

# Prévia da Defesa de Mestrado

## Estudo e Implementação de Códigos Corretores de Erros no Sistema de Arquivos Distribuído do Hadoop

Celina d' Ávila Samogin  
Instituto de Computação — UNICAMP

17 de dezembro de 2012

Orientadora: Profa. Dra. Islene Calciolari Garcia

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados
- 5 Referências Bibliográficas

# Agenda

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados
- 5 Referências Bibliográficas

# Motivação

- Este trabalho é uma contribuição para *software* livre em sistemas distribuídos.
- Armazenamento de dados  $\implies$  componente essencial na computação de alto desempenho.
- Códigos Corretores de Erro (*Erasure codes*) introduzem redundância para alcançar confiabilidade e redução do custo de armazenamento.

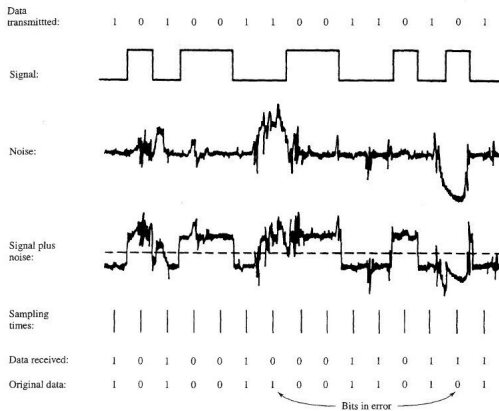


Figure 2.9 Effect of Noise on a Digital Signal

Figura: Efeito do ruído no sinal digital [22]

# Motivação

- Popularização dos computadores e as pesquisas espaciais  $\implies$  os códigos corretores de erros tornaram-se parte comum de comunicações por satélite, de redes de computadores, de armazenamento em discos óticos e outros meios magnéticos.
- Uso frequente em nosso cotidiano

# Motivação



Figura: Assistir programa de televisão Vila Sésamo[24]

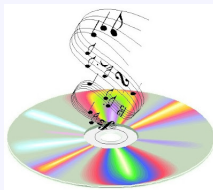


Figura: Ouvir música a partir de um CD[3]

# Motivação



Figura: Atender um telefonema[9]

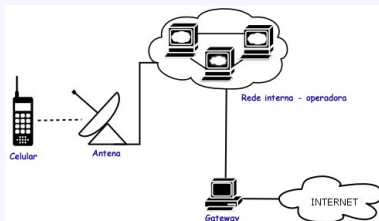


Figura: Navegar pela internet[7]



- Alguns sistemas que utilizam códigos corretores de erros:
  - ▶ *NASA's Deep Space Network* no envio e na recepção de sinais e dados de telemetria (*downlinks*) vindos de veículos espaciais (*very distant spacecrafts*) e para enviar telecomandos (*uplinks*) para veículos espaciais;
  - ▶ *Delay and Disruption Tolerant Networks*, redes de sensores e redes *peer-to-peer*;
  - ▶ armazenamento de grande volume de dados, como também o sistema de arquivos distribuído do Hadoop (HDFS).

# Motivação

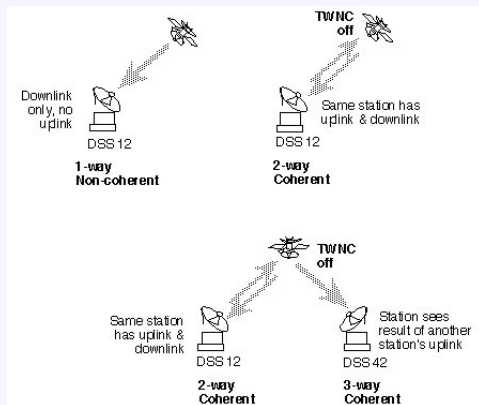


Figura: *NASA's Deep Space Network*[15]

# Motivação

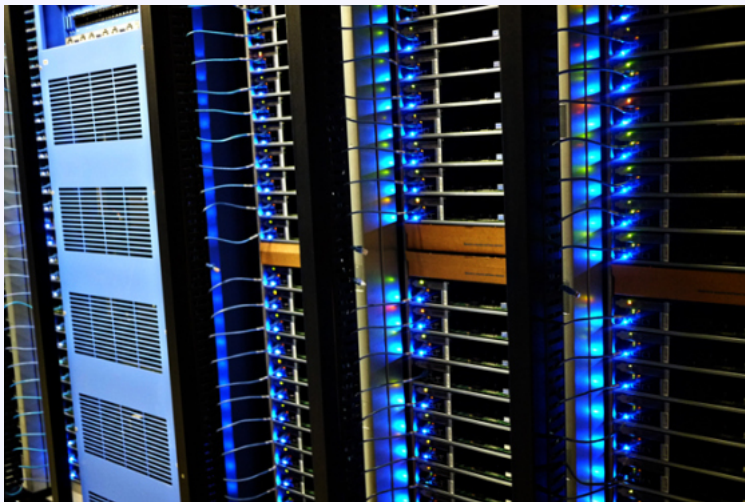


Figura: Facebook data center[25]

# Motivação

- O HDFS, por padrão, implementa alta disponibilidade dos dados via replicação simples dos blocos de dados. Esta abordagem acarreta um alto custo de armazenamento para garantir que os dados estarão sempre disponíveis.
- Esforços iniciais ==> *Redundant Array of Independent Drives* (RAID) [10, 16] e mais recentemente, Reed-Solomon (RS) [12].

# Motivação

- RAID (*Redundant Arrays of Inexpensive [Independent] Disks* (RAID), do ponto de vista de suas estruturas algébricas, é uma classe de códigos de blocos lineares.
- RAID é um método para prover tolerância a falhas ou alto desempenho em sistemas de armazenagem utilizando, para isso, distribuição de dados em diferentes dispositivos e uma codificação de correção de erros ou paridade ou replicação de dados [18].
- RAID foi introduzido por D. A. Patterson na Universidade da California, Berkeley (UC Berkeley) em 1988 [16].
- O código fonte do Hadoop implementa RAID-5, que na sua versão clássica, "fatia" dados e paridade, através dos discos, como num arranjo. A única paridade é calculada através da operação *bitwise* XOR.

