The Hadoop Distributed File System

Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler
Yahoo!
2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)

Apresentado por: Edelberto Franco Silva
Instituto de Computação - Universidade Federal Fluminense (UFF)

Junho, 2012

Agenda

Objetivos

- Objetivos do *Hadoop Distributed File System* (HDFS).
- Armazenar uma grande massa de dados de forma **confiável** e transmiti-las em uma **grande largura de banda** ao usuário.
- Abordagem do artigo: Descrever a arquitetura do HDFS e relatar a experiência de gerência de 25 Petabytes no Yahoo!.

Introdução

- Hadoop é um projeto da Apache.
- Componentes sob licença Apache de código aberto.
- Yahoo! colaborou com 80% do núcleo de HDFS e MapReduce.
- Composição do Hadoop:
  - MapReduce: Mapear a requisição e agregar o resultado;
  - HBase: Base de dados. Provê leitura/escrita em tempo real para base de dados do tipo Big Data.
  - Hive: Infraestrutura de data warehouse.
  - Pig: Linguagem base para análise de grandes massas. Gera diversos MapReduces para excecução em paralelo.
  - ZooKeeper: serviço de coordenação em aplicações distribuídas.
  - Chukwa: sistema de coleta e admnistração de grandes massas de dados.
  - Avro: sistema de serialização de dados. Sincronização de informações dos dados.
  - HDFS: a ser apresentado por este trabalho

HDFS

- Sistema de arquivos distribuídos do Hadoop.
- Com particularidades, mas herda características do UNIX.
- Armazena Metadados e arquivos em locais separados.
  - NameNode: Metadados.
  - DataNodes: Dados.
- Comunicação entre servidores baseado em TCP.
- Não usa RAID para confiabilidade dos dados, mas replicação em diversos DataNodes.

Arquitetura do NameNode

- Namespace hierarquico de arquivos e diretórios (que são representados por *inodes*.
- Conteúdo do arquivo divido em blocos de 128MB e replicado independentemente.
- NameNode: mantém a arvore namespace e o mapeamento de bloco/DataNode.
- Cliente: quem requisitará ao NameNode a localização dos blocos para acesso (premissa de acesso ao DataNode mais próximo).
- Arquitetura composta por 1 NameNode por cluster de DataNode, atendendo a milhares de clientes.

Arquitetura do DataNode

- Responsável por armazenar os blocos de dados.
- Blocos representados por dois arquivos: 1 de dados e 1 metadata (que contém um checksum).
- Blocos são armazenados em seu tamanho exato.
- Ao iniciar o DataNode, é realizado um *handshake* com o NameNode.
  - Sua finalidade é verificar o namespace ID e a versão de software.
- *Storage ID*: permite ao DataNode troca de IP ou porta sem perder a identificação.
- O DataNode envia a cada hora relatório ao NameNode contendo: id dos blocos e timestamp.
  - Heartbeats a cada 3 segundos.
  - Sem Heartbeat por 10 minutos, DataNode *down*.
  - Comunicação entre NameNode e DataNode, piggyback nos heartbeats.

Cliente HDFS

- Aplicações de usuários utilizam HDFS Client como interface ao HDFS.
- Para realizar uma leitura:
  1. Client pergunta ao NameNode quais DataNodes têm cópias.
  2. Client contacta diretamente o DataNode.
- Para escrita:
  1. Client pergunta ao NameNode quais DataNodes devem armazenar o 1º bloco do arquivo.
  2. Client escreve por pipeline nos DataNodes escolhidos.
  3. Gravado o 1º bloco, são requisitados novos DataNodes para armazenar réplicas do próximo blocos.
- É exposto o local de armazenamento dos blocos.
  1. Ganho na leitura por MapReduce.
  2. Aplicação pode dfefinir quantas réplicas deseja inserir (padrão 3).

