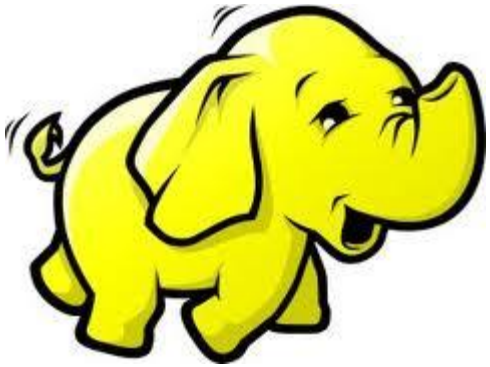
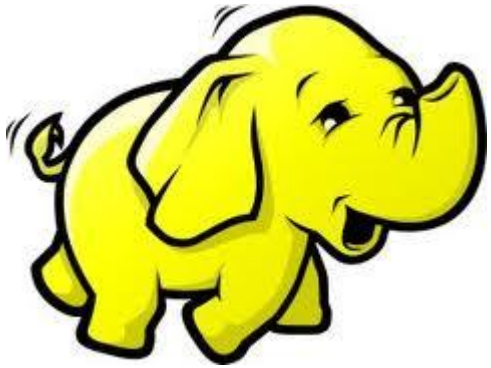


# Apache Hadoop

Bruno Antunes da Silva
UFSCar – Sorocaba

# Agenda

- Introdução
- HDFS
  - Arquitetura
  - Leitura e escrita
  - Distribuição de nós
  - Controle de réplicas
  - Balancer
- MapReduce
- Conclusão



# Introdução

- Aplicações web com grandes quantidades de dados (petabytes)
  - Armazenamento
  - Processamento
  - Escalabilidade
  - Confiabilidade

Armazenamento e processamento distribuído

# Hadoop framework

- Doug Cutting
- Yahoo!

- Computação distribuída
- Escalável
- Segura

YAHOO!

# Hadoop framework

- Petabytes de dados
- Milhares de nós

- Hardware vs Cluster

- Licença Apache

# Vídeo

Using Hadoop to Solve Real Business Problems

# Contribuintes

- Yahoo!
- Google
  - Google File System e MapReduce
- Facebook
- Cloudera
- ...

# Sistema de Arquivos Distribuído Hadoop (HDFS)

- Google File System
- Dados e metadados separados
- Servidor dedicado a metadados
  - NameNode

# Sistema de Arquivos Distribuído Hadoop (HDFS)

- Dados em DataNodes
- Todos servidores são conectados
- Conexão baseada em TCP

# Sistema de Arquivos Distribuído Hadoop (HDFS)

- Não possui mecanismo de proteção de dados
- Dados replicados em 3 diferentes nós
  - Confiabilidade
  - Durabilidade
  - Processamento perto do dado

# Arquitetura do HDFS

- Cluster
- NameNode
- DataNode
- Cliente HDFS
- Image e journal
- CheckpointNode
- BackupNode

# Cluster

Conjunto de servidores

- Um NameNode
- Milhares de DataNodes
- Dezenas de milhares de clientes HDFS

# NameNode

- inodes (metadados)
  - Permissões
  - Horário de acessos e modificações
  - Namespace (hierarquia de arquivos e diretórios)
  - Espaço ocupado em disco.

# NameNode

- Mapear blocos de dados a endereços físicos nos DataNodes
  - Blocos com 128 MB (pode-se alterar)
  - Cada bloco com 3 réplicas (pode-se alterar)
- Gerenciar o número de réplicas

# NameNode

- Cliente requer a leitura de um bloco
  - Devolve o endereço da réplica mais próxima

- Cliente requer a escrita de um bloco
  - Aloca 3 DataNodes para escrita das réplicas (ou um outro número especificado)

# NameNode

- Mantém o namespace na RAM
- *image* de um arquivo
  - inode
  - Lista de blocos nos DataNodes

# NameNode

- *checkpoint*
  - *image* escrita em disco
- *journal*
  - Lista de atualizações de uma *image*

Réplicas dos *checkpoints* e dos *journals* são mantidas em diferentes servidores

# DataNode

- Bloco
  - dados
  - metadados
- Tamanho do arquivo = espaço em disco
- Em cada DataNode de um cluster é gravado o ID do seu namespace

