeto

Facebook Fabric

<https://www.youtube.com/watch?v=mLEawo60zFM>

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Tipos
Arquiteturas
Projeto

[Leia mais sobre arquitetura](#)

Introdução
Projeto
Objetivos e
Desafios
Da pedra ao
smartphone
Tipos e
Arquiteturas
Tipos
Arquiteturas
Projeto

Fim da aula 3

O projeto

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto
Entrega 1

O objetivo deste projeto é desenvolver um sistema de bancos de dados genérico/multiuso, a ser usado como bloco de construção em outros projetos.

Com este objetivo, replicaremos abordagens bem conhecidas e funcionais, aplicando diversas técnicas de desenvolvimento de sistemas distribuídos.

O projeto será dividido em 3 entregas, definidas abaixo.

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto
Entrega 1

6 Projeto

■ Entrega 1

Projeto 2018 I

Entrega 1

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

Para a primeira entrega vocês desenvolverão a “cara” do banco de dados, permitindo que clientes se conectem e realizem operações de acordo com a especificação da API. Desenvolverão também um cliente em linha de comando para que se possa manipular o banco de dados, bem como um cliente de testes, que estressará o banco para verificar sua corretude e funcionalidades.

O cliente interativo

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

Leitura de comandos

- 1 thread em loop infinito apresentando menu de comandos e lendo comandos do teclado
- uma vez digitado um comando, o mesmo é validado *localmente*
- se válido, comando é enviado ao servidor
- se inválido, mensagem de erro é apresentada
- o comando *sair* termina a execução deste thread

O cliente interativo

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

Apresentação de respostas

- 1 thread em loop infinito recebendo mensagens do servidor
- uma vez recebida uma mensagem, a mesma é apresentada na tela
- uma vez terminado o thread de leitura de comandos, o thread de apresentação espera pelo menos 5 segundos por novas mensagens do servidor e então se termina, terminando o processo.

O servidor

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Entrega 1

A base de dados

- é um mapa de BigInteger (inteiro de precisão infinita) para um vetor de bytes (ou algo que o valha)
- mantido em memória apenas (por enquanto)
- e manipulado por 4 operações (**CRUD**)
- observando a semântica de cada operação (create \neq update)
- todos os comandos retornam um código de erro/sucesso *mais* resposta, se for adequado.

Apesar do banco ser em memória, toda operação será logada em disco.

O servidor

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

Terá arquitetura em estágios, tendo

- 1 ou mais threads recebendo comandos e colocando em uma fila F1
- 1 thread consumindo comandos de F1 e colocando cópias do comando em uma fila F2 e em outra fila F3
- 1 thread consumindo comandos de F2 e gravando-os em disco.
- 1 thread consumindo de F3 aplicando o comando no banco de dados (mapa)

O thread de log

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

grava comandos em um arquivo de log

- mantendo o arquivo aberto durante a execução do programa
- adicionando comandos sempre ao fim do arquivo
- somente se o comando altera a base de dados (Reads são descartados)

O thread de processamento

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto
Entrega 1

executa os comandos

- contra o mapa
- emitindo mensages de sucesso (create/update/delete)
- respondendo com informação solicitada (read)
- emitindo erros quando adequado (create/update/delete/read)
- na ordem em que os comandos foram enfileirados em F3

Filas

Introdução
Projeto
Objetivos e Desafios
Da pedra ao smartphone
Tipos e Arquiteturas
Projeto
Entrega 1

são estruturas de dados com semântica bem definida.

- listas não são filas
- pilhas não são filas
- arrays não são filas
- embora possam ser usados para implementar filas

Tolerância a falhas

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

Como o mapa é mantido em memória, no caso de falhas, todo o banco apagado. Para recuperá-lo

- Na reinitialização do processo
- abra o arquivo de log
- e processe-o na sequência em que foi escrito
- reexecutando todas as operações gravadas
- antes de aceitar novas requisições de clientes.

Acesso concorrente

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto

Entrega 1

Diversos clientes podem ser iniciados em paralelo e contactando o mesmo servidor.

Comunicação

Introdução

Projeto

Objetivos e
Desafios

Da pedra ao
smartphone

Tipos e
Arquiteturas

Projeto
Entrega 1

- Toda comunicação é feita via **TCP**.
- E o canal de comunicação com o cliente é mantido aberto enquanto o mesmo estiver executando.
- **Todas** as portas usadas na comunicação são especificadas via arquivos de configuração.

Programando SD

RPC

Thrift

gRPC

Parte II

Comunicação

1 Programando Sistemas Distribuídos

■ Redes

Estabelecendo comunicação

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Canal de comunicação: meio pelo qual componentes em um sistema distribuído se comunicam.
- Protocolo: especificação de como o canal deve ser usado de forma que componentes se entendam.

Canal de Comunicação

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Ponto-a-ponto
 - Eficiente
 - Inviável para muitos nós
- Compartilhado
 - Colisões
 - Menor custo
 - Roteamento mais simples

Protocolo

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Acesso ao meio
- Transmissão
- Colisões (Evitar ou tratar)

Roteamento I

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

■ Packet Switching

- Dados divididos em pacotes
- Cada pacote viaja independentemente
- Pacotes são perdidos
- Latência variável

Roteamento II

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

■ Circuit switching

- Caminho dedicado
- Recursos reservados
- Pacotes de tamanho fixo
- Latência constante

Roteamento III

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Esta complexidade é abstraída dos desenvolvedores.

A Internet

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Internetworking Protocol
- Redes subjacentes são abstraídas
- Melhor esforço
- Roteadores conectam as redes.

