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

9 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 arquivos é um componente essencial na computação de alto desempenho.
- Códigos Corretores de Erro (*Erasure codes*) introduzem redundância e tem sido utilizados em sistemas para alcançar confiabilidade e redução do custo de armazenamento.

Motivação

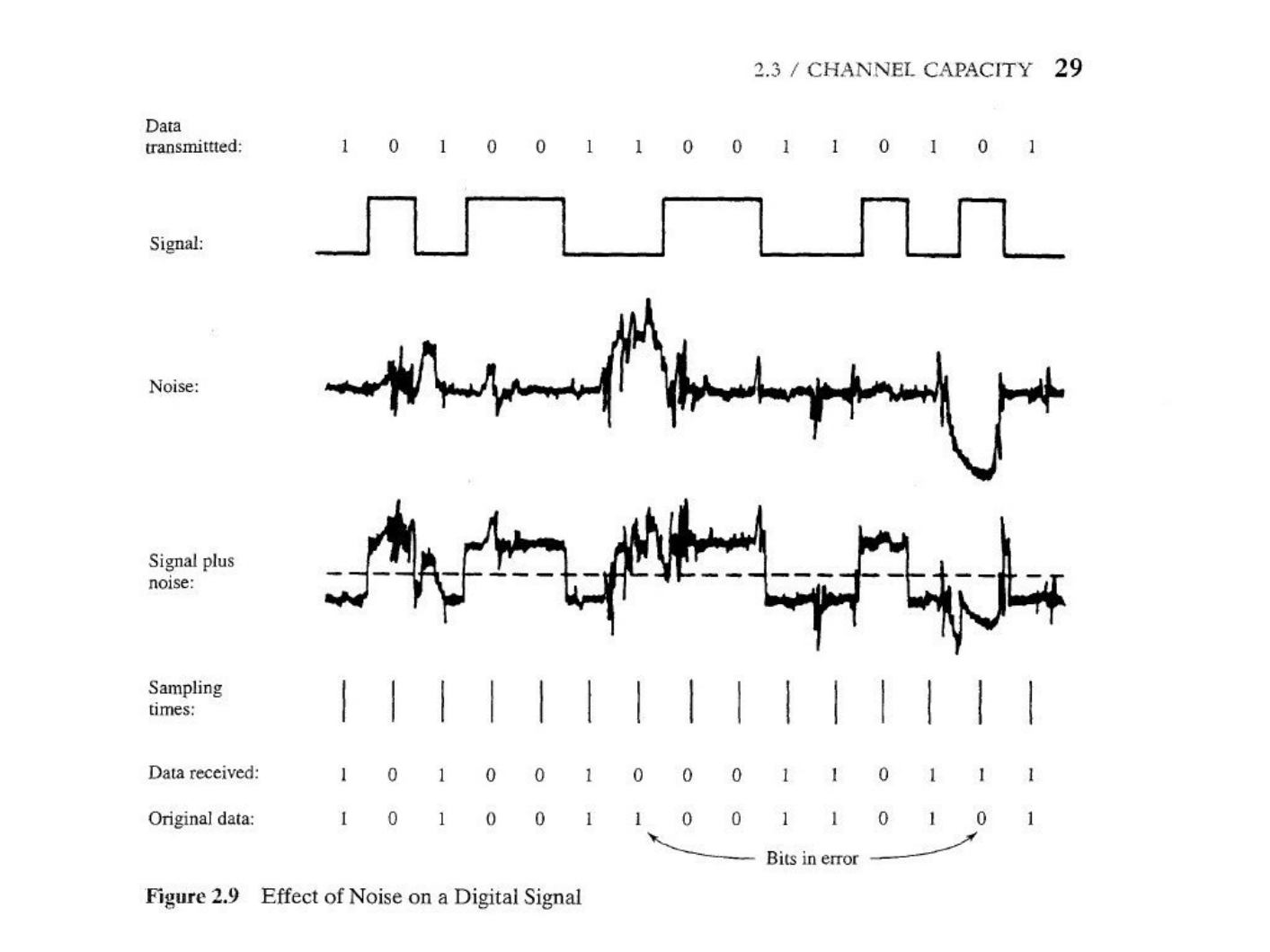

Figure 2.9 Effect of Noise on a Digital Signal

Figura: Efeito do ruído no sinal digital [19]