# DataNode

- *block report*
  - Identificador
  - Generation stamp
  - Tamanho

- Envia um *block report* ao NameNode (1 /hora)
  - Atualizar a lista de onde estão as réplicas de seus blocos

# DataNode

- A cada 3 segundos envia um heartbeat ao NameNode
  - Confirmar que ainda está ativo
- Timeout 10 minutos
  - NameNode assume que o DataNode está off
  - Suas réplicas não estão mais disponíveis
  - Cria novas réplicas em outros DataNodes

# DataNode

- *heartbeat*
  - capacidade de armazenamento
  - espaço em uso
  - número de dados sendo transferidos

NameNode utiliza para fazer o balanceamento dos arquivos

# DataNode

- NameNode não envia mensagens ao DataNode
- Responde aos *heartbeats*
  - Replicar blocos para outros nós
  - Remover réplica local de um bloco
  - Refazer o registro ou desativar um nó
  - Enviar um *block report*

# Cliente HDFS

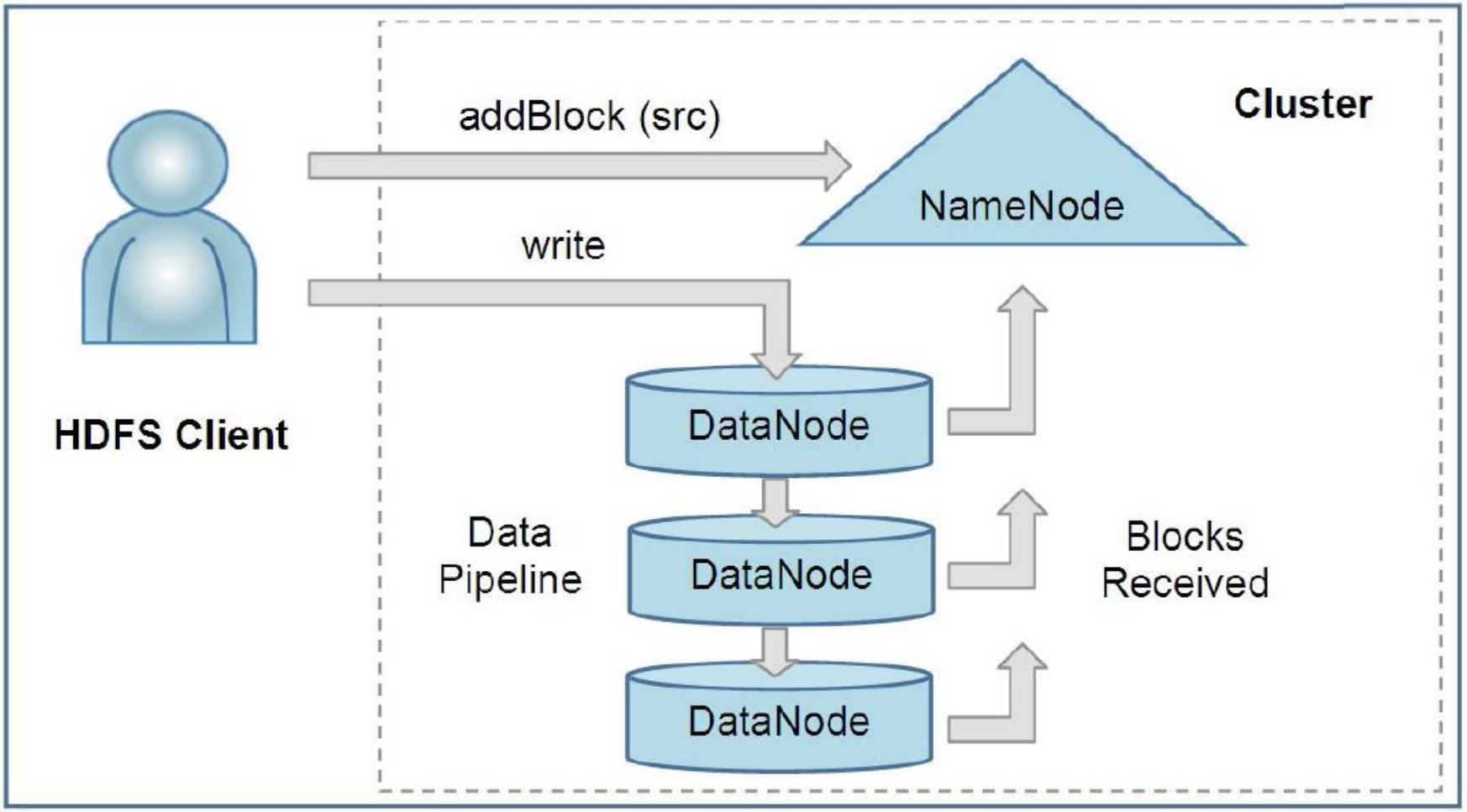
- Interface ao HDFS
  - Arquivos
    - Criação
    - Leitura
    - Escrita
    - Exclusão
  - Diretórios
    - Criação
    - Exclusão
- Abstração da distribuição e replicação