A Internet

Programando SD

Redes

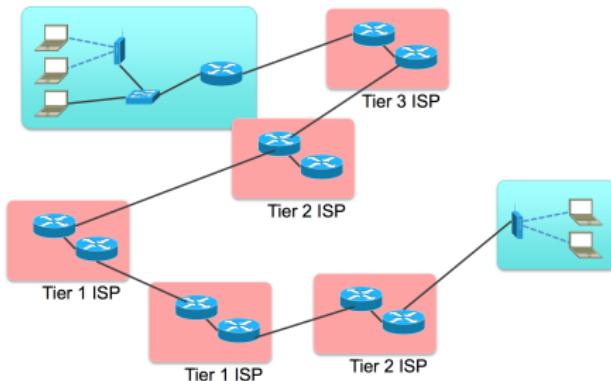
Sockets

IP-Multicast

RPC

Thrift

gRPC



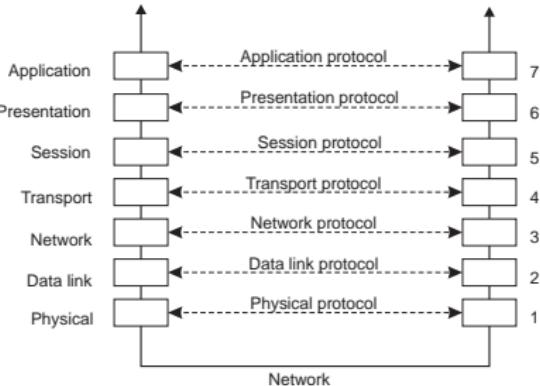
OSI

Programando SD
Redes
Sockets
IP-Multicast

RPC

Thrift

gRPC



- Bits
- Frames/quadros; controle de fluxo; acesso ao meio.
- Datagramas/pacotes; roteamento
- Controle de fluxo; fim a fim; confiabilidade; tcp e udp
- Streams/fluxos; conexões lógicas; restart; checkpoint; http, ssl
- Objetos; json, xml; criptografia
- Aplicações; http, pop, ftp
- Headers das camadas superiores são dados nas camadas inferiores

OSI

Programando SD

Redes

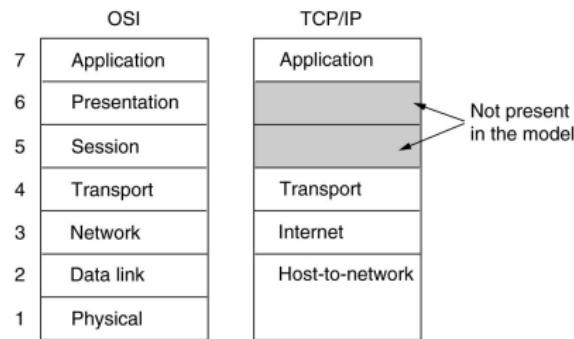
Sockets

IP-Multicast

RPC

Thrift

gRPC



Fonte

- Camadas 1,2,3 e 4 – equivalentes
- Camadas 5, 6 e 7 – aplicação
- Bibliotecas/middleware provêm o restante das funcionalidades
 - (De)Serialização
 - Nomeamento
 - Criptografia
 - Replicação
 - Invocação remota de procedimentos
 - ...

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

1 Programando Sistemas Distribuídos

■ Sockets

Hosts

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Cada interface tem um endereço MAC

Hosts

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Cada interface tem um endereço MAC – Somente comunicação direta
- Cada interface tem um endereço IPv4/IPv6

Hosts

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Cada interface tem um endereço MAC – Somente comunicação direta
- Cada interface tem um endereço IPv4/IPv6 – 32×128 bits

Hosts

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Cada interface tem um endereço MAC – Somente comunicação direta
- Cada interface tem um endereço IPv4/IPv6 – 32 x 128 bits
- Como falar com uma aplicação?

Sockets

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Pontos finais da comunicação
- Porta: 16 bits
 - IANA [Internet Assigned Numbers Authority](#)
 - Bem conhecidas – 0-1023
 - Proprietárias – 49151
 - Dinâmicas – 65535
- Domínio: AF_INET (Internet), PF_UNIX, PF_X25..., PF_INET (Internet),
- Tipo: SOCK_STREAM x SOCK_DGRAM (TCP x UDP)
- Utilizado como um arquivo
- Protocolo: por sua conta

Sockets

Programando SD

Redes

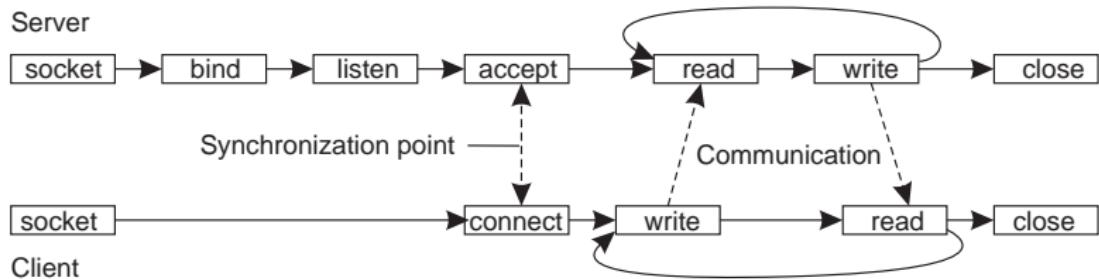
Sockets

IP-Multicast

RPC

Thrift

gRPC



Exemplo

server.py (nao usar socket.py)

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
1 #!/usr/bin/python                                # This is server.py file
2
3 import socket                                    # Import socket module
4
5 s = socket.socket()                            # Create a socket object
6 host = ' ' #socket.gethostname() # Get local machine name
7 port = 12345                                     # Reserve a port for your service.
8 s.bind((host, port))                           # Bind to the port
9
10
11 s.listen(5)                                    # Now wait for client connections.
12 while True:
13     c, addr = s.accept()                         # Establish connection with client.
14     print 'Got connection from', addr
15     c.send('Thank you for connecting')
16     c.close()                                    # Close the connection
```

Execução

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- `python server.py`
- `telnet localhost 12345`
- `netcat localhost 12345`

Exemplo

Servidor.java

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
1 ...
2     Socket s = new ServerSocket(port);
3 ...
```

Exemplo

client.py

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
1  #!/usr/bin/python                      # This is client.py file
3
4  import socket                           # Import socket module
5
6  s = socket.socket()                     # Create a socket object
7  host = '' # socket.gethostname() # Get local machine name
8  port = 12345                            # Reserve a port for your service.
9
10 s.connect((host, port))
11 data = s.recv(1024)
12 print data
13 s.close                                # Close the socket when done
```

Execução

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

■ `python client.py`

Exemplo

Client.java

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
1 ...
2     Socket s = new Socket(hostname,port);
3 ...
```

Encerrando a conexão

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

`socket.close()` encerra a conexão do lado de quem invoca. Na contraparte, invocações a `socket.recv()` retornam com 0 bytes lidos.