# Motivação

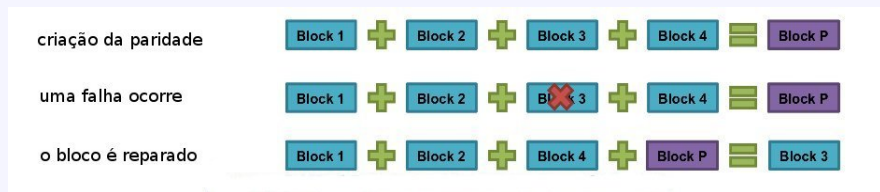


Figura: RAID 5: 1 bloco de paridade e *stripe* = 4 blocos [13]

# Motivação

- Códigos Reed-Solomon (RS) são códigos de bloco, lineares e cíclicos (sobre anéis comutativos sob algumas condições).
- Parametrizáveis  $\implies$  capacidade de correção de erros pode ser alterada facilmente.
- RS são códigos MDS  $\implies$  as palavras-código apresentam máxima distância de Hamming entre si, permitida pelo número de dígitos de verificação de paridade do código.
- Segundo os autores em [2, 21], códigos RS são úteis para correção de erros em rajada.

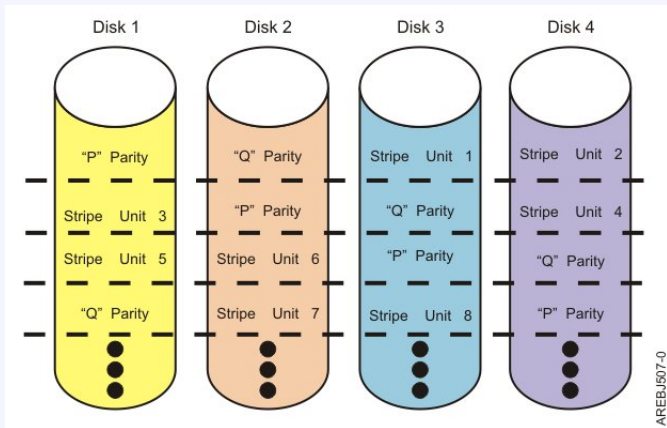
## Definition

**Burst Errors** Uma sequência de erros em rajada ou *burst errors* de tamanho  $b$  é da forma:

$$(0, 0, \dots, 0, 1, \dots, 1, 0, \dots, 0, 0)$$

onde o primeiro bit 1 está na  $i$ -posição e o último bit 1 está na  $(i + b - 1)$ -posição e entre esses primeiro e último bits existe uma

# Motivação



**Figura:** RAID 6 clássico "fatia" dados e paridade, através dos discos, como num arranjo. A paridade P e Q são geradas pela operação *bitwise* XOR e pelo algoritmo Reed-Solomon.[14]



# Objetivos deste trabalho

- avaliação de desempenho, ganhos, e custos de diferentes estratégias de códigos corretores de erro;
- implementação de otimizações ou extensões para o código fonte que atualmente implementa Reed-Solomon, tentando melhorar, principalmente, a parte de distribuição de blocos;
- implementação de novos algoritmos (Tornado e Turbo *codes*) e extensão da interface atual para aceitá-los;
- integração do código fonte atual com o HDFS.

# Agenda

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados
- 5 Referências Bibliográficas

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto
  - ▶ gigabyte -  $10^9$  - um DVD armazena 4.7 GB de um filme

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto
  - ▶ gigabyte -  $10^9$  - um DVD armazena 4.7 GB de um filme
  - ▶ terabyte -  $10^{12}$  - 200 filmes



# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto
  - ▶ gigabyte -  $10^9$  - um DVD armazena 4.7 GB de um filme
  - ▶ terabyte -  $10^{12}$  - 200 filmes
  - ▶ petabyte -  $10^{15}$

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto
  - ▶ gigabyte -  $10^9$  - um DVD armazena 4.7 GB de um filme
  - ▶ terabyte -  $10^{12}$  - 200 filmes
  - ▶ petabyte -  $10^{15}$
  - ▶ exabyte -  $10^{18}$

# Sistemas de Arquivos Distribuídos

- Processar um volume relativamente grande de dados é possível, em poucas horas, com alguns dólares e com algumas máquinas:  
<http://aws.amazon.com/>
- Isto também pode ser feito com o Hadoop, um *framework* para processamento de dados em larga escala.
- Volume grande de dados ?
  - ▶ megabyte -  $10^6$  - uma foto
  - ▶ gigabyte -  $10^9$  - um DVD armazena 4.7 GB de um filme
  - ▶ terabyte -  $10^{12}$  - 200 filmes
  - ▶ petabyte -  $10^{15}$
  - ▶ exabyte -  $10^{18}$
  - ▶ zettabyte -  $10^{21}$  - um disco de 140 GB para cada pessoa no mundo

# Software Open Source



- O Hadoop disponibiliza um armazenamento compartilhado (HDFS/*Hadoop Distributed Filesystem*) e um sistema de análise (MapReduce), formando um *framework* ou arcabouço.

# RDBMS X HDFS/MapReduce

Por que não usamos banco de dados ?

- Em um banco de dados, uma atualização da árvore-B pode gerar muitas operações de *seeking*.

# RDBMS X HDFS/MapReduce

Por que não usamos banco de dados ?

- Em um banco de dados, uma atualização da árvore-B pode gerar muitas operações de *seeking*.
- *seek time* - processo de mover a cabeça do disco para um dado local no disco com o objetivo de ler ou de escrever dados

# RDBMS X HDFS/MapReduce

Por que não usamos banco de dados ?

- Em um banco de dados, uma atualização da árvore-B pode gerar muitas operações de *seeking*.
- *seek time* - processo de mover a cabeça do disco para um dado local no disco com o objetivo de ler ou de escrever dados
- É mais rápido usar um algoritmo de ordenação (*sort/merge*) para:

# RDBMS X HDFS/MapReduce

Por que não usamos banco de dados ?

- Em um banco de dados, uma atualização da árvore-B pode gerar muitas operações de *seeking*.
- *seek time* - processo de mover a cabeça do disco para um dado local no disco com o objetivo de ler ou de escrever dados
- É mais rápido usar um algoritmo de ordenação (*sort/merge*) para:
  - ▶ analisar um grande volume de dados do tamanho de petabytes



# RDBMS X HDFS/MapReduce

Por que não usamos banco de dados ?