# Cliente HDFS

- Leitura
  - Solicita todos os blocos de um arquivo ao NameNode
  - Recebe a lista de DataNodes
  - Solicita a transferência

- Escrita
  - Solicita um DataNode para escrever um bloco
  - Escreve
  - Solicita DataNode para escrever o próximo bloco

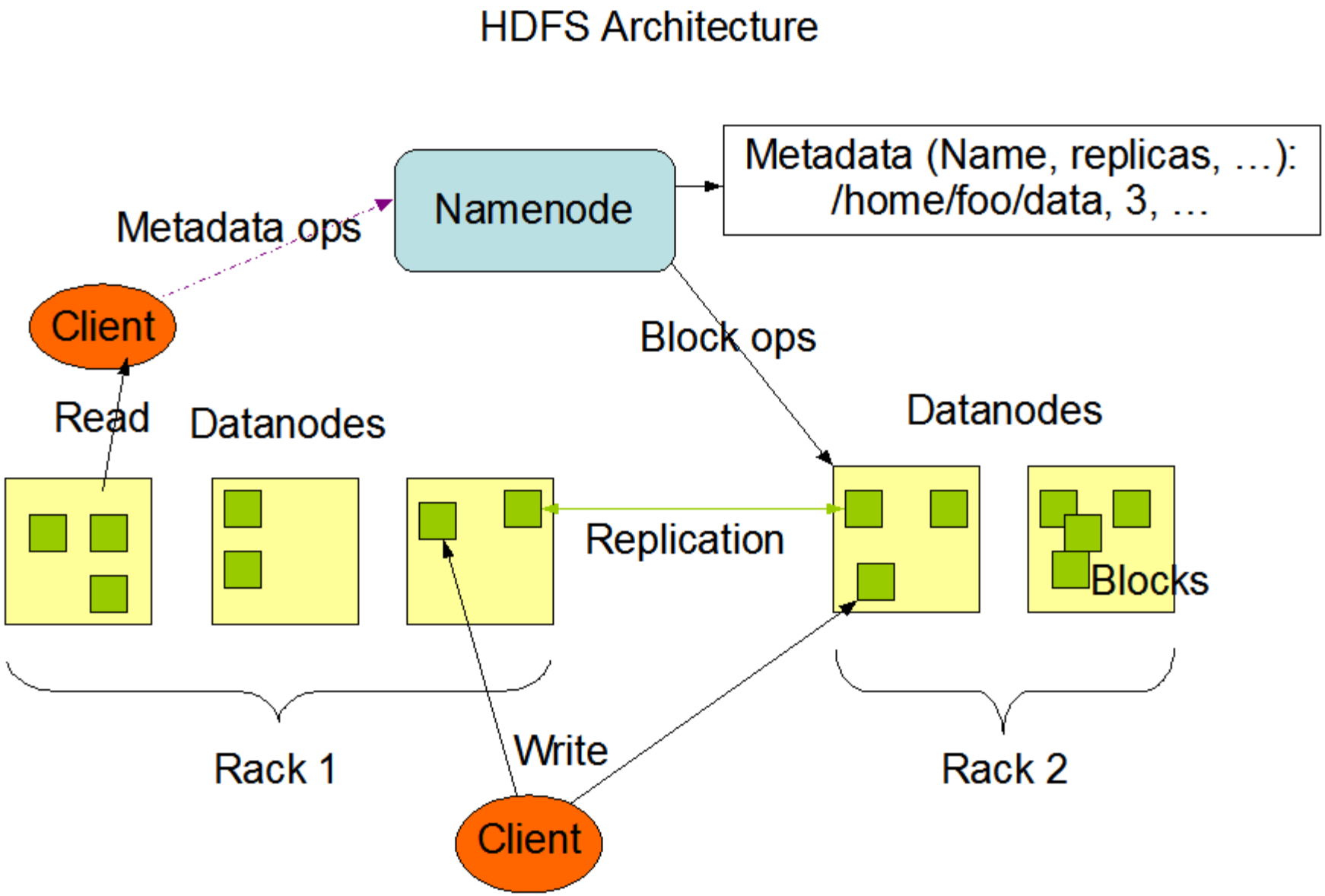
# Cliente HDFS



# Cliente HDFS

- O HDFS fornece funções que retornam o endereço físico dos blocos
  - Permite que aplicações como o MapReduce agendem tarefas onde o dado está
  - Aumenta o desempenho