Exercício: ping-pong

Programando SD
Redes
Sockets
IP-Multicast
RPC
Thrift
gRPC

Modifique cliente e servidor tal que o cliente envie uma mensagem passada na linha de comando ao servidor e fique esperando uma resposta, e tal que o servidor fique esperando uma mensagem e então solicite ao operador que digite uma resposta e a envie para o cliente. O loop continua até que o usuário digite SAIR, e a conexão seja encerrada.

Terminal 1

```
python server.py
Esperando conexão.
Esperando mensagem.
Mensagem recebida: lalala
Digite resposta: lelele
Resposta enviada.
Conexão encerrada.
Esperando conexão.
```

Terminal 2

```
python client.py
Digite mensagem: lalala
Mensagem enviada.
Esperando resposta.
Resposta recebida: lelele
Digite mensagem: SAIR
Desconectando.
```

Leitura do teclado

- `x = raw_input()` – Python 2
- `x = input()` – Python 3

UDP – em Python

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Em Python, um servidor UDP não executa `listen` OU `accept`, mas simplesmente `data, addr = sock.recvfrom(4096)`, onde

- `data` é o conteúdo recebido e
- `addr` o endereço de quem enviou o datagrama.

`addr` pode ser usado para enviar uma resposta, como em

```
sent = sock.sendto(data, addr)  
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

Exercício

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Modifique o código do exercício anterior para usar UDP em vez de TCP na comunicação entre nós.

Pegadinhas na comunicação

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- falta de ordem
- falta de confiabilidade
- menos dados lidos que enviados.
- mais dados lidos que enviados (pode acontecer no TCP)
- para que serve UDP então?

Referências

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

pymotw.com/2/socket/udp.html

www.tutorialspoint.com/python/python_networking.htm

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Fim da aula 4

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

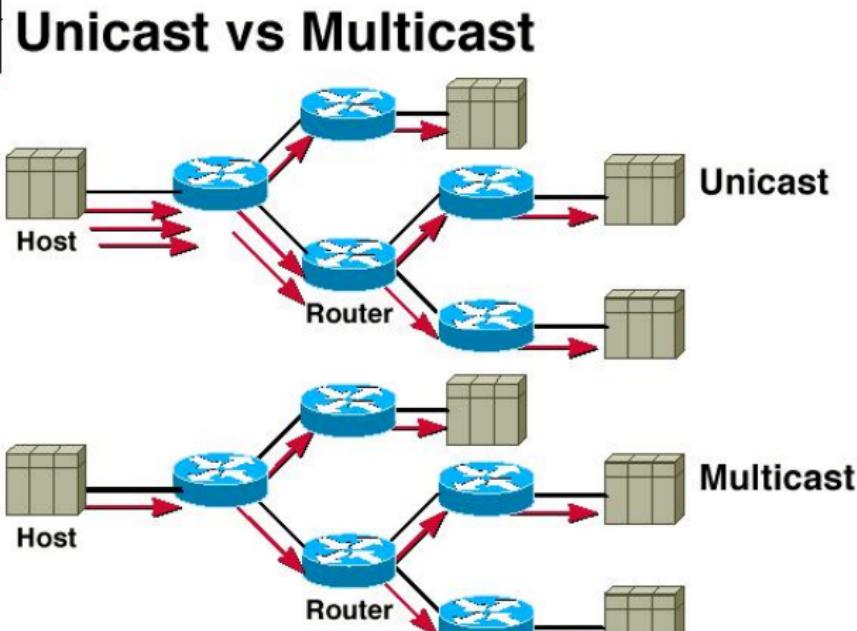
1 Programando Sistemas Distribuídos

■ IP-Multicast

IP-Multicast

Programando SD
Redes
Sockets
IP-Multicast

RPC
Thrift
gRPC



Copyright © 1998, Cisco Systems, Inc.

IP-Multicast

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- UDP
- Mensagem entregue a todos que se juntaram ao grupo.
- Grupo identificado por IP Classe D
(224.0.0.0-239.255.255.255)

{
224.0.0.0 to
239.255.255.255}

Addresses	Use
224.0.0.0 - 224.0.0.255	Local Multicast (not forwarded by L3 devices)
224.0.1.0 - 224.0.1.255	Routed Multicast (forwarded by L3 devices)
232.0.0.0 - 232.255.255.255	Source Specific Applications
233.0.0.0 - 233.255.255.255	GLOP Addressing
239.0.0.0 - 239.255.255.255	Private addressing

Fonte

Servidor

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

- Criar Socket UDP
- Uni-lo a um grupo
- Receber pacotes.

IP-Multicast I

MReceiver.java

Programando SD
Redes
Sockets
IP-Multicast
RPC
Thrift
gRPC

```
import java.io.*;
2 import java.net.*;

4 public class MReceiver {
    public static void main(String[] args) {
        byte[] inBuf = new byte[256];
        try {
            MulticastSocket socket = new MulticastSocket(8888);
            InetAddress address = InetAddress.getByName("224.2.2.3");
            socket.joinGroup(address);
            while (true) {
                DatagramPacket inPacket = new DatagramPacket(inBuf, inBuf.length);
                socket.receive(inPacket);
                String msg = new String(inBuf, 0, inPacket.getLength());
                System.out.println("From " + inPacket.getAddress() + " Msg : " + msg);
            }
        }catch (IOException ioe) {
            System.out.println(ioe);
        }
    }
}
```

IP-Multicast

MSender.java