- Em um banco de dados, uma atualização da árvore-B pode gerar muitas operações de *seeking*.
- *seek time* - processo de mover a cabeça do disco para um dado local no disco com o objetivo de ler ou de escrever dados
- É mais rápido usar um algoritmo de ordenação (*sort/merge*) para:
  - ▶ analisar um grande volume de dados do tamanho de petabytes
  - ▶ reconstruir toda a base de dados

# RDBMS X HDFS/MapReduce

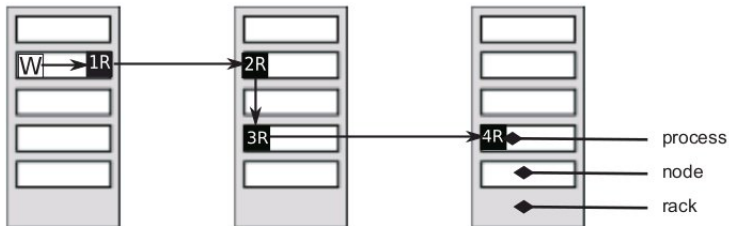
|             | Aplicação para RDBMS            | Aplicação para HDFS/MapReduce    |
|-------------|---------------------------------|----------------------------------|
| tamanho     | gigabytes                       | petabytes                        |
| acesso      | OLTP(interativo) e <i>batch</i> | <i>batch</i>                     |
| atualização | lê e escreve muitas vezes       | escreve uma vez, lê muitas vezes |
| esquema     | estático                        | dinâmico                         |
| integridade | alta                            | baixa                            |
| escala      | não linear                      | linear                           |

Fonte: [27]

# HDFS



HFS disponibiliza espaço para sistema de arquivos e permite que os dados do usuário sejam armazenados em arquivos. Internamente, um arquivo é dividido em um ou mais blocos e esses blocos são armazenados em um conjunto de DataNodes. O tamanho *default* de cada bloco é 64MB.



# Distributed Raid File

- 2 componentes

# Distributed Raid File

- 2 componentes
  - ▶ RaidNode: *daemon* cria os arquivos de paridade para os arquivos armazenados no hdfs e executa o *scan* nos arquivos com mais de 2 blocos e que não foram modificados nas últimas 24 horas; as exceções tratadas são: `ChecksumException` e `BlocMissingException`

# Distributed Raid File

- 2 componentes

- ▶ RaidNode: *daemon* cria os arquivos de paridade para os arquivos armazenados no hdfs e executa o *scan* nos arquivos com mais de 2 blocos e que não foram modificados nas últimas 24 horas; as exceções tratadas são: `ChecksumException` e `BlocMissingException`
- ▶ raidfs: intercepta os pedidos de leitura vindos de uma aplicação ao hdfs *client*

# Distributed Raid File

- 2 componentes
  - ▶ RaidNode: *daemon* cria os arquivos de paridade para os arquivos armazenados no hdfs e executa o *scan* nos arquivos com mais de 2 blocos e que não foram modificados nas últimas 24 horas; as exceções tratadas são: `ChecksumException` e `BlocMissingException`
  - ▶ raidfs: intercepta os pedidos de leitura vindos de uma aplicação ao hdfs *client*
- existe a ferramenta de linha de comando `fsckraid`

# MapReduce



- Os dados que MapReduce processa são dados não estruturados como texto ou imagens. O HDFS, através do *pipeline* dos DataNodes, tenta colocar esses dados no nó onde são feitas as computações, desta forma, o acesso aos dados é rápido, pois é "mais" local.



# MapReduce



- Os dados que MapReduce processa são dados não estruturados como texto ou imagens. O HDFS, através do *pipeline* dos DataNodes, tenta colocar esses dados no nó onde são feitas as computações, desta forma, o acesso aos dados é rápido, pois é "mais" local.
- O MapReduce pode resolver problemas genéricos cujos dados podem ser divididos em matrizes de dados, para cada matriz a mesma computação é necessária e não existe necessidade de comunicação entre as tarefas.

# MapReduce



- Os dados que MapReduce processa são dados não estruturados como texto ou imagens. O HDFS, através do *pipeline* dos DataNodes, tenta colocar esses dados no nó onde são feitas as computações, desta forma, o acesso aos dados é rápido, pois é "mais" local.
- O MapReduce pode resolver problemas genéricos cujos dados podem ser divididos em matrizes de dados, para cada matriz a mesma computação é necessária e não existe necessidade de comunicação entre as tarefas.
- Problemas como empacotamento, linha de fábrica, otimização não são resolvidos pelo modelo de computação do MapReduce.

# MapReduce



Problema genérico:

- iteração sobre um número grande de registros
- *Map* extrai algo de cada registro (chave, valor)
- rearranjo (*shuffle* e ordenação de resultados intermediários por (chave, valor))
- *Reduce* agrega os resultados intermediários
- geração da saída

# Agenda

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados
- 5 Referências Bibliográficas

# Álgebra Abstrata e de Códigos Corretores de Erros

- *Erasure Coding*

# Álgebra Abstrata e de Códigos Corretores de Erros

- *Erasure Coding*
- Exame de Qualificação para o Mestrado (EQM)

# Álgebra Abstrata e de Códigos Corretores de Erros

- *Erasure Coding*
- Exame de Qualificação para o Mestrado (EQM)
- Foco: armazenamento de dados

# Álgebra Abstrata e de Códigos Corretores de Erros

- *Erasure Coding*
- Exame de Qualificação para o Mestrado (EQM)
- Foco: armazenamento de dados
- Códigos corretores de erro disponíveis na versão 0.22 do Hadoop: RAID e RS





# Álgebra Abstrata e de Códigos Corretores de Erros

- *Erasure Coding*
- Exame de Qualificação para o Mestrado (EQM)
- Foco: armazenamento de dados
- Códigos corretores de erro disponíveis na versão 0.22 do Hadoop: RAID e RS



- Muitos artigos e material de aulas das disciplinas de universidades de Portugal, Índia, Paquistão e EUA

# Intuições na Álgebra Abstrata

- Congruência linear

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana
- Independência linear

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana
- Independência linear
- Espaço Vetorial, Grupo, Anel e Corpo (principalmente  $\mathbb{GF}_2$  e  $\mathbb{GF}_{256}$ )

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana
- Independência linear
- Espaço Vetorial, Grupo, Anel e Corpo (principalmente  $\mathbb{GF}_2$  e  $\mathbb{GF}_{256}$ )
- Sistema de Equações Lineares

# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana
- Independência linear
- Espaço Vetorial, Grupo, Anel e Corpo (principalmente  $\mathbb{GF}_2$  e  $\mathbb{GF}_{256}$ )
- Sistema de Equações Lineares
- Matrizes



# Intuições na Álgebra Abstrata

- Congruência linear
- Classes Residuais
- Resto da divisão euclidiana
- Independência linear
- Espaço Vetorial, Grupo, Anel e Corpo (principalmente  $\mathbb{GF}_2$  e  $\mathbb{GF}_{256}$ )
- Sistema de Equações Lineares
- Matrizes
- Polinômios

# Intuições na Álgebra Abstrata

## Definition

**Congruência Linear** Seja  $q \in \mathbb{Z}$ . Dois inteiros  $a$  e  $b$  dizem-se congruentes módulo  $q$  se, e somente se,  $q$  divide  $a - b$ . A notação é  $a \equiv b \pmod{q}$ . Assim,  $a - b \in \{x \text{ tal que } x = kq, k \in \mathbb{Z}\}$ .