Motivação

- Popularização dos computadores e as pesquisas espaciais ==> os códigos corretores 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[19]

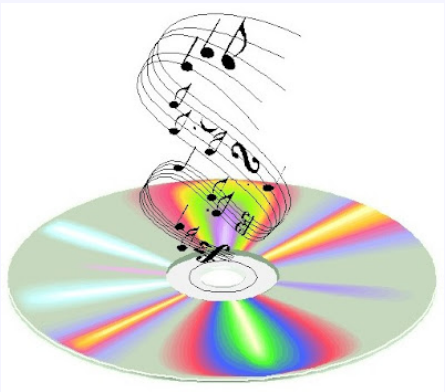

Figura: Ouvir música a partir de um CD[19]

Motivação

Figura: Atender um telefonema[19]

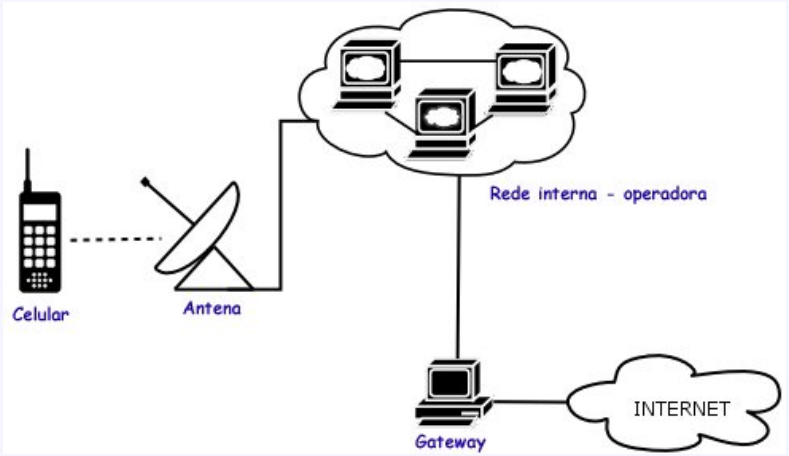

Figura: Navegar pela internet[19]

Motivação

- 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 [1, 2, 6, 28];
  - *Delay and Disruption Tolerant Networks*, redes de sensores e redes *peer-to-peer* [5, 8, 14, 15, 17, 22, 31];
  - armazenamento de grande volume de dados [3, 16, 23, 24, 29, 30, 34], como também o sistema de arquivos distribuído do Hadoop (HDFS) [11].