```
1 import java.io.*;
2 import java.net.*;
3 public class MSender {
4     public static void main(String[] args) {
5         byte[] outBuf;
6         final int PORT = 8888;
7         try {
8             DatagramSocket socket = new DatagramSocket();
9             long counter = 0;
10            while (true) {
11                counter++;
12                outBuf = ("Multicast numero " + counter).getBytes();
13                InetAddress address = InetAddress.getByName("224.2.2.3");
14                DatagramPacket outPacket =
15                    new DatagramPacket(outBuf, outBuf.length, address, PORT);
16                socket.send(outPacket);
17                System.out.println("Server sends : " + msg);
18                try { Thread.sleep(500); } catch (InterruptedException ie) {}
19            }
20        } catch (IOException ioe) { System.out.println(ioe); }
21    }
22}
```

Referências

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

lycog.com/programming/multicast-programming-java/

Exercício

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Modifique o código que desenvolveu em Python para que, em vez de usar “localhost” como endereço, use o endereço multicast 224.2.2.4.

IPv6

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

“In IPv6, the left-most bits of an address are used to determine its type. For a multicast address, the first 8 bits are all ones, i.e. FF00::/8. Further, bit 113-116 represent the scope of the address, which can be either one of the following 4: Global, Site-local, Link-local, Node-local.

In addition to unicast and multicast, IPv6 also supports anycast, in which a packet can be sent to any member of the group, but need not be sent to all members.”

Fonte

Multicast Server – Em Python (3)

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
import socket
2 import struct
4 MCAST_GRP = '224.1.1.1'
5 MCAST_PORT = 5007
6
7 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
8 sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
9 sock.bind((MCAST_GRP, MCAST_PORT))
10 mreq = struct.pack("=4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)
11 #4 bytes (4s) seguidos de um long (1), usando ordem nativa (=)
12
13 sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
14
15 while True:
16     print(sock.recv(10240).decode())
```

Fonte

Multicast Client – Em Python (3)

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

```
1 import socket
2 MCAST_GRP = '224.1.1.1'
3 MCAST_PORT = 5007
4
5 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
6 sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 2)
7 sock.sendto(input().encode(), (MCAST_GRP, MCAST_PORT))
```

Fonte

Programando SD

Redes

Sockets

IP-Multicast

RPC

Thrift

gRPC

Fim da aula 5

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

2 Invocação Remota de Procedimentos

- Abaixo os Sockets!

Comunicação Baseada em Sockets

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Acesso baixo nível à rede... e nada mais.
- “arquivo” em que se lê e escreve bytes
- Controle de fluxo é por conta da aplicação
 - send envia x bytes e recv recebe $y \leq x$ bytes
 - send envia x e depois y bytes e recv recebe $x + y$
- Codificação é por conta da aplicação
- Controle de erro é por conta da aplicação

Dados complexos

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

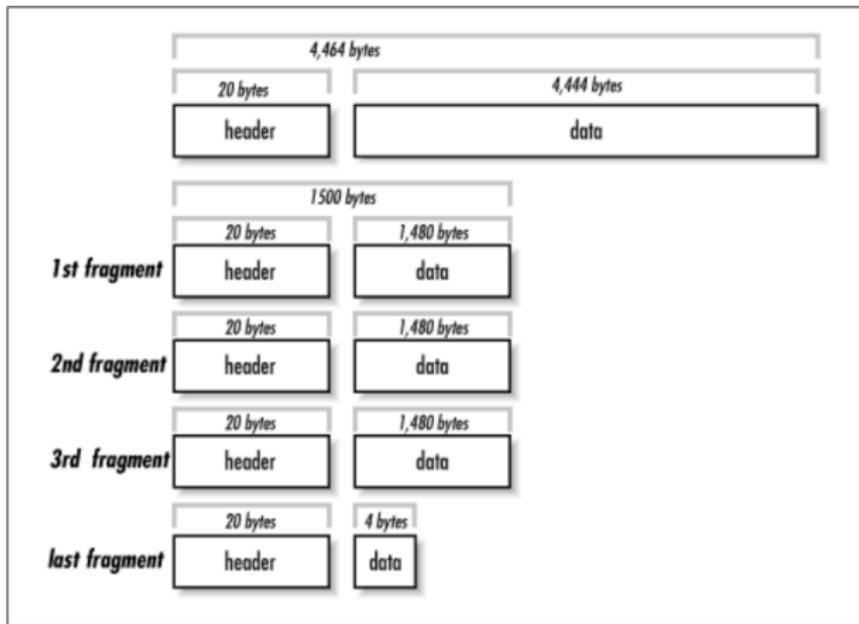
Thrift

gRPC

Imagine-se enviando uma struct com diversos campos, incluindo números de diversos tipos, strings, estruturas aninhadas, tudo somando vários k-bytes.