# Resumo


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

# Funções do NameNode

- Principal
  - Atender a solicitações do cliente e responder DataNodes
- CheckNode
- BackupNode

Definida na inicialização do NameNode

# CheckpointNode

- Combina *checkpoints* e *journals* existentes
- Cria um novo *checkpoint*
- Limpa o *journal*

- Executa em local diferente
- Baixa os *checkpoints* e *journals* do NameNode
- Sincroniza localmente
- Retorna um novo *checkpoint*

# BackupNode

- Contém as funções do CheckpointNode

- Mantém uma *image* do namespace sincronizada com o estado atual do NameNode

# BackupNode

- Recebe o *journal* do NameNode
- Aplica as transações em sua *image*
- Em caso de falha do NameNode
  - Recupera a *image* e o *checkpoint* do BackupNode

# Escrita

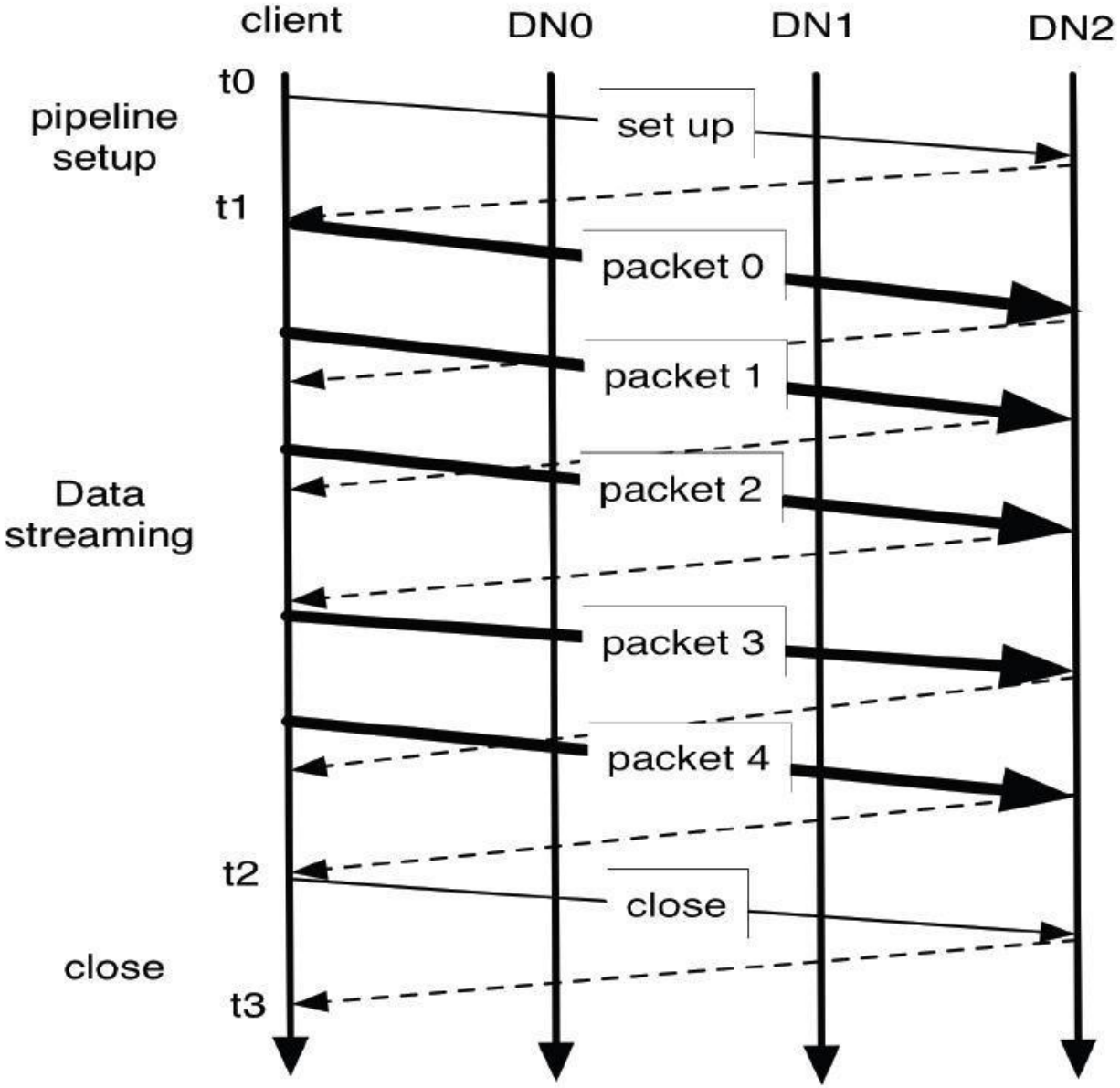
- Escrita
  - Cliente cria um novo arquivo
  - Escreve seu conteúdo
  - Fecha
- O cliente que abre um arquivo para escrita
  - Direito exclusivo de escrever enquanto aberto
  - Envia *heartbeat* para manter permissão
  - Dois tipos de Timeout
    - Curto
    - Longo