Operação de escrita

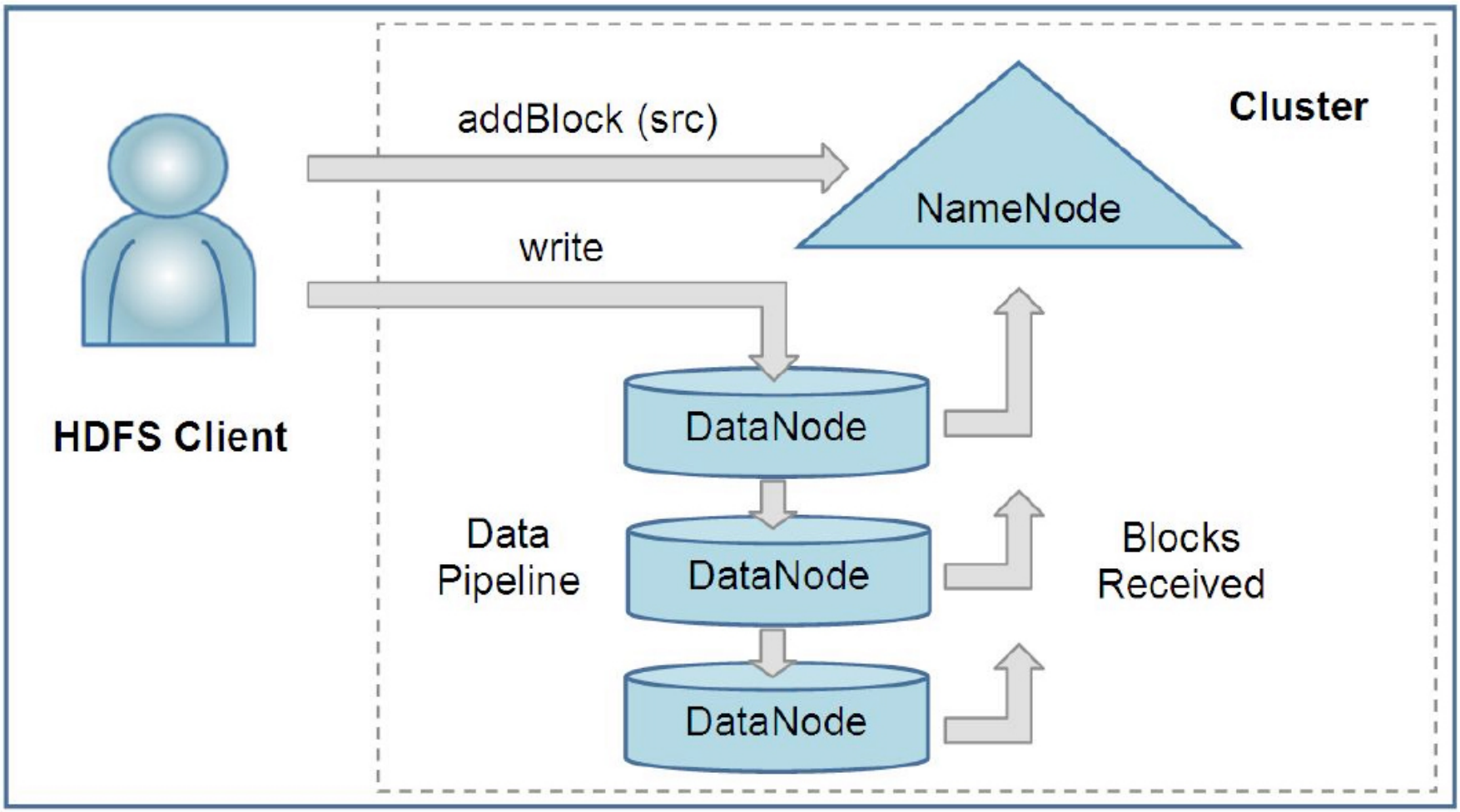


Image e Journal

- Image: metadata do sistema de arquivos. Descreve a organização dos dados de aplicação em diretórios e arquivos.
- Journal: guarda as alterações realizadas no sistema de arquivos.
- Checkpoint: Gravação persistente da imagem.

CheckpointNode e BackupNode

- Papéis extras ao NameNode.
- CheckpointNode: periodicamente salva o checkpoint atual e cria um novo journal.
- BackupNode:
  1. mantém uma Image (sincronizada periodicamente) atualizada em memória.
  2. como um NameNode só de leitura.
  3. agiliza a recuperação em caso de falha do NameNode.