Fragmentação

Programando SD
RPC
Abaixo os Sockets!
RPC
Frameworks
Thrift
gRPC



Fonte

Codificação

Programando SD
RPC
Abaixo os Sockets!
RPC
Frameworks
Thrift
gRPC

- Tamanho de Tipos
32 x 64
- Ordem de bytes
IP usa big-endian (MSB no endereço menor)
 - Big endian: SPARC (< V9), Motorola, PowerPC
 - Little endian: Intel x64, IA-32
 - Bi-endian: ARM, MIPS, IA-64
- Representação de ponto flutuante
- Conjunto de caracteres
- Alinhamento de bytes
- Linguagens
Estruturas x Classes
- Sistemas operacionais
(DOS) crlf x lf (Unix)

Codificação

Protocolos textuais: leitura de linhas

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

■ Amigável à humanos

```
telnet www.google.com 80
Trying 187.72.192.217...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
host: www.google.com

HTTP/1.1 302 Found
Location: http://www.google.com.br/?gws_rd=cr&ei=HTDqWJ3BDYe-wATs_a3ACA
Cache-Control: private
Content-Type: text/html; charset=UTF-8
P3P: CP="This is not a P3P policy! See https://www.google.com/support/accounts/answer/151657?hl=en for more
Date: Sun, 09 Apr 2017 12:59:09 GMT
Server: gws
Content-Length: 262
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=100=NBAruuFWL0hXk2-h7VDduH0_UkjAr6RaqqG7VbccTsfLzFfhxEKx21Xpa2EH7IgshgcE9vU4W1TyKsa07wQeu

<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.com.br/?gws_rd=cr&ei=HTDqWJ3BDYe-wATs_a3ACA">here</A>.
</BODY></HTML>
```

■ Espaçoso

Comunicação Baseada em Sockets

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- read/write
- programadores gostam de funções

Comunicação Baseada em Sockets

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- read/write
- programadores gostam de funções
- funções em sistemas distribuídos?

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

2 Invocação Remota de Procedimentos

- RPC

Remote Procedure Calls

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Birrel e Nelson (1984): mecanismo para invocação de procedimentos em outros processos/máquinas.

```
x = substring(a,3,"teste"); //procura "teste" em a, a partir do indice 3.
```

Onde `substring` está em outra máquina.

Invocação Local

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

```
x = substring(a,3,"teste"); //procura "teste" em a, a partir do indice 3.
```

- coloque o endereço de "teste" na pilha.
- coloque 3 na pilha.
- coloque o valor de a na pilha.
- salte para substring (dados de controle também irão na pilha)
- ...
- coloque resultado no acumulador
- limpe a pilha
- salte de volta
- coloque resultado em x

Invocação remota

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Não há pilha compartilhada entre chamador e chamado.

Invocação Remota I

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Simula invocação local usando invocações locais de funções que usam sockets.

- Stub/Skeleton

Invocação Remota II

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Stub cliente: função `substring(char*, int, char*)` que
 - abre socket para servidor
 - envia parâmetros
 - especifica função
 - espera resposta
 - retorna resultado
- Stub servidor (skeleton)
 - espera conexão
 - recebe parâmetros
 - recebe especificação da função
 - invoca função localmente
 - envia resultado para cliente

Invocação Remota III

aplicação cliente

aplicação servidor

stub cliente (proxy)

stub servidor
(skeleton)

SO

SO

rede

Marshalling: representar parâmetros de forma própria para transmissão "no fio".

Stub cliente

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Marshalling e Unmarshalling
- Comunicação

Stub servidor/Skeleton

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Multiplexação de clientes
- Marshalling e Unmarshalling

Benefícios

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Interface baseada em procedimentos
- Mais fácil escrever aplicações distribuídas (portas? sockets? codificação?)
- Camada de sessão 5: gerenciamento de conexões
- Camada de apresentação 6: marshalling
- Tudo é transparente!!!!

Transparência

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Tudo corre como se a execução fosse local, de forma transparente ao programador.

Transparência

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Tudo corre como se a execução fosse local, de forma transparente ao programador.

Será?

- passagem de parâmetro por referência?
- onde se conectar?
- segurança
- latência
- como lidar com erros?

Passagem por referência

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Passagem por referência passa o valor de um endereço.
- Como skeleton acessa um endereço no cliente?
 - Requisita dado ao cliente
 - Dado é enviado
 - Dado é modificado
 - Dado é retornado
 - Dado é sobreescrito.
- Como passar um grafo como parâmetro?
- Java serializa todo o grafo.

Onde se conectar?

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Páginas amarelas (Birel e Nelson)
- Quem administra?
- Mesmo serviço em múltiplos lugares?

Onde está meu servidor?

Banco de dados Distribuído

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Cada servidor mantém sua própria lista
- Como se descobrem?

Cadê minha resposta?

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Latência
- Quando parar de esperar?

Tratamento de erros

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Se servidor quebra, cliente fica esperando
- Erro de conexão
- Não há transparência total em sistemas distribuídos

Tratamento de erros

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Invocação local: exatamente 1 vez
- Invocação remota:
 - 0: falhas
 - 1: tudo funcionou
 - ≥ 1 : retransmissão

Múltiplas Invocações

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Frameworks RPC oferecem
 - at least once
 - at most once
- Entenda suas funções
 - idempotente: pode ser invocada múltiplas vezes ($x = 10$)
 - não idempotente: múltiplas invocações tem efeitos distintos ($x += 10$)
- Projete para idempotência!

Concorrência

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Múltiplos clientes acessando o mesmo servidor
- Dados compartilhados
- Controle de concorrência

Outros fatores

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Latência

- Segurança

- Conexão precisa de sigilo?
- Dados precisam de sigilo?
- Autenticação e autorização

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

2 Invocação Remota de Procedimentos

■ Frameworks

Linguagens

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Sem RPC: C, C++, Java < 5.0 (1.5), Python
- Com RPC: Java, Go, Erlang, Scala, Haskell
- Ambientes heterogêneos: Thrift, gRPC, Akka, SOAP

IDL: Interface Definition Language I

Programando SD

RPC

Abaixo os Sockets!

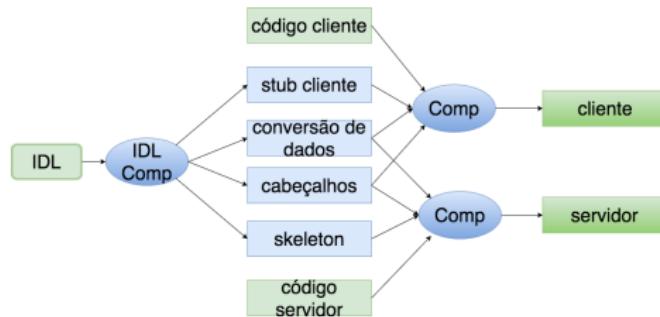
RPC

Frameworks

Thrift

gRPC

- Linguagem de definição de
 - serviços acessados remotamente
 - estruturas de dados usadas no serviços
- Pre-compilador
 - Gera stubs e skeletons



Cliente

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Inicializar RPC
 - Tipo de transporte
 - SSL?
 - Localizar servidor
- Lidar com falhas

Servidor

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Nadie, nill, zip, nulla!