Motivação

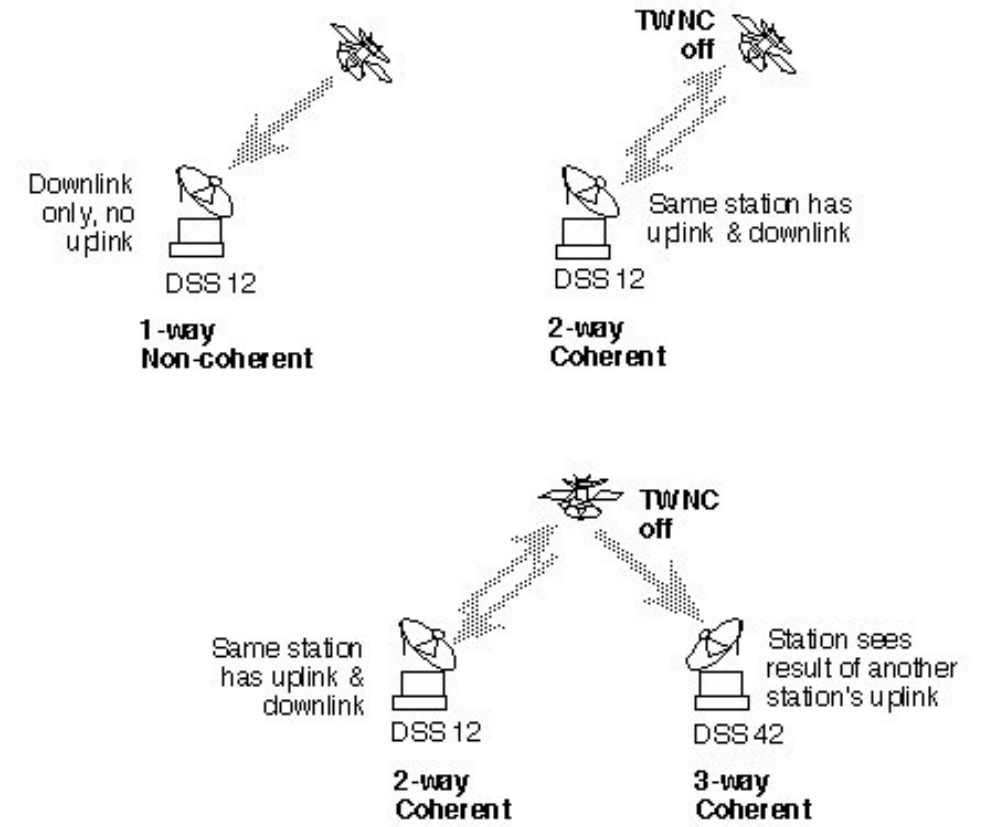

Figura: *NASA's Deep Space Network*[19]