Snapshots

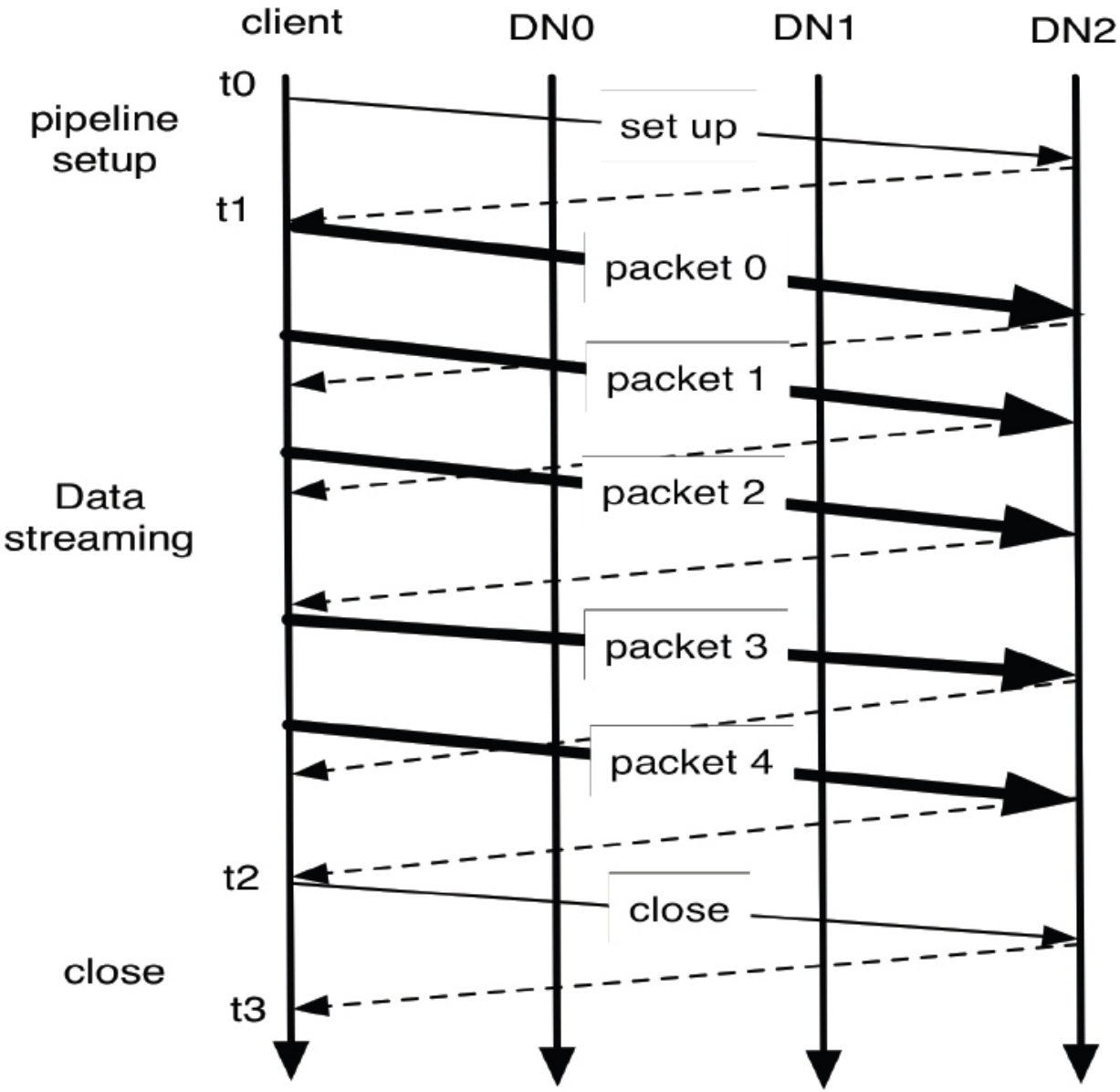
- Atualizações podem gerar falhas (*e.g.* devido a *bugs*).
- Snapshots:
  - Visam minimizar os danos do sistema durante atualizações.
  - Guarda o namespace e o estado de armazenamento antes da atualização.
  - Criado na inicialização do sistema.
- Passos:
  1. Administrador do cluster requisita Snapshot;
  2. NameNode lê checkpoint e journal e faz um merge na memória;
  3. NameNode gera novo checkpoint e um journal vazio;
  4. NameNode indica no handshake que o DataNode deve criar um snapshot local;
  5. DataNode copia apenas a estrutura de diretórios e hard links dos blocos.
- Não existe *roll forward*, apenas roll back.