Servidor

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

Nadie, nill, zip, nulla! Em geral...

Como o sistema RPC é construído?

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Naming: exportar e localizar serviços
- Gerenciamento de portas
- Conexões

Serialização

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Converter para array de bytes
- ASN.1 (Abstract Syntax Notation – ISO)
- XDR (eXternal Data Representation)
- JSON (JavaScript Object Notation)
- XML (Extensible Markup Language)
- Google Protocol Buffers
- Thrift
- Java serialization

JSON

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Tipos implícitos
- Tipos explícitos

```
{  
2   "id": 1,  
3   "name": "A green door",  
4   "price": 12.50,  
5   "tags": ["home", "green"]  
6 }
```

XML/JSON

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Verborrágico
- Amigável a humanos
- Auto descriptivo
- Fácil de fazer parsing
- Largamente suportado
- Questão de opinião
- JSON é XML sem a gordura.

Google Protocol Buffers I

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- Dados come estrutura pré-definida
- Eficiente e rápido
- Agnóstico à linguagem
- Mensagens são nomes e tipos
- Compilação de mensagens para linguagens específicas gera código para manipular dados
- Largamente utilizado dentro e fora do Google.

Google Protocol Buffers II

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }
  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }
  repeated PhoneNumber phone = 4;
}
```

Google Protocol Buffers III

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

```
1 Person person;
2 person.set_name("John Doe");
3 person.set_id(1234);
4 person.set_email("jdoe@example.com");
5 fstream output("myfile", ios::out | ios::binary);
6 person.SerializeToOstream(&output);
```

Google Protocol Buffers IV

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

```
fstream input("myfile", ios::in | ios::binary);
2 Person person;
person.ParseFromIstream(&input);
4 cout << "Name: " << person.name() << endl;
cout << "E-mail: " << person.email() << endl;
```

Google Protocol Buffers V

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

```
1 <person>
    <name>John Doe</name>
3     <email>jdoe@example.com</email>
</person>
```

“When this message is encoded to the protocol buffer binary format, it would probably be 28 bytes long and take around 100-200 nanoseconds to parse. The XML version is at least 69 bytes if you remove whitespace, and would take around 5,000-10,000 nanoseconds to parse.”

Fonte: Dev guide

Referências

Programando SD

RPC

Abaixo os Sockets!

RPC

Frameworks

Thrift

gRPC

- <https://www.madboa.com/geek/openssl/#cs-smtp>
- <https://developers.google.com/protocol-buffers/docs/overview>
- <https://www.cs.rutgers.edu/~pxk/417>

Thrift

Programando SD

RPC

Thrift

Instalação

IDL

gRPC

<https://thrift.apache.org/>

Programando SD

RPC

Thrift

Instalação

IDL

gRPC

Fim da aula 6

Programando SD

RPC

Thrift

Instalação

IDL

gRPC

3 Estudo de Caso RPC: Thrift

- Instalação

Instalação

Programando SD

RPC

Thrift

Instalação

IDL

gRPC

- Baixe e compile o thrift: <http://www.apache.org/dyn/closer.cgi?path=/thrift/0.10.0/thrift-0.10.0.tar.gz>
- ou instale-o usando apt-get, por exemplo. (apt-get install thrift-compiler)
- execute "thrift" na linha de comando
- Para thrift com java, também precisarão dos seguintes arquivos
 - <http://mvnrepository.com/artifact/org.slf4j/slf4j-api/1.7.21>
 - <https://sites.google.com/site/lasaro/sistemasdistribuidos/libthrift0.9.3.jar>
- coloque-os na pasta “jars”

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

3 Estudo de Caso RPC: Thrift

■ IDL

- Arquitetura
- Exemplo

Interface Definition Language I

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

■ Tipos básicos

- bool: boolean (true/false)
- byte: 8-bit; inteiro sinalizado
- i16: 16-bit; inteiro sinalizado
- i32: 32-bit; inteiro sinalizado
- i64: 64-bit; inteiro sinalizado
- double: 64-bit; ponto-flutuante
- string: string UTF-8
- binary: sequência de bytes

Interface Definition Language II

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

■ Estruturas

```
struct Example {  
    1:i32 number,  
    2:i64 bigNumber,  
    3:double decimals,  
    4:string name="thrifty"  
}
```

Interface Definition Language III

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

■ Serviços

```
service ChaveValor {  
    void set(1:i32 key, 2:string value),  
    string get(1:i32 key) throws (1:KeyNotFound knf),  
    void delete(1:i32 key)  
}
```

■ Não se pode retornar NULL!!!

Interface Definition Language IV

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

■ Exceções

```
exception KeyNotFound {
    1:i64 hora r,
    2:string chaveProcurada="thrifty"
}
```

■ Containers

- List
- Map
- Set

IDL

chavevalor.thrift

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

```
1 namespace java chavevalor
2 namespace py chavevalor
3
4 exception KeyNotFound
5 {
6 }
7
8
9
10 service ChaveValor
11 {
12     string getKV(1:i32 key) throws (1:KeyNotFound knf),
13     bool setKV(1:i32 key, 2:string value),
14     void delKV(1:i32 key)
15 }
```

lab/thrift/chavevalor.thrift.txt

Geração

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

Compilação

```
thrift --gen java chavevalor.thrift  
thrift --gen py chavevalor.thrift
```

Handler

ChaveValorHandler.java