Motivação

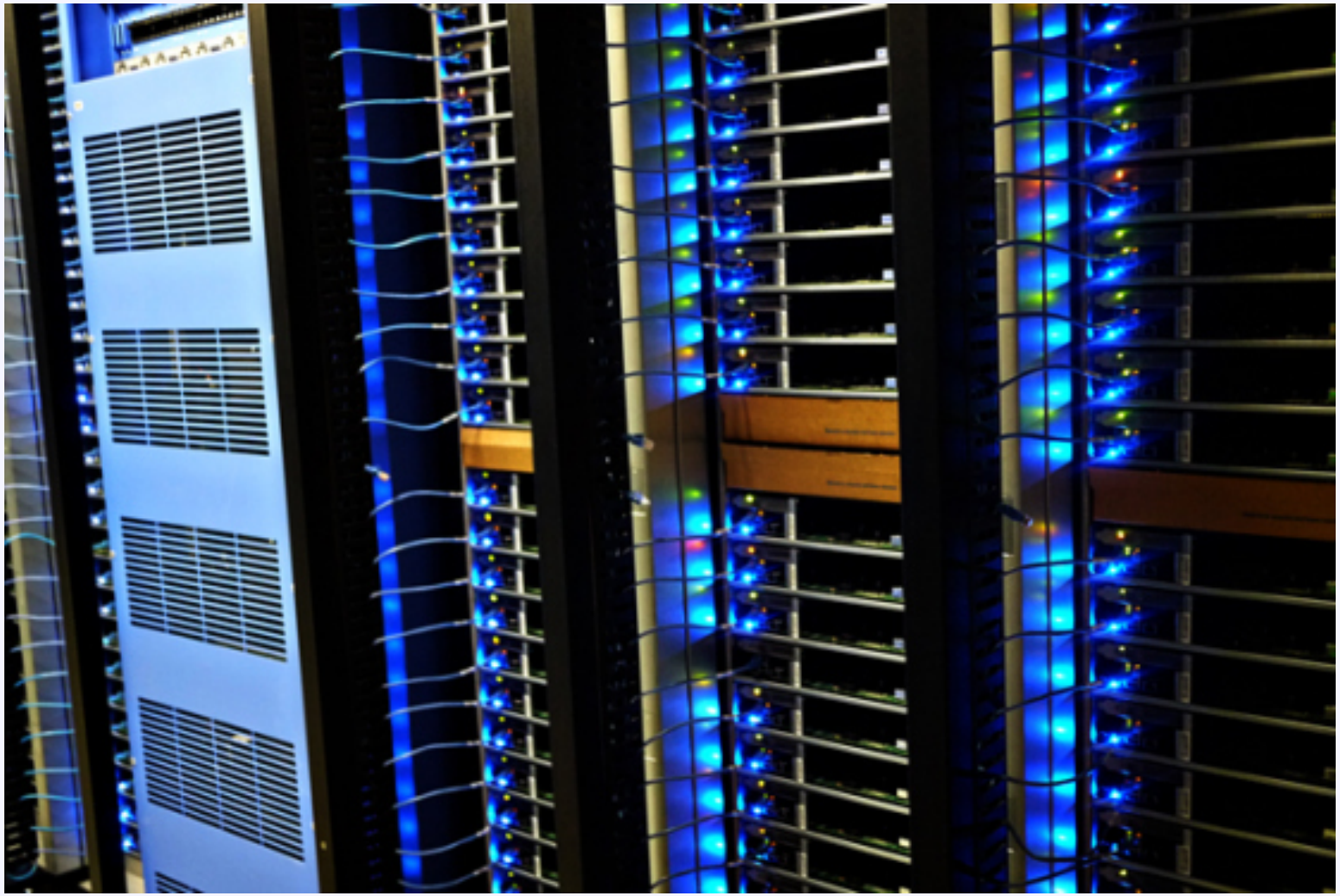

Figura: Facebook data center[19]

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 nessa linha foram feitos utilizando técnicas de *Redundant Array of Independent Drives* (RAID) [11, 18] e mais recentemente do algoritmo Reed-Solomon (RS) [13].

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 [21]. RAID foi introduzido por D. A. Patterson na Universidade da California, Berkeley (UC Berkeley) em 1988 [18].
- O código 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

- Códigos Reed-Solomon (RS) são códigos de bloco, lineares e cíclicos.
- Parametrizáveis ===> capacidade de correção de erros pode ser alterada facilmente.
- RS são códigos MDS ===> 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, 27], 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:

$$(0, 0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0, 0)$$

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 sequência de b bits quaisquer.