Operações de I/O e Gerência de Replicação

Operações

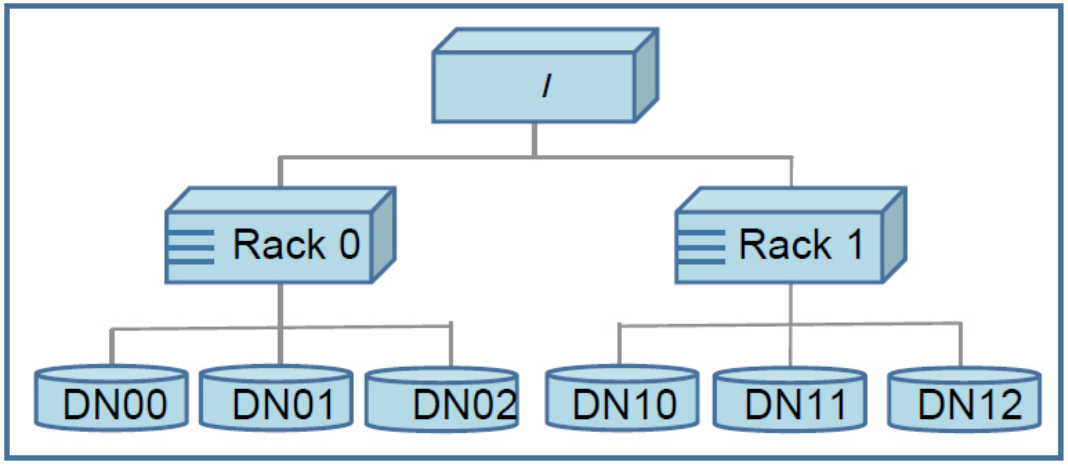
- Leitura e escrita: modelo *single-writer, multiple-reader*.
- Escrita:
  - Coordenado por heartbeats ao NameNode.
  - Arquivo aberto por mais de 1 hora, sem renovação, é fechado.
  - hflush: liberação para visualização.
- Leitura:
  - Mesmo sendo realizada a escrita aberto (réplicas armazenadas).

Operação de escrita



Estrátegia de gravação de blocos

- HDFS visa minimizar o custo de escrita e maximizar a confiabilidade, disponibilidade e banda agregada para acesso a um dado.
- 1º bloco armazenado no DataNode que está escrevendo o dado, 2º e 3º blocos em DataNodes diferentes de racks diferentes e os demais em nós randômicos.
  - Restrições: não gravar mais de mesma réplica em um mesmo nó e não mais de duas réplicas em um rack.



Experimentos

- Ambiente:
  - Yahoo!
  - 3500 nós
  - 2 quad core Xeon processors @ 2.5ghz
  - Red Hat Enterprise Linux Server Release 5.1
  - Sun Java JDK $1.6.0\_13 - b03$
  - 4 directly attached SATA drives (one terabyte each)
  - 16G RAM
  - 1-gigabit Ethernet
  - 70% para o HDFS e o resto para o SO, logs etc.
  - 40 nós por rack ligados a 1 switch que se ligam a um dos 8 core switchs
  - NameNode e BackupNode: 64GB de RAM
  - Total de espaço de armazenamento: 9.8PB (replicado 3x) = 3.3PB livres. 1000 nós = 1PB

- Ilustrando:
  - em um cluster de 3500 nós há 60 milhões de arquivos, que são representados por 63 milhões de blocos.
  - 63.000.000 / 3500 = 18000 * 3 (réplicas) = 54000 blocos em cada nó.
- **Resultados:**
  - Replicar 3x cada bloco é eficiente. Probabilidade de perda de 1 bloco em 1 ano é $< 0,005$.
  - Apenas 0,8% dos nós falham por mês.
  - Re-replicação para um nó recuperado é rápido.
  - Falhas mais graves: perda de rack ou switch. **Mas**, réplicas em racks diferentes auxiliam na recuperação.

Benchmarks

- DFSIO:
  - Ferramenta disponível no Hadoop (MapReduce).
  - Mede a vazão média para operações de leitura, escrita e modificação.
- Tempo de leitura e escrita por DFSIO:
  - Leitura: 66MB/s por nó
  - Escrita: 40MB/s por nó
- Medidas em produção:
  - Executando por algumas semanas.
  - Representa a utilização de milhares de usuários.
  - Mede a vazão média para operações de leitura, escrita e modificação..
- Tempo de leitura e escrita em produção:
  - Leitura: 1.02 MB/s por nó
  - Escrita: 1.09 MB/s por nó

Campeonato de "ordenação". Estressar o sistema movendo dados

Bytes (TB)	Nodes	Maps	Reduces	Time	HDFS I/0 Bytes/s	
					Aggregate (GB)	Per Node (MB)
1	1460	8000	2700	62 s	32	22.1
1000	3658	80 000	20 000	58 500 s	34.2	9.35

Limite superior de operações realizadas pelo NameNode

Operation	Throughput (ops/s)
Open file for read	126 100
Create file	5600
Rename file	8300
Delete file	20 700
DataNode Heartbeat	300 000
Blocks report (blocks/s)	639 700

Table 3. NNThroughput benchmark

Conclusões

- Exposição de uma ferramenta real aplicada à manipulação de Big Data.
- Utilizado por diversas empresas atualmente.
- Único ponto de falha (NameNode) Apesar da existência do BackupNode.
- Utilização de diversas técnicas e também MapReduce.
- Sob licença Apache.

The Hadoop Distributed File System

Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler
Yahoo!
2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)

Apresentado por: Edelberto Franco Silva
Instituto de Computação - Universidade Federal Fluminense (UFF)

Junho, 2012