```
1 package chavevalor;
2
3 import org.apache.thrift.TException;
4 import java.util.HashMap;
5 import chavevalor.*;
6
7 public class ChaveValorHandler implements ChaveValor.Iface {
8     private HashMap<Integer, String> kv = new HashMap<>();
9
10    @Override
11    public String getKV(int key) throws TException {
12        if(kv.containsKey(key))
13            return kv.get(key);
14        else
15            throw new KeyNotFound();
16    }
17    @Override
18    public boolean setKV(int key, String valor) throws TException {
19        kv.put(key, valor);
20        return true;
21    }
22    @Override
23    public void delKV(int key) throws TException {
24        kv.remove(key);
25    }
26}
```

Arquitetura

Programando SD
RPC
Thrift
Instalação
IDL
Arquitetura
Exemplo
gRPC

- Runtime library – componentes podem ser selecionados em tempo de execução e implementações podem ser trocadas
- Protocol – responsável pela serialização dos dados
 - TBinaryProtocol
 - TJSONProtocol
 - TDebugProtocol
 - ...
- Transport – I/O no “fio”
 - TSocket
 - TFramedTransport (non-blocking server)
 - TFileTransport
 - TMemoryTransport
- Processor – Conecta protocolos de entrada e saída com o *handler*
- Handler – Implementação das operações oferecidas

Thrift II

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

- Server – Escuta portas e repassa dados (protocolo) para os processors
 - TSimpleServer
 - TThreadPool
 - TNonBlockingChannel

ChaveValorServer.java I

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

```
1 package chavevalor;  
2  
3 import org.apache.thrift.server.TServer;  
4 import org.apache.thrift.transport.TServerSocket;  
5 import org.apache.thrift.transport.TServerTransport;  
6 import org.apache.thrift.server.TServer.Args;  
7 import org.apache.thrift.server.TSimpleServer;  
8 import org.apache.thrift.server.TThreadPoolServer;  
9  
10 // Generated code  
11 import chavevalor.*;  
12  
13 import java.util.HashMap;  
14  
15 public class ChaveValorServer {  
16     public static void main(String [] args) {  
17         try {  
18             TServer server =  
19                 new TSimpleServer(  
20                     new Args(  
21                         new TServerSocket(9090))  
22                         .processor(new ChaveValor.Processor(  
23                                         new ChaveValorHandler()  
24                                     )));  
25     }  
26 }
```

ChaveValorServer.java II

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

```
27     System.out.println("Starting the simple server...");  
28     server.serve();  
29 } catch (Exception x){  
30     x.printStackTrace();  
31 }  
32 }  
33 }
```

lab/thrift/ChaveValorServer.java

ChaveValorClient.java I

Programando SD
RPC
Thrift
Instalação
IDL
Arquitetura
Exemplo
gRPC

```
1 package chavevalor;
2
3 import chavevalor.*;
4
5 import org.apache.thrift.TException;
6 import org.apache.thrift.transport.TTransport;
7 import org.apache.thrift.transport.TSocket;
8 import org.apache.thrift.protocol.TBinaryProtocol;
9 import org.apache.thrift.protocol.TProtocol;
10
11 public class ChaveValorClient {
12     public static void main(String [] args) {
13         try {
14             TTransport transport = new TSocket("localhost", 9090);
15             transport.open();
16
17             TProtocol protocol = new TBinaryProtocol(transport);
18             ChaveValor.Client client = new ChaveValor.Client(protocol);
19
20             int k1 = 1;
21             String v1 = "lalaal";
22
23             client.setKV(k1,v1);
24             System.out.println(client.getKV(k1));
25             client.delKV(k1);
```

ChaveValorClient.java II

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

```
27     System.out.println(client.getKV(k1));  
28  
29     transport.close();  
30 } catch (TException x) {  
31     x.printStackTrace();  
32 }  
33 }
```

lab/thrift/ChaveValorClient.java

classpath

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

```
javac  
-cp jars/libthrift0.9.3.jar:jars/slf4japi1.7.21.jar:gen-java  
-d .  
*.java
```

```
java  
-cp jars/libthrift0.9.3.jar:jars/slf4japi1.7.21.jar:gen-java:.  
chavevalor.ChaveValorServer
```

```
java  
-cp jars/libthrift0.9.3.jar:jars/slf4japi1.7.21.jar:gen-java:.  
chavevalor.ChaveValorClient
```

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

- Baixe e compile o thrift: <http://www.apache.org/dyn/closer.cgi?path=/thrift/0.10.0/thrift-0.10.0.tar.gz>
- ou instale-o usando apt-get, por exemplo.

Referências

Programando SD

RPC

Thrift

Instalação

IDL

Arquitetura

Exemplo

gRPC

`http://thrift-tutorial.readthedocs.org/en/latest/index.html`

gRPC

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

<https://grpc.io>

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

4 Estudo de Caso RPC: gRPC

■ Instalação

Instalação – Java I

Programando SD
RPC
Thrift
gRPC
Instalação
IDL gRPC &
Protobuffers

A instalação depende da distribuição. No caso do Java, não há instalação propriamente dita.

- Seguiremos este tutorial:

<https://grpc.io/docs/quickstart/java.html>

- Baixe os exemplos:

```
git clone -b v1.19.0 https://github.com/grpc/grpc-java
```

- cd grpc-java

- Certifique-se de que está no branch correto:

```
git checkout v1.19.0
```

- cd examples

- Compile os exemplos assim, se estiver na UFU:

```
./gradlew -Dhttp.proxyHost=proxy.ufu.br -Dhttp.proxyPort=3128 -Dhttps.  
proxyHost=proxy.ufu.br -Dhttps.proxyPort=3128 installDist
```

Instalação – Java II

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

- Execute servidor e cliente de acordo com o tutorial.

- `./build/install/examples/bin/hello-world-server`
- `./build/install/examples/bin/hello-world-client`

./src/main/proto/helloworld.proto

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

```
// The greeting service definition.  
2 service Greeter {  
    rpc SayHello (HelloRequest) returns (HelloReply) {}  
4 }  
  
6 message HelloRequest {  
    string name = 1;  
8 }  
  
10 message HelloReply {  
    string message = 1;  
12 }
```

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Observe que há apenas um serviço e um método. Múltiplos métodos podem ser colocados no mesmo serviço.

src/main/proto/helloworld.proto

Modificado

Observe as mudanças no arquivo.