# Escrita

- Timeout Curto
  - Outro cliente pode solicitar a permissão de escrita no arquivo
- Timeout Longo
  - HDFS assume que cliente caiu
  - Fecha o arquivo

**A permissão de escrita é exclusiva mas não impede que outro leiam o arquivo**

# Escrita


client
DN0
DN1
DN2
t0
pipeline setup
set up
t1
packet 0
packet 1
Data streaming
packet 2
packet 3
packet 4
t2
close
close
t3

# Arquivo Corrompido

- Criação do arquivo
  - É calculado um *checksum* para cada bloco
  - DataNode armazena o *checksum* com o bloco

# Arquivo Corrompido

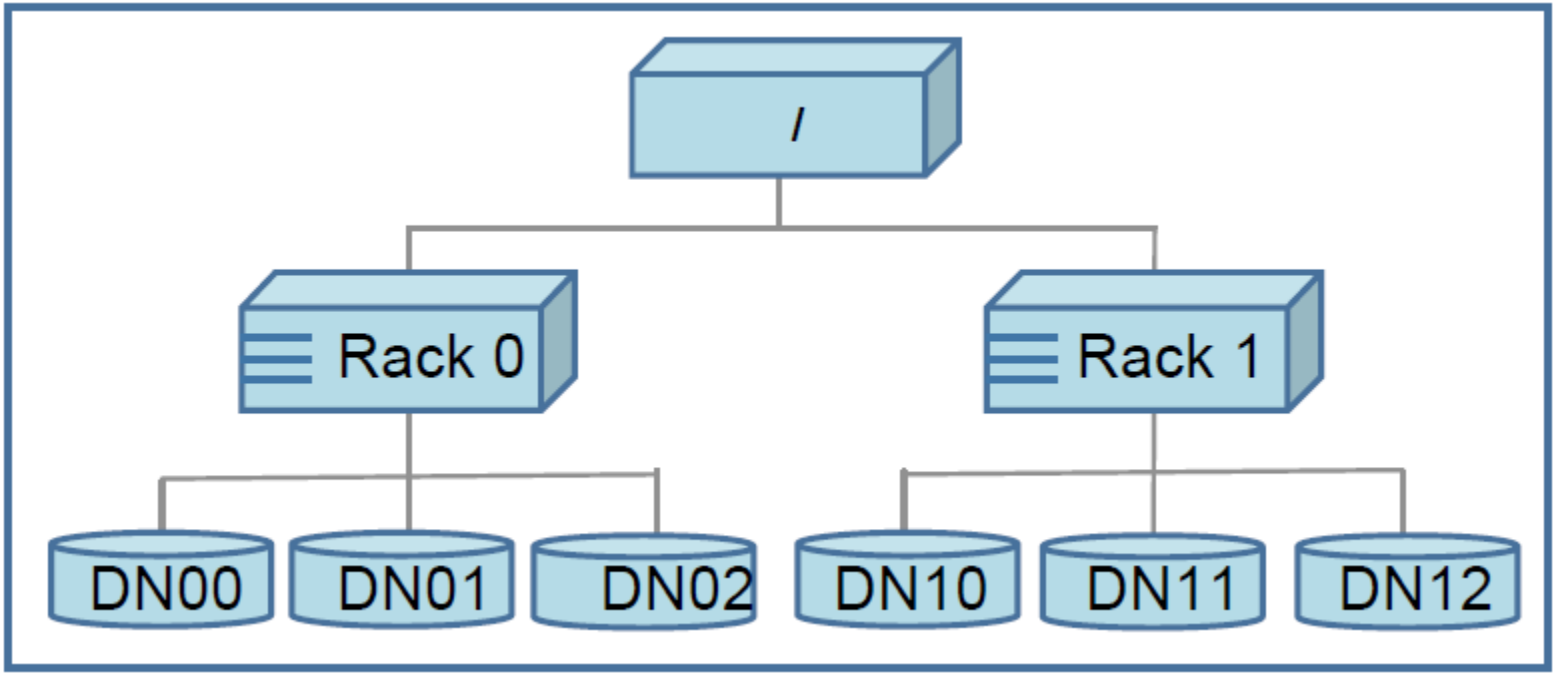
- Cliente recebe um bloco
  - Calcula seu *checksum*
  - Confere com o *checksum* armazenado
- Caso não correto
  - Informa ao NameNode
  - Busca outra réplica

