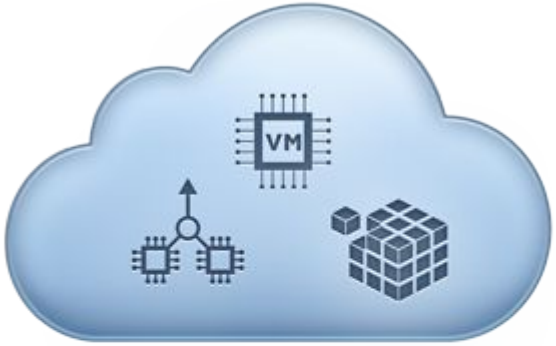
Computação em nuvem

Tecnologias de Suporte à Computação em Nuvem



Prof. Dr. Marcos A. Simplicio Jr.
Laboratório de Arquitetura e Redes de Computadores
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Objetivos – Aula 21

- Entender como o **Google File System** (GFS) facilita o gerenciamento de arquivos em sistemas distribuídos

"Big Data": Google File System (GFS)

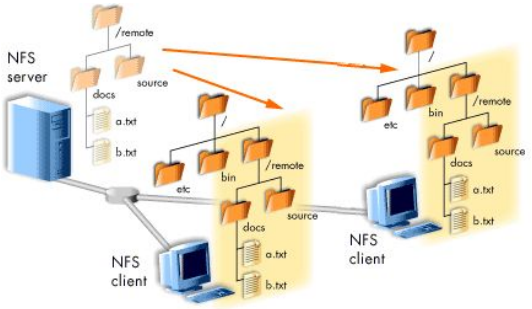
- **MapReduce** requer **sistema de arquivos** adequado
  - Distribuição de dados, tolerância a falhas, redundância ...
- Para atacar problema, o Google desenvolveu o **GFS**
  - Projetado para as necessidades e **cargas de trabalho do Google**
  - Linguagens de programação principais: C++ (base) ou Sawzall (otimizada para MapReduce)
    - Sawzall: código MapReduce ~10x menor que equivalente em C++
  - **Gerenciamento** automático de **tarefas MapReduce** por ferramenta Workqueue

"Big Data": Google File System (GFS)

- Por que não usar sistemas de arquivos existentes?

  - **"Big Data": problemas** neste cenário **são diferentes** dos encontrados no compartilhamento de arquivos por usuários
  - Logo, bom **desempenho** requer algumas "personalizações"

- Por exemplo: Network File System (**NFS**)
  - Sistema de arquivos de **propósito geral**
  - Cenário de uso comum: compartilhar vários usuários



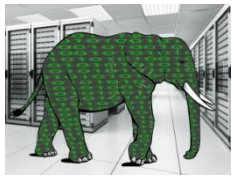
  - **Leituras/escritas em pontos aleatórios**
  - **Muitos arquivos, em geral pequenos**
  - **Sem suporte a redundância** (provida externamente, se preciso)

"Big Data": Cenário do Google/GFS

- Alta taxa de **falhas: redundância necessária**

  - Grande número de componentes de hardware baratos (ex.: discos magnéticos), que comumente acabam falhando

- **Número modesto** de **arquivos gigantescos**



  - Alguns milhões de arquivos tendo 100 MB ou mais (arquivos de GB são comuns)

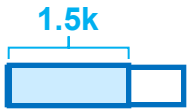
- **Acesso** aos arquivos é "**atípico**"

  - Múltiplas **leituras sequenciais**, seguidas de operações de **anexar** ao final do arquivo (ex.: MapReduce!)
  - **Leituras curtas e aleatórias** ocorrem, mas **são raras**

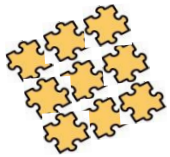
- **Alta vazão** é mais importante do que baixa latência

"Big Data": Projeto do GFS

- Arquivos armazenados em **"chunks" grandes**
  - Tamanho fixo: **64 MB**



  - Isso **facilita leitura sequencial**
  - No **NFS**: o tamanho típico é **2k-8k**