## Example

$$32 \equiv 2 \pmod{3}$$

$$27 \equiv 5 \pmod{11}$$

$$63 \equiv 7 \pmod{8}$$

# Intuições na Álgebra Abstrata

## Definition

**Conjunto das Classes Residuais** Seja  $\mathbb{Z}/q = \{\bar{0}_q, \bar{1}_q, \dots, \overline{q-1}_q\}$  o conjunto das classes residuais dos inteiros módulo  $q$ .

Note que  $\mathbb{Z}/q$  tem exatamente  $q$  elementos, portanto  $\mathbb{Z}/q$  é finito. As classes residuais possuem propriedades:

- (i) Se  $a \equiv b \pmod{q}$ , então  $\bar{a}_q = \bar{b}_q$ .
- (ii) Se  $\bar{a}_q \cap \bar{b}_q \neq 0$ , então  $\bar{a}_q = \bar{b}_q$ .
- (iii) Para  $a \in \mathbb{Z}$ ,  $\bigcup \bar{a}_q = \mathbb{Z}$ .

# Conceitos em Códigos Corretores de Erros

- Teoria da codificação  $\implies$  propriedades dos códigos e suas aplicações

# Conceitos em Códigos Corretores de Erros

- Teoria da codificação  $\implies$  propriedades dos códigos e suas aplicações
- Teoria da Informação, Claude Shannon [20]  $\implies$  incerteza da informação e capacidade do canal

# Conceitos em Códigos Corretores de Erros

- Teoria da codificação  $\implies$  propriedades dos códigos e suas aplicações
- Teoria da Informação, Claude Shannon [20]  $\implies$  incerteza da informação e capacidade do canal
- C. Shannon estudou o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação  $\implies$  hoje, é área da engenharia elétrica e da matemática aplicada

# Conceitos em Códigos Corretores de Erros

- Teoria da codificação  $\implies$  propriedades dos códigos e suas aplicações
- Teoria da Informação, Claude Shannon [20]  $\implies$  incerteza da informação e capacidade do canal
- C. Shannon estudou o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação  $\implies$  hoje, é área da engenharia elétrica e da matemática aplicada
- Sinais podem ser som, áudio, dados biológicos como eletrocardiogramas ou sequências de DNA [6, 8], sinais de sistemas de telecomunicações, entre muitos outros.

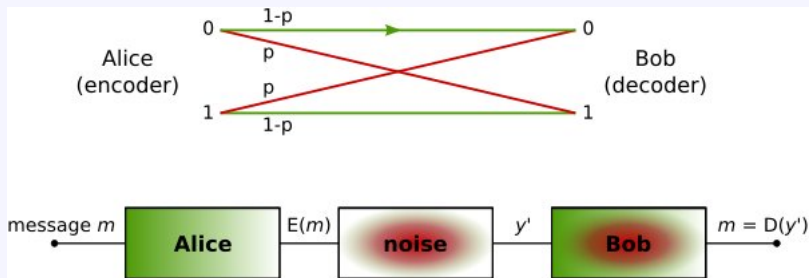
# Conceitos em Códigos Corretores de Erros

- Teoria da codificação  $\implies$  propriedades dos códigos e suas aplicações
- Teoria da Informação, Claude Shannon [20]  $\implies$  incerteza da informação e capacidade do canal
- C. Shannon estudou o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação  $\implies$  hoje, é área da engenharia elétrica e da matemática aplicada
- Sinais podem ser som, áudio, dados biológicos como eletrocardiogramas ou sequências de DNA [6, 8], sinais de sistemas de telecomunicações, entre muitos outros.
- Objetivo da codificação de canal é aumentar a resistência do sistema de comunicações digital face aos efeitos do ruído de canal.



# Conceitos em Códigos Corretores de Erros

- Canal binário simétrico



# Conceitos em Códigos Corretores de Erros

- A ciclicidade de códigos sob anéis comutativos está associada à multiplicação do polinômio gerador pelas diferentes potências de "x" reduzida módulo um polinômio irredutível.

# Conceitos em Códigos Corretores de Erros

- Detectar erros e codificar dados tem a mesma complexidade

# Conceitos em Códigos Corretores de Erros

- Detectar erros e codificar dados tem a mesma complexidade
- Decodificar é np-completo

# Conceitos em Códigos Corretores de Erros

- Detectar erros e codificar dados tem a mesma complexidade
- Decodificar é np-completo
- Estruturas algébricas: matrizes

# Conhecer e Compilar o código fonte disponível



- Facebook, Yahoo, Fundação Apache

# Conhecer e Compilar o código fonte disponível



- Facebook, Yahoo, Fundação Apache
- O *kernel* do Hadoop é constituído do core, hdfs e mapred.

# Conhecer e Compilar o código fonte disponível



- Facebook, Yahoo, Fundação Apache
- O *kernel* do Hadoop é constituído do core, hdfs e mapred.
- As versões oficiais para instalação  
(<http://www.apache.org/dist/hadoop/core/>) e



# Conhecer e Compilar o código fonte disponível



- Facebook, Yahoo, Fundação Apache
- O *kernel* do Hadoop é constituído do core, hdfs e mapred.
- As versões oficiais para instalação (<http://www.apache.org/dist/hadoop/core/>) e
- as versões do código fonte (common, hdfs, mapreduce) estão disponíveis em repositório SVN (<https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.22>)

# Conhecer e Compilar o código fonte disponível



- Facebook, Yahoo, Fundação Apache
- O *kernel* do Hadoop é constituído do core, hdfs e mapred.
- As versões oficiais para instalação (<http://www.apache.org/dist/hadoop/core/>) e
- as versões do código fonte (common, hdfs, mapreduce) estão disponíveis em repositório SVN (<https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.22>)
- Na atividade *Scripts for building Hadoop 0.22.0 release* <https://issues.apache.org/jira/browse/HADOOP-6846>, os mantenedores e contribuidores do Hadoop discutiam como compilar a versão 0.22