# Distribuição de DataNodes

- Número muito grande de clusters
  → impróprio para conexões lineares

- Divisão em *racks*
  - Clusters em um mesmo rack → mesmo *switch*
  - Racks são conectados por outros *switchs*

# Distribuição de DataNodes



# Distribuição de DataNodes

- Distâncias
  - Distância de um nó a seu pai é um.
  - Distância entre nós
    → Distância dos nós ao seu ancestral mais próximo

# Distribuição de blocos

- Pode ser programada
  - Maior distribuição → Maior confiabilidade
  - Menor distribuição → Maior desempenho
- Padrão
  - DataNode com uma réplica
  - Rack com menos de três réplicas

# Controle de réplicas

- NameNode
  - Detecta falta ou sobra de réplicas
- Sobra
  - Evita exclusão de *rack*
  - Exclui no DataNode com menor espaço livre

# Controle de réplicas

- Falta
  - Fila com prioridade
    - Menor o número de réplicas → Maior prioridade
- Thread de fundo
  - Captura primeiro elemento da lista
  - Escolhe DataNodes
  - Cria suas réplicas

# Controle de réplicas

- Réplicas em um mesmo *rack*
  - Coloca na fila com prioridade
  - Cria réplicas em outro *rack*
  - Exclui réplica no mesmo *rack*

# Balancer

- Equilibra o uso do espaço em disco
- Cluster equilibrado
  - Para cada nó
  - Razão
    - Espaço ocupado do nó
    - Espaço total do nó
  - Razão
    - Espaço ocupado do Cluster
    - Espaço total do Cluster
  - Diferença menor que o limiar [0,1]

# MapReduce

- Framework para escrever aplicações distribuídas
  - Milhares de clusters
  - Confiável
  - Tolerante a falhas
  - Não precisa implementar em Java

# MapReduce

- Divide os dados em blocos para o processamento paralelo
- Distribui as saídas dos *maps* para as tarefas *reduce*
- Gerencia as tarefas
  - Agenda
  - Monitora
  - Re-executa em caso de falha

# MapReduce

- Armazenamento e processamento em um mesmo nó
  - HDFS e MapReduce rodam juntos
- Alta largura de banda no cluster

# MapReduce

- JobTraker (mestre)
  - Agendamento
  - Monitoramento
  - Re-execução em caso de falha
- TaskTraker (escravo)
  - Execução das tarefas

# MapReduce

- Interface ou classe abstrata para determinar
  - Local de entrada e saída
  - Fontes das funções *map* e *reduce*

# MapReduce

- Job cliente que submete o serviço e configuração do JobTraker é responsável
  - Status
  - Diagnóstico

# Conclusão

- Ferramenta de código aberto
- Grandes volumes de dados
  - Escalável
  - Seguro
  - Tratável
  - Baixo custo

# Conclusão

- Baixo custo
  - Super servidor
  - Milhares de máquinas pequenas
- Menor investimento em hardware
  - Confiança no Software

# Conclusão

- Amazon
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

# Perguntas

?