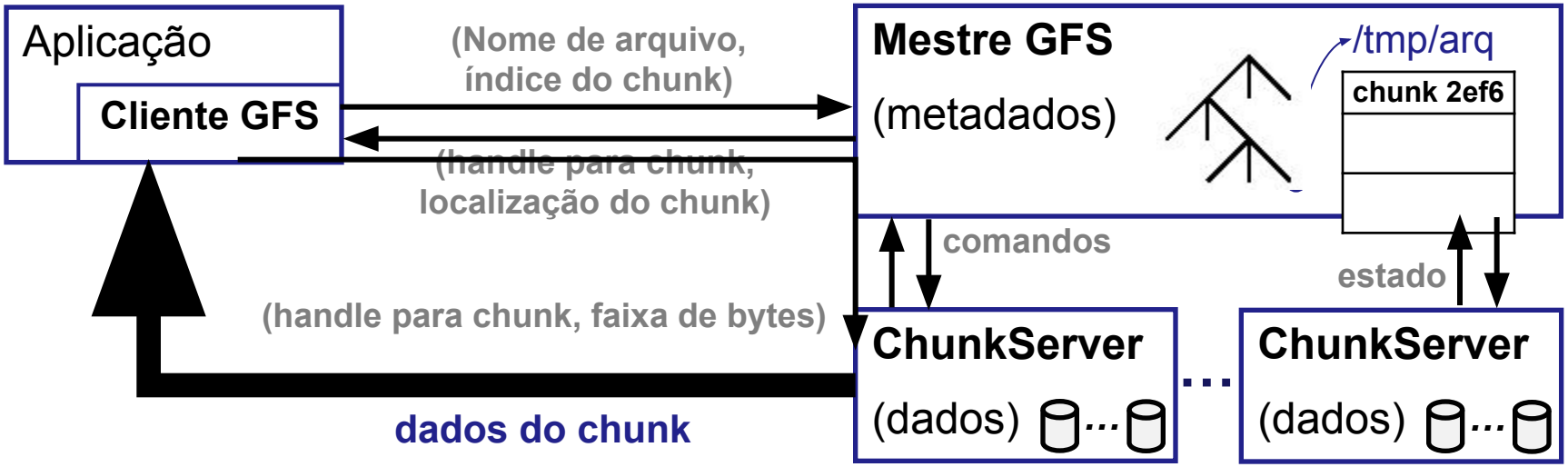
➔ **Tabela de arquivos grande**, armazenada em disco, levando a menor desempenho na leitura de grandes quantidades de dados

- **Sem caching** no cliente
  - Traria poucas vantagens dado o tamanho dos dados e leitura sequencial

  - No **NFS**: **sincronização entre caches** locais nos clientes aumenta **congestionamento na rede**.

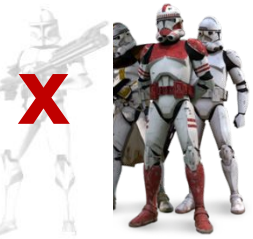
"Big Data": Projeto do GFS (cont.)

- Confiabilidade de dados via **replicação de chunks**
  - Cada chunk armazenado em **3 ou mais ChunkServers**
  - Um único **mestre mantém metadados com localização dos chunks**: controla acesso e permite visão global da rede
  - **Tabela de metadados é pequena** (poucos chunks): pode ficar na **memória principal**, acelerando acesso

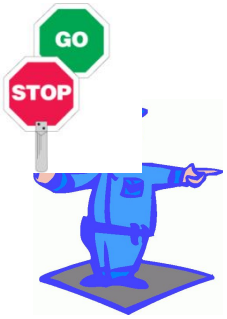


"Big Data": Projeto do GFS (cont.)

- Confiabilidade de dados via replicação de chunks (cont.)
  - Mestre deve garantir número de chunks replicados:



    - Distribuir **nova réplica** quando um ChunkServer **falha**
    - **Remover réplicas** excedentes quando ChunkServer é restabelecido
  - Mestre define ChunkServer "primário" para interagir com clientes

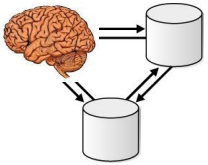


    - Em caso de **alterações** em chunks: cliente contacta diretamente os ChunkServers com as réplicas, mas **primário controla quando alterações são aplicadas**.

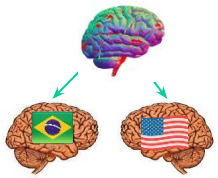
"Big Data": Projeto do GFS (cont.)