# Conhecer e Compilar o código fonte disponível

- Eclipse IDE for Java

# Conhecer e Compilar o código fonte disponível

- Eclipse IDE for Java
- comandos do linux, principalmente, grep, find e editores de texto

# Conhecer e Compilar o código fonte disponível

- Eclipse IDE for Java
- comandos do linux, principalmente, grep, find e editores de texto
- Doxygen para gerar diagramas de herança e de colaboração das classes a partir do código fonte

# Conhecer e Compilar o código fonte disponível

- Eclipse IDE for Java
- comandos do linux, principalmente, grep, find e editores de texto
- Doxygen para gerar diagramas de herança e de colaboração das classes a partir do código fonte
- Implementar uma aplicação do *framework* do Hadoop: conta-palavras, temperatura máxima, *page rank*

# Conhecer e Compilar o código fonte disponível

- Eclipse IDE for Java
- comandos do linux, principalmente, grep, find e editores de texto
- Doxygen para gerar diagramas de herança e de colaboração das classes a partir do código fonte
- Implementar uma aplicação do *framework* do Hadoop: conta-palavras, temperatura máxima, *page rank*
- Fazer uma pequena alteração no código fonte do RaidNode () para escrever uma mensagem no arquivo de log, compilar o código fonte modificado, instalá-lo e testá-lo

# Conhecer e Compilar o código fonte disponível

- Hadoop foi escrito quase que inteiramente em Java: construtores de classe, asserções



# Conhecer e Compilar o código fonte disponível

- Hadoop foi escrito quase que inteiramente em Java: construtores de classe, asserções
- Java: 1o capítulo do livro vermelho do Sedgewick, Algorithms

# Conhecer e Compilar o código fonte disponível

- Hadoop foi escrito quase que inteiramente em Java: construtores de classe, asserções
- Java: 1o capítulo do livro vermelho do Sedgewick, Algorithms
- Entender a camada RAID para poder extendê-la

# Conhecer e Compilar o código fonte disponível

- Hadoop foi escrito quase que inteiramente em Java: construtores de classe, asserções
- Java: 1o capítulo do livro vermelho do Sedgewick, Algorithms
- Entender a camada RAID para poder extendê-la
- Entender de álgebra abstrata e de uma de suas aplicações: códigos corretores de erros

# Conhecer e Compilar o código fonte disponível

- Hadoop foi escrito quase que inteiramente em Java: construtores de classe, asserções
- Java: 1o capítulo do livro vermelho do Sedgewick, Algorithms
- Entender a camada RAID para poder extendê-la
- Entender de álgebra abstrata e de uma de suas aplicações: códigos corretores de erros
- Propor os algoritmos dos novos *codecs*: *encode* e *decode*

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu



# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu
  - ▶ Laboratório da pós-graduação do IC; estação 64 bits com Fedora

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu
  - ▶ Laboratório da pós-graduação do IC; estação 64 bits com Fedora
  - ▶ Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu
  - ▶ Laboratório da pós-graduação do IC; estação 64 bits com Fedora
  - ▶ Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- Ambiente de instalação do *tarball*

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu
  - ▶ Laboratório da pós-graduação do IC; estação 64 bits com Fedora
  - ▶ Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- Ambiente de instalação do *tarball*
  - ▶ Notebook com Ubuntu

# Conhecer e Compilar o código fonte disponível

- Ambiente de edição do código fonte das classes
  - ▶ Notebook com Ubuntu 8.04, depois 10.04, depois 12.04 (2GB de RAM, *dual-core*, 150MB de disco interno, 1.35TB de disco externo)
- Ambiente de compilação do Hadoop versão 0.22
  - ▶ Notebook com Ubuntu
  - ▶ Laboratório da pós-graduação do IC; estação 64 bits com Fedora
  - ▶ Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- Ambiente de instalação do *tarball*
  - ▶ Notebook com Ubuntu
  - ▶ Instâncias da Amazon EC2

# Testes Iniciais

- Notebook com Ubuntu

# Testes Iniciais

- Notebook com Ubuntu
- Testes funcionais: *encode* e *decode* das novas codificações e correção de erros no código fonte das classes das codificações implementadas por esse trabalho ==> linguagem Java.

# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)



# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- AWS Console e Amazon EC2 API tools (ferramentas de linha de comando)

# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- AWS Console e Amazon EC2 API tools (ferramentas de linha de comando)
- 3 fases de Testes de Desempenho (coleta de dados) da operação de *encode* e de Injeção de Falhas das operações de *encode* e *decode* das novas codificações

# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- AWS Console e Amazon EC2 API tools (ferramentas de linha de comando)
- 3 fases de Testes de Desempenho (coleta de dados) da operação de *encode* e de Injeção de Falhas das operações de *encode* e *decode* das novas codificações
- Primeira fase: preparação do ambiente de teste

# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- AWS Console e Amazon EC2 API tools (ferramentas de linha de comando)
- 3 fases de Testes de Desempenho (coleta de dados) da operação de *encode* e de Injeção de Falhas das operações de *encode* e *decode* das novas codificações
- Primeira fase: preparação do ambiente de teste
- Rede de 4 instâncias: 1 *master/slave* e 3 *slaves*

# Testes Finais

- Instâncias da Amazon EC2 (servidoras virtuais na Nuvem) com Ubuntu 12.10, *kernel*, (613MB de RAM, *dual-core*, alguns *gigabytes* de disco)
- AWS Console e Amazon EC2 API tools (ferramentas de linha de comando)
- 3 fases de Testes de Desempenho (coleta de dados) da operação de *encode* e de Injeção de Falhas das operações de *encode* e *decode* das novas codificações
- Primeira fase: preparação do ambiente de teste
- Rede de 4 instâncias: 1 *master/slave* e 3 *slaves*
- Base de dados: imagens ISO (+ – 800MB cada uma)

# Testes Finais

- Segunda fase: coleta de dados para medição de desempenho da operações de *encode* e de *decode* para cada codificação:
  - ▶ replicação simples 2X
  - ▶ replicação simples 3X
  - ▶ *XOR*(6, 5)
  - ▶ *RS*(8, 5)
  - ▶ *Tornado*(10, 5)
  - ▶ *Tornado*(20, 10)
  - ▶ *Turbo — like*(10, 5)
  - ▶ *Turbo — like*(20, 10)