Motivação

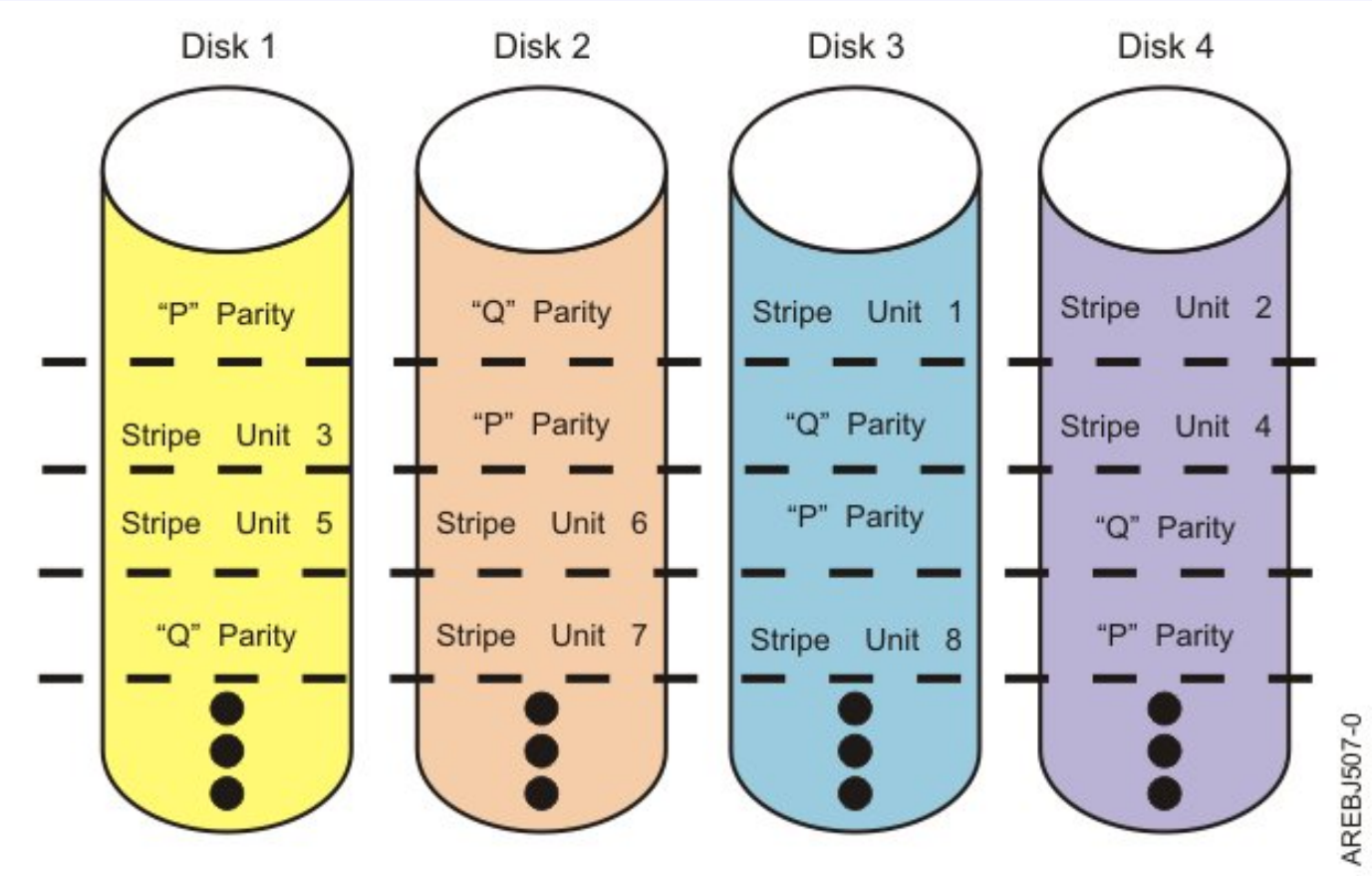

Figura: RAID 6 clássico "fatia"dados e paridade, através dos discos, como num arranjo. A paridade P e Q são geradas pela operação *bitwise* XOR e pelo algoritmo Reed-Solomon.[19]

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 que atualmente implementa Reed-Solomon, tentando melhorar, principalmente, a parte de distribuição de blocos;
- implementação de novos algoritmos (e.g., Tornado codes) e extensão da interface atual para aceitá-los;
- integração do código 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).

Software Open Source

- O Hadoop disponibiliza um armazenamento compartilhado (HDFS/Hadoop *Distributed Filesystem*) e um sistema de análise (MapReduce).
- Facebook, Google, Twitter, Microsoft e IBM armazenam dados do tamanho de petabytes e utilizam o Hadoop.

Software Open Source

- O Hadoop disponibiliza um armazenamento compartilhado (HDFS/Hadoop *Distributed Filesystem*) e um sistema de análise (MapReduce).
- Facebook, Google, Twitter, Microsoft e IBM armazenam dados do tamanho de petabytes e utilizam o Hadoop.
- O projeto Hadoop é hoje um projeto independente dentro da hierarquia de projetos da Apache Software Foundation.

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 *batch*	*batch*
atualização	lê e escreve muitas vezes	escreve uma vez, lê muitas vezes
esquema	estático	dinâmico
integridade	alta	baixa
escala	não linear	linear

Fonte: [33]

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.