```
service Greeter {  
    rpc SayHello (HelloRequest) returns (HelloReply) {}  
    rpc DigaOla (OlaRequest) returns (OlaReply) {} // <<<<=====  
}  
  
message HelloRequest {  
    string name = 1;  
}  
  
message HelloReply {  
    string message = 1;  
}  
  
message OlaRequest {      // <<<<=====  
    string name = 1;  
}  
  
message OlaReply {        // <<<<=====  
    string message = 1;  
}
```

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

É boa prática separar requests e responses para cada método, a não ser que não haja dúvida de que serão para sempre iguais.

Re-executando I

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Adicione o método ao servidor

src/main/java/io/grpc/examples/helloworld/HelloWorldServer.java

```
...
2 private class GreeterImpl extends GreeterGrpc.GreeterImplBase {
...
4     @Override
5     public void digaOla(OlaRequest req, StreamObserver<OlaReply>
6             responseObserver) {
7         OlaReply reply =
8             OlaReply.newBuilder().setMessage("Ola " + req.getName()).build();
9         responseObserver.onNext(reply);
10        responseObserver.onCompleted();
11    }
12}
```

Re-executando II

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Adicione a chamada ao cliente

src/main/java/io/grpc/examples/helloworld/HelloWorldClient.java

```
public void greet(String name) {
2   logger.info("Will try to greet " + name + " ...");
...
4   OlaRequest request2 = OlaRequest.newBuilder().setName(name).build();
5   OlaReply response2;
6   try {
7     response2 = blockingStub.digaOla(request2);
8   } catch (StatusRuntimeException e) {
9     logger.log(Level.WARNING, "RPC failed: {0}", e.getStatus());
10    return;
11  }
12  logger.info("Greeting: " + response2.getMessage());
}
```

Re-executando III

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Execute

- `./gradlew installDist`
- `./build/install/examples/bin/hello-world-server`
- `./build/install/examples/bin/hello-world-client`

Percebeu como foi fácil adicionar um método ao serviço? Agora exploremos outras features.

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

Fim da aula 7

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

4 Estudo de Caso RPC: gRPC

- IDL gRPC & Protobuffers

Interface Definition Language I

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

■ Tipos básicos

- bool: boolean (true/false)
- double: 64-bit; ponto-flutuante
- float: 32-bit; ponto-flutuante
- i32: 32-bit; inteiro sinalizado
- i64: 64-bit; inteiro sinalizado
- siXX: signed
- uiXX: unsigned
- sfixedXX: codificação de tamanho fixo
- bytes: 8-bit; inteiro sinalizado
- string: string UTF-8 ou ASCII 7-bit
- Any: tipo indefinido

- Diferentes traduções: <https://developers.google.com/protocol-buffers/docs/proto3>

Interface Definition Language II

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

■ Messages

```
message Person {  
    string name = 1;  
    int32 id = 2;  
    bool has_ponycopter = 3;  
}
```

- Campos tem identificador para casar valor com campo na codificação/decodificação.

Interface Definition Language III

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

■ Serviços

```
1 service ChaveValor {  
2     rpc set (RequestMessageType) returns (ReplyMessageType)  
3 }
```

- "... we as developers are really bad at guessing what we might want in the future. So I recommend being safe by always defining custom params and results types for every method, even if they are empty."

src/main/proto/helloworld.proto

Stream

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Observe as mudanças no arquivo.

```
1 service Greeter {  
2     ...  
3     rpc DigaOlas (OlaRequest) returns (stream OlaReply) {}  
4     ...  
5 }
```

Re-executando I

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Adicione o método ao servidor

src/main/java/io/grpc/examples/helloworld/HelloWorldServer.java

```
1 private class GreeterImpl extends GreeterGrpc.GreeterImplBase {
2     List<String> listOfHi = Arrays.asList("e aih", "ola", "ciao", "bao", "howdy", "s'up");
3
4     @Override
5     public void digaOlas(OlaRequest req, StreamObserver<OlaReply>
6             responseObserver) {
7         for (String hi: listOfHi)
8         {
9             OlaReply reply = OlaReply.newBuilder().setMessage(hi + ", " + req.getName())
10                .build();
11             responseObserver.onNext(reply);
12         }
13         responseObserver.onCompleted();
14     }
15 }
```

Re-executando II

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Adicione a chamada ao cliente

src/main/java/io/grpc/examples/helloworld/HelloWorldClient.java

```
public void greet(String name) {
2   logger.info("Will try to greet " + name + " ...");
3   OlaRequest request = OlaRequest.newBuilder().setName(name).build();
4   try {
5     Iterator<OlaReply> it = blockingStub.digaOlas(request);
6     while (it.hasNext()){
7       OlaReply response = it.next();
8       logger.info("Greeting: " + response.getMessage());
9     }
10  } catch (StatusRuntimeException e) {
11    logger.log(Level.WARNING, "RPC failed: {0}", e.getStatus());
12    return;
13  }
14 }
```

Re-executando III

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &

Protobuffers

Execute

- `./gradlew installDist`
- `./build/install/examples/bin/hello-world-server`
- `./build/install/examples/bin/hello-world-client`

Outros tipos

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

Modifique o request de `digas` para passar uma lista de nomes a serem cumprimentados. Como resultado, `digas` deve cumprimentar todos os nomes com todos os cumprimentos especificados.

Programando SD

RPC

Thrift

gRPC

Instalação

IDL gRPC &
Protobuffers

Fim da aula 8