# Testes Finais

- Segunda fase: coleta de dados para medição de desempenho da operações de *encode* e de *decode* para cada codificação:
  - ▶ replicação simples  $2X$
  - ▶ replicação simples  $3X$
  - ▶  $XOR(6, 5)$
  - ▶  $RS(8, 5)$
  - ▶  $Tornado(10, 5)$
  - ▶  $Tornado(20, 10)$
  - ▶  $Turbo - like(10, 5)$
  - ▶  $Turbo - like(20, 10)$
- Rede de 16 instâncias *large*, provavelmente
- Base de dados: imagens ISO

# Testes Finais

- Terceira fase: coleta de dados para medição da injeção de falhas
  - falha em 1 réplica** : corromper 1 réplica de blocos de dados de um arquivo
  - falha em  $n$  réplicas** corromper  $n$  réplicas de blocos de dados de um arquivo
  - falha em réplicas e em blocos de dados** corromper  $n$  réplicas de blocos de dados e os  $k - 1$  blocos de dados de um arquivo



# Testes Finais

- Terceira fase: coleta de dados para medição da injeção de falhas
  - falha em 1 réplica** : corromper 1 réplica de blocos de dados de um arquivo
  - falha em  $n$  réplicas** corromper  $n$  réplicas de blocos de dados de um arquivo
  - falha em réplicas e em blocos de dados** corromper  $n$  réplicas de blocos de dados e os  $k - 1$  blocos de dados de um arquivo
- Rede de 16 instâncias *large*, provavelmente

# Testes Finais

- Terceira fase: coleta de dados para medição da injeção de falhas
  - falha em 1 réplica** : corromper 1 réplica de blocos de dados de um arquivo
  - falha em  $n$  réplicas** corromper  $n$  réplicas de blocos de dados de um arquivo
  - falha em réplicas e em blocos de dados** corromper  $n$  réplicas de blocos de dados e os  $k - 1$  blocos de dados de um arquivo
- Rede de 16 instâncias *large*, provavelmente
- Base de dados: imagens ISO

# Testes Finais

| Métricas por Codificação                                 | Origem                          |
|--|---------------------------------|
| Tempo total de execução do experimento                   | sistema operacional             |
| Tamanho do arquivo carregado no hdfs                     | interface web do hdfs           |
| Tempo de carga no hdfs por arquivo                       | arquivo de log do RaidNode      |
| Tempo de execução/encode por arquivo codificado          | arquivo de log do RaidNode      |
| Tempo de execução/decode por arquivo decodificado        | arquivo de log do RaidNode      |
| Número de iterações na execução do decode                | arquivo de log do RaidNode      |
| Tamanho do bloco do hdfs                                 | arquivo de configuração do hdfs |
| Número de blocos da <i>stripe</i>                        | arquivo de configuração do hdfs |
| Fator de redundância previsto por arquivo                | texto desta dissertação         |
| Fator de redundância por arquivo obtido nos experimentos | interface web do hdfs           |
| Número de blocos com falhas suportado                    | arquivo de configuração do hdfs |
| Número de blocos com falhas obtido nos experimentos      | arquivo de log do RaidNode      |

**Tabela:** Métricas quantitativas coletadas por codificação nos experimentos para operações de *encode* e de *decode*

# Testes Finais

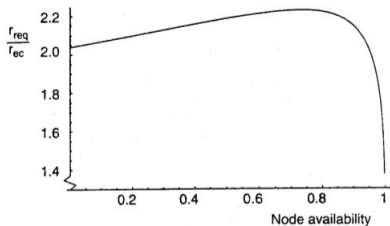


Figure 11-27. The ratio  $r_{rep}/r_{ec}$  as a function of node availability  $\alpha$ .

**Figura:** Razão sobrecarga da replicação/sobrecarga da codificação X disponibilidade dos nós [23]

# Agenda

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados**
- 5 Referências Bibliográficas

# Resultados

- Esse trabalho estendeu e alterou código fonte distribuído sob a licença Apache.

---

<sup>1</sup><https://github.com/celinasam/>

# Resultados

- Esse trabalho estendeu e alterou código fonte distribuído sob a licença Apache.
- Com as codificações RAID, RS, já disponíveis no HDFS e as codificações Tornado e Turbo-like, implementadas por esse trabalho de mestrado, o HDFS disponibiliza, com uma flexível configuração, as principais codificações para canal binário simétrico

---

<sup>1</sup><https://github.com/celinasam/>

# Resultados

- Esse trabalho estendeu e alterou código fonte distribuído sob a licença Apache.
- Com as codificações RAID, RS, já disponíveis no HDFS e as codificações Tornado e Turbo-like, implementadas por esse trabalho de mestrado, o HDFS disponibiliza, com uma flexível configuração, as principais codificações para canal binário simétrico
- Os diagramas de herança e de colaboração para as classes e o código fonte dos procedimentos de compilação e de geração do arquivo *tarball* foram disponibilizados em repositório público <sup>1</sup>.

---

<sup>1</sup><https://github.com/celinasam/>



# Resultados

- Esse trabalho estendeu e alterou código fonte distribuído sob a licença Apache.
- Com as codificações RAID, RS, já disponíveis no HDFS e as codificações Tornado e Turbo-like, implementadas por esse trabalho de mestrado, o HDFS disponibiliza, com uma flexível configuração, as principais codificações para canal binário simétrico
- Os diagramas de herança e de colaboração para as classes e o código fonte dos procedimentos de compilação e de geração do arquivo *tarball* foram disponibilizados em repositório público <sup>1</sup>.
- O código fonte da implementação das codificações será disponibilizado em breve

---

<sup>1</sup><https://github.com/celinasam/>

# Resultados

- Foram criados um site <sup>2</sup> com comentários dos resultados obtidos desse trabalho e um *blog* <sup>3</sup> para comentar assuntos que a autora considerou de grande interesse para outras pessoas.

---

<sup>2</sup><https://sites.google.com/site/newerasurecodinginhadoop>

<sup>3</sup><http://mcss.posterous.com/>

# Resultados

- Foram criados um site <sup>2</sup> com comentários dos resultados obtidos desse trabalho e um *blog* <sup>3</sup> para comentar assuntos que a autora considerou de grande interesse para outras pessoas.
- Um dos cenários de uso de codificações baseadas em grafos seria em ambientes de *data center*, onde um grande número de máquinas poderiam falhar. Com blocos de paridade íntegros dos arquivos com, pelo menos, 1 dos blocos de dados dos mesmos arquivos disponível, há uma grande probabilidade desse tipo de codificação recuperar os blocos de dados corrompidos. Essa é uma aplicação das codificações implementadas por esse trabalho.