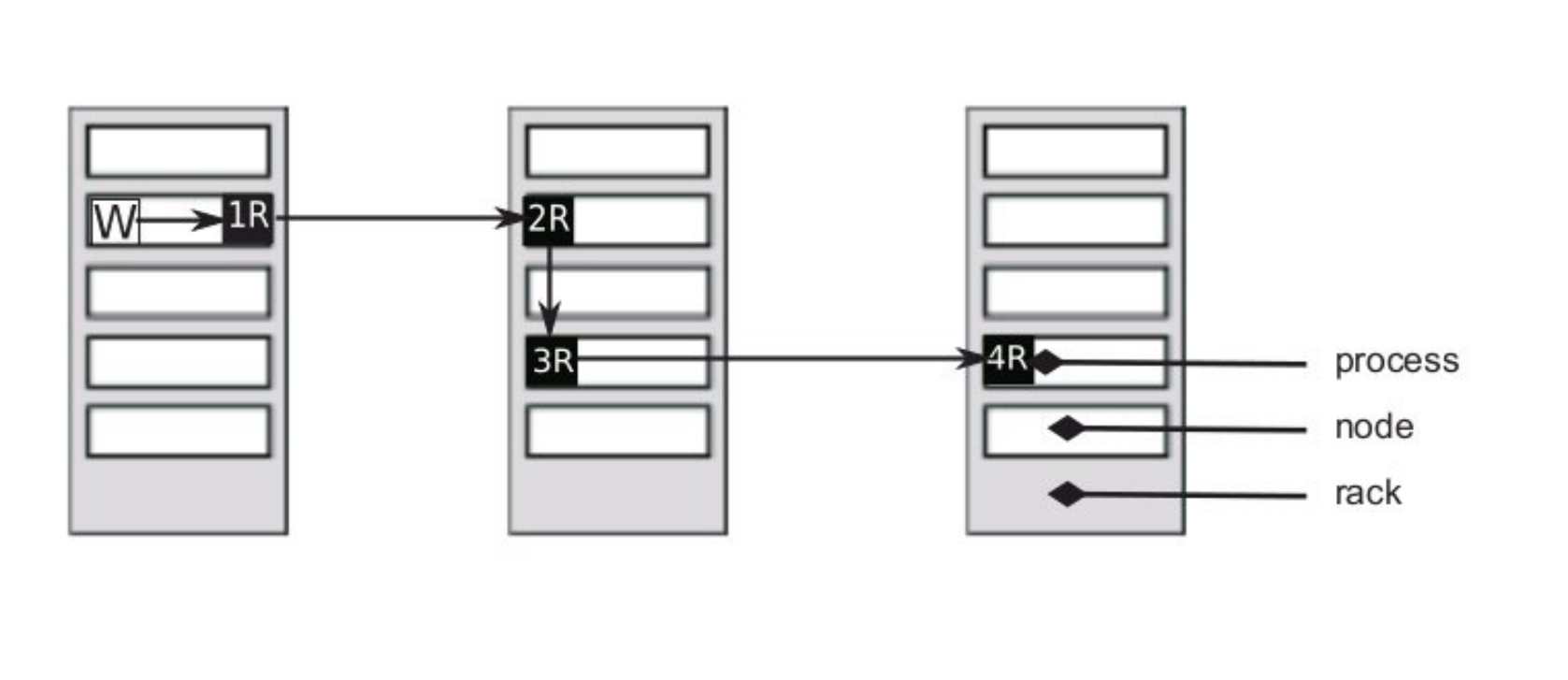

Figura: Arquitetura do HFS - Datanodes e Blocos [12]

MapReduce

- Os dados que MapReduce processa são dados não estruturados como texto ou imagens. O MapReduce 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 MapReduce 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 MapReduce 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)
- 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: 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: RAID e RS
- Artigos e material de disciplinas de universidades de Portugal, Índia, Paquistão e EUA

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: RAID e RS
- Artigos e material de disciplinas de universidades de Portugal, Índia, Paquistão e EUA
- Facebook e Yahoo

Intuições na Álgebra Abstrata

- Congruência linear

Intuições na Álgebra Abstrata

- Congruência linear
- Resto da divisão euclidiana

Intuições na Álgebra Abstrata

- Congruência linear
- Resto da divisão euclidiana
- Independência linear

Intuições na Álgebra Abstrata

- Congruência linear
- Resto da divisão euclidiana
- Independência linear
- Espaço Vetorial, Grupo, Anel e Corpo

Conceitos em Códigos Corretores de Erros

- A teoria da codificação estuda as propriedades dos códigos e suas aplicações. O principal objetivo da codificação é projetar eficientes e confiáveis métodos de transmissão e armazenamento de dados.