- Um único mestre: simples, mas

  - **Ponto único de falha**
  - **Gargalo** para escalabilidade do sistema
- Soluções do GFS:
  - **Replicação** de log de operações do mestre em locais remotos ("shadows"): **restabelecimento rápido em caso de falha**



  - **Envolvimento mínimo do mestre**: após obter metadados, cliente não acessa mestre enquanto lê chunk (grande)
- Obs.: o Google vem trabalhando em solução com **mestre distribuído ("Colossus")**



  - Poucas informações públicas, exceto pela estrutura da base de dados "Spanner": hierarquia por zonas (http://research.google.com/archive/spanner-osdi2012.pdf)

"Big Data": Uso do GFS

- Basicamente todos os sistemas do Google...
  - Mais de 50 clusters, gerenciando petabytes de dados
- Dentre eles: **BigTable**
  - Armazenamento distribuído de dados da ordem de **terabytes**
  - **Dados não estruturados**: armazenamento na forma de mapa indexado por <linha, coluna, timestamp>
  - **Ordenação das chaves por linhas**
  - Cada célula da tabela pode conter **diversas instâncias** de um mesmo arquivo, **diferenciadas pelo timestamp**
  - Admite uso de **locks**: serviço provido por ferramenta "Chubby"

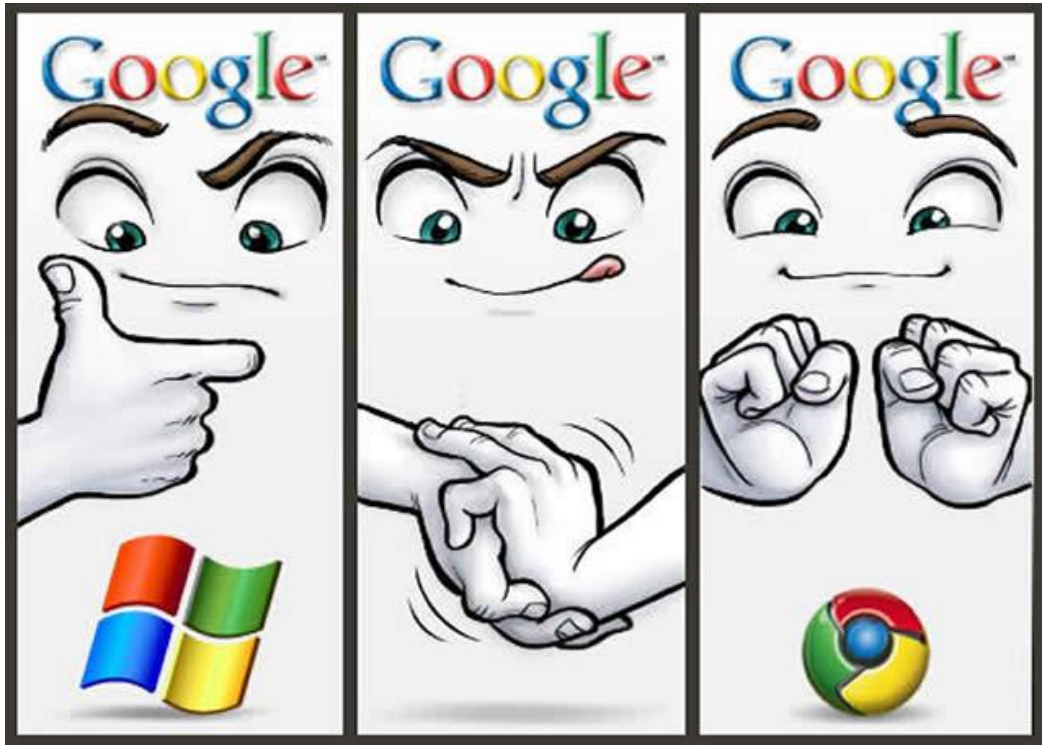
Google BigTable

"Big Data": Hadoop

- **GFS** é sistema **proprietário**, de código fechado
- **Hadoop** Distributed File System (HDFS): **versão de código aberto do GFS**
  - Obtido por **engenharia reversa** do GFS, feita pelo Yahoo!
  - Distribuído pela **Apache**
  - Linguagens de programação principais: **Java** (base) ou **Pig** (otimizada para **MapReduce**)
    - **Pig**: **código MapReduce 20x menor**, requer **16x menos tempo para programar** e executa pouco mais lentamente do que Java.
  - Usa **JobTracker** para **agendar tarefas** de MapReduce e **TaskTracker para executá-las**

Resumo

- Entender como o Google File System (GFS) facilita o gerenciamento de arquivos em sistemas distribuídos
  - **Controle automático de tarefas típicas de big data:** distribuição de dados, tolerância a falhas, redundância de dados
  - Voltado a **armazenamento e acesso sequencial** de **arquivos giganatescos**
  - **Versão aberta** equivalente: **Hadoop**
- Próxima aula: Resumo do Bloco 1
  - Conceitos básicos de nuvem


Google
Google
Google