---

<sup>2</sup><https://sites.google.com/site/newerasurecodinginhadoop>

<sup>3</sup><http://mcss.posterous.com/>

# Trabalhos Futuros

- Esse trabalho discutiu esquemas adequados de redundância de dados apenas sobre o aspecto do esquema de dados, sem comentar algoritmos de atualização das réplicas.

# Trabalhos Futuros

- Esse trabalho discutiu esquemas adequados de redundância de dados apenas sobre o aspecto do esquema de dados, sem comentar algoritmos de atualização das réplicas.
- Pode-se concluir com uma aplicação do teorema da Codificação do Canal [1, 19], que um sistema com largura de banda a maior possível tem uma capacidade de canal finita. Em esboços preliminares, comparando-se, para uma dada codificação, pode-se concluir que a probabilidade de erro em uma palavra código sem codificação é maior que a probabilidade de erro em uma palavra código com codificação.

# Trabalhos Futuros

- Esse trabalho discutiu esquemas adequados de redundância de dados apenas sobre o aspecto do esquema de dados, sem comentar algoritmos de atualização das réplicas.
- Pode-se concluir com uma aplicação do teorema da Codificação do Canal [1, 19], que um sistema com largura de banda a maior possível tem uma capacidade de canal finita. Em esboços preliminares, comparando-se, para uma dada codificação, pode-se concluir que a probabilidade de erro em uma palavra código sem codificação é maior que a probabilidade de erro em uma palavra código com codificação.
- O estudo da independência das réplicas de um esquema de redundância é um campo promissor para pesquisas. A avaliação de esquemas de redundância é muitas vezes baseada na suposição de que as réplicas falham de forma independente. Na prática, as falhas não são tão independentes, segundo [4, 26].

# Trabalhos Futuros

- Uma extensão do algoritmo da camada RAID pode melhorar o desempenho da tolerância à falhas.

# Trabalhos Futuros

- Uma extensão do algoritmo da camada RAID pode melhorar o desempenho da tolerância à falhas.
- Aplicações de códigos corretores de erros em outros canais binários simétricos e suas extensões como *joint source-channel coding* parecem uma pesquisa promissora, pois "contenar" codificação de fonte e de canal, tem sido proposto para implementar aplicações que tem requisitos de tempo-real como transmissão de áudio e imagem em um canal binário simétrico com ruído [17].



# Trabalhos Futuros

- Uma extensão do algoritmo da camada RAID pode melhorar o desempenho da tolerância à falhas.
- Aplicações de códigos corretores de erros em outros canais binários simétricos e suas extensões como *joint source-channel coding* parecem uma pesquisa promissora, pois "contenar" codificação de fonte e de canal, tem sido proposto para implementar aplicações que tem requisitos de tempo-real como transmissão de áudio e imagem em um canal binário simétrico com ruído [17].
- O estudo e a implementação de outros mecanismos e técnicas para redução de armazenamento e disponibilidade de dados, otimizando o desempenho de codificações baseadas em grafos, como códigos LDPC

# Trabalhos Futuros

- Uma extensão do algoritmo da camada RAID pode melhorar o desempenho da tolerância à falhas.
- Aplicações de códigos corretores de erros em outros canais binários simétricos e suas extensões como *joint source-channel coding* parecem uma pesquisa promissora, pois "contenar" codificação de fonte e de canal, tem sido proposto para implementar aplicações que tem requisitos de tempo-real como transmissão de áudio e imagem em um canal binário simétrico com ruído [17].
- O estudo e a implementação de outros mecanismos e técnicas para redução de armazenamento e disponibilidade de dados, otimizando o desempenho de codificações baseadas em grafos, como códigos LDPC
- O estudo e implementação de aplicações de códigos corretores de erros, tanto a com memória como a sem memória, para os canais do DNA e do RNA [5, 8] é uma pesquisa bastante desafiadora e promissora.

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop
- Analisadas um pouco mais de 120 discussões do jira

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop
- Analisadas um pouco mais de 120 discussões do jira
- Buscas no jira [issues.apache.org/](https://issues.apache.org/), selecionando projeto "Hadoop Map/Reduce" e componente "contrib/raid", mostram algumas discussões sobre as camadas de codificação por apagamento

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop
- Analisadas um pouco mais de 120 discussões do jira
- Buscas no jira [issues.apache.org/](https://issues.apache.org/), selecionando projeto "Hadoop Map/Reduce" e componente "contrib/raid", mostram algumas discussões sobre as camadas de codificação por apagamento
- Camada RAID: versão 0.21.0 (discussão HDFS-503)

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop
- Analisadas um pouco mais de 120 discussões do jira
- Buscas no jira [issues.apache.org/](https://issues.apache.org/), selecionando projeto "Hadoop Map/Reduce" e componente "contrib/raid", mostram algumas discussões sobre as camadas de codificação por apagamento
- Camada RAID: versão 0.21.0 (discussão HDFS-503)
- Codificação RS: versão 0.22.0 (discussões MAPREDUCE-1969 e MAPREDUCE-1970)

# Interação com a comunidade

- As versões do hadoop relacionadas a esta proposta: 0.20.1, 0.21.0 e 0.22.0 do Hadoop
- Analisadas um pouco mais de 120 discussões do jira
- Buscas no jira [issues.apache.org/](https://issues.apache.org/), selecionando projeto "Hadoop Map/Reduce" e componente "contrib/raid", mostram algumas discussões sobre as camadas de codificação por apagamento
- Camada RAID: versão 0.21.0 (discussão HDFS-503)
- Codificação RS: versão 0.22.0 (discussões MAPREDUCE-1969 e MAPREDUCE-1970)
- Codificação Tornado e Turbo-*like*: possivelmente em versões futuras



# Interação com a comunidade

- Existe um grupo de contribuidores (de várias empresas como Cloudera, Facebook, Yahoo e de universidades como *University of Waterloo* e *Carnegie Mellow University*) da camada RAID, dos quais, destacamos Rodrigo Schmidt, ex-aluno do programa de pós-graduação do Instituto de Computação da Unicamp, que sugeriu o tema deste trabalho e que tem contribuído com várias idéias para a realização deste trabalho.
- Foi possível uma saudável interação e colaboração com os desenvolvedores.

# Agradecimentos

- CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico)
- FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo)
- IC (Instituto de Computação da Unicamp)
- Embrapa (Empresa Brasileira de Pesquisa Agropecuária)

# Agenda

- 1 Introdução
- 2 Contexto em Sistemas Distribuídos e em Software Livre
- 3 Métodos utilizados nesse Estudo e nessa Implementação
- 4 Resultados
- 5 Referências Bibliográficas

# Referências Bibliográficas I



Silvio A. Abrantes.

*Códigos Corretores de Erros em Comunicações Digitais*, chapter 1,2,3,4,5,6,7,8,9, pages 17–600.  
FEUP edições, Porto, Portugal, 2001.



G. M. Almeida.

Códigos corretores de erros em hardware para sistemas de telecomando e telemetria em aplicações espaciais.  
Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul - Faculdade de Informática, Porto Alegre, Brasil, março 2007.



Mari angela.

Notas da alma.

URL=<http://vidaanascer.blogspot.com.br/2012/09/notas-da-alma.html>. Acessado em 10 de dezembro de 2012.

# Referências Bibliográficas II



M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T.J. Giuli, and P. Bungle.

A fresh look at the reliability of long-term digital storage.

In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*, EuroSys '06, pages 221–234, New York, NY, USA, 2006. ACM.



Andréa Santos Leite da Rocha.

*Modelo de Sistema de Comunicações Digital para o Mecanismo de Importação de Proteínas Mitocondriais Através de Códigos Corretores de Erros.*

doctoral dissertation, Departamento de Telemática, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, 2010.

# Referências Bibliográficas III



Luzinete Cristina Bonani de Faria.

*Existencias de Códigos Corretores de Erros e Protocolos de Comunicação em Sequências de DNA.*

doctoral dissertation, Departamento de Telemática, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, 2011.



Fábio Berbert de Paula.

Criando sites para celular com wml (parte 1).

URL=[http://phpbrasil.com/artigo/ozk4Kml6\\_tdC/2/criando-sites-para-celular-com-wml-parte-1](http://phpbrasil.com/artigo/ozk4Kml6_tdC/2/criando-sites-para-celular-com-wml-parte-1). Acessado em 10 de dezembro de 2012.

# Referências Bibliográficas IV



Luzinete C. B. Faria, Andréa S. L. Rocha, João H. Kleinschmidt, Márcio C. Silva-Filho, Edson Bim, Roberto H. Herai, Michel E. B. Yamagishi, and Reginaldo Palazzo Jr.

Is a genome a codeword of an error-correcting code?

*PLoS ONE*, 7(5):e36644, 05 2012.



Volney Faustini.

O telefone tocou na madrugada.

URL=[http://volneyf.blogspot.com.br/2008\\_04\\_01\\_archive.html](http://volneyf.blogspot.com.br/2008_04_01_archive.html).

Acessado em 10 de dezembro de 2012.



Apache Software Foundation.

Hdfs-503 - implement erasure coding as a layer on hdfs.

URL=<https://issues.apache.org/jira/browse/HDFS-503>. Acessado em 08 de maio de 2010.

# Referências Bibliográficas V



Apache Software Foundation.

Hdfs architecture.

URL=[http://hadoop.apache.org/common/docs/current/hdfs\\_design.htm](http://hadoop.apache.org/common/docs/current/hdfs_design.htm)

Acessado em 08 de maio de 2010.



Apache Software Foundation.

Mapreduce-1969 - allow raid to use reed-solomon erasure codes.

URL=<https://issues.apache.org/jira/browse/MAPREDUCE-1969>.

Acessado em 20 de agosto de 2010.



Apache Software Foundation.

Mapreduce-2036 - enable erasure code in tool similar to hadoop archive.

URL=<https://issues.apache.org/jira/browse/MAPREDUCE-2036>.

Acessado em 10 de novembro de 2010.



# Referências Bibliográficas VI



IBM.

Raid 6.

URL=<http://pic.dhe.ibm.com/infocenter/powersys/v3r1m5/index.jsp?topnav=infocenter%2Fpowersys%2Fv3r1m5%2Findex.jsp&lang=en>  
Acessado em 10 de dezembro de 2012.



California Institute of Technology Jet Propulsion Laboratory.

Telecommunications.

URL=<http://www2.jpl.nasa.gov/basics/bsf10-1.php>. Acessado em 10 de dezembro de 2012.



D. A. Patterson, G. Gibson, and R. H. Katz.

A case for redundant arrays of inexpensive disks (raid).

In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 109–116, New York, NY, USA, 1988. ACM.

# Referências Bibliográficas VII



Wikipedia project.

Joint source and channel coding.

URL=[http://en.wikipedia.org/wiki/Joint\\_source\\_and\\_channel\\_coding](http://en.wikipedia.org/wiki/Joint_source_and_channel_coding).

Acessado em 7 de junho de 2012.



Raghu Ramakrishnan and Johannes Gehrke.

*Database Management Systems*, chapter 9, pages 304–337.

McGraw-Hill, New York, NY, USA, 2003.



Mischa Schwartz.

*Information transmission, modulation and noise*, chapter 3, 7, pages 95–200, 564–730.

McGraw-Hill, Singapore, 1990.



C. E. Shannon.

A mathematical theory of communication.

*The Bell System Technical Journal*, 27(3):379–423, 623–656, 1948.

# Referências Bibliográficas VIII



F. MacWilliams J. N. J. A. Sloane.

*The Theory of Error-Correcting Codes*, chapter 3, 4, 7, 10, pages 80–124.

North-Holland Publishing Company, Amsterdam, Netherlands, 1977.



William Stallings.

*Wireless Communications and Networks*, chapter 2, 8, pages 14–45, 193–235.

Prentice Hall, Upper Saddle River, NJ, USA, 2005.



Andrew S. Tanenbaum and Maarten Van Steen.

*Distributed systems: principles and paradigms*, chapter 8,11, pages 321–375,491–544.

Prentice Hall, Upper Saddle River, New Jersey, USA, 2006.

# Referências Bibliográficas IX



## World Tv.

Programas infantis que fizeram história.

URL=<http://worldtelevisao.blogspot.com.br/2011/05/programas-infantis-que-fizeram-historia.html>. Acessado em 10 de dezembro de 2012.



## VentureBeat.

How facebook kept its servers cool in the southern summer heat.

URL=<http://venturebeat.com/2012/11/14/facebook-north-carolina-data-center/>. Acessado em 10 de dezembro de 2012.



## H. Weatherspoon, T. Moscovitz, and J. Kubiawicz.

Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems.

*21st IEEE Symposium on Reliable Distributed Systems*, 0:362–367, 2002.

# Referências Bibliográficas X



T. White.

*Hadoop: The Definitive Guide*, chapter 1, 2, 3, pages 1–74.

O'Reilly, Sebastopol, CA, USA, 2009.