Conceitos em Códigos Corretores de Erros

- A teoria da codificação estuda as propriedades dos códigos e suas aplicações. O principal objetivo da codificação é projetar eficientes e confiáveis métodos de transmissão e armazenamento de dados.
- Teoria da Informação, Shannon: incerteza e capacidade do canal

Conceitos em Códigos Corretores de Erros

- A teoria da codificação estuda as propriedades dos códigos e suas aplicações. O principal objetivo da codificação é projetar eficientes e confiáveis métodos de transmissão e armazenamento de dados.
- Teoria da Informação, Shannon: incerteza e capacidade do canal
- Essa teoria foi desenvolvida por Claude Shannon [26] para estudar o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação, área essa da engenharia elétrica e da matemática aplicada. Esses sinais podem ser som, áudio, dados biológicos como eletrocardiogramas ou sequências de DNA [9, 10], sinais de sistemas de telecomunicações, entre muitos outros.

Conceitos em Códigos Corretores de Erros

- A teoria da codificação estuda as propriedades dos códigos e suas aplicações. O principal objetivo da codificação é projetar eficientes e confiáveis métodos de transmissão e armazenamento de dados.
- Teoria da Informação, Shannon: incerteza e capacidade do canal
- Essa teoria foi desenvolvida por Claude Shannon [26] para estudar o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação, área essa da engenharia elétrica e da matemática aplicada. Esses sinais podem ser som, áudio, dados biológicos como eletrocardiogramas ou sequências de DNA [9, 10], sinais de sistemas de telecomunicações, entre muitos outros.
- Canal binário simétrico

Conceitos em Códigos Corretores de Erros

- A teoria da codificação estuda as propriedades dos códigos e suas aplicações. O principal objetivo da codificação é projetar eficientes e confiáveis métodos de transmissão e armazenamento de dados.
- Teoria da Informação, Shannon: incerteza e capacidade do canal
- Essa teoria foi desenvolvida por Claude Shannon [26] para estudar o processamento digital de sinais (sinais em tempo discreto e em tempo contínuo) em um sistema de comunicação, área essa da engenharia elétrica e da matemática aplicada. Esses sinais podem ser som, áudio, dados biológicos como eletrocardiogramas ou sequências de DNA [9, 10], sinais de sistemas de telecomunicações, entre muitos outros.
- Canal binário simétrico
- O 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

- Detectar e codificar tem a mesma complexidade

Conceitos em Códigos Corretores de Erros

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

Conceitos em Códigos Corretores de Erros

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

Conhecer e Compilar o código disponível

- O *kernel* do Hadoop é constituído do core, hdfs e mapred.

Conhecer e Compilar o código disponível

- 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 disponível

- 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 disponível

- 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 disponível

- Eclipse

Conhecer e Compilar o código disponível

- Eclipse
- comandos do linux, principalmente, grep, find

Conhecer e Compilar o código disponível

- Eclipse
- comandos do linux, principalmente, grep, find
- Doxygen (gerar diagramas de herança e de colaboração das classes a partir do código)

Conhecer e Compilar o código disponível

- Eclipse
- comandos do linux, principalmente, grep, find
- Doxygen (gerar diagramas de herança e de colaboração das classes a partir do código)
- Implementar uma aplicação do *framework* (arcabouço) do Hadoop: conta-palavras, temperatura máxima, *page rank*

Conhecer e Compilar o código disponível

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

Conhecer e Compilar o código disponível

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

Conhecer e Compilar o código disponível

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

Conhecer e Compilar o código disponível

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

Conhecer e Compilar o código disponível

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

Conhecer e Compilar o código disponível

- Ambiente de edição do código das classes e de compilação do Hadoop

Conhecer e Compilar o código disponível

- Ambiente de edição do código das classes e de compilação do Hadoop
- Ambiente de instalação do *tarball*

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.
- 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, o código dos procedimentos de compilação e de geração do arquivo *tarball* e o código da implementação das codificações foram disponibilizados em repositório público [1].

[1] https://github.com/celinasam/

Resultados

- Foram criados um site [2] com comentários dos resultados obtidos desse trabalho e um *blog* [3] 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. Esse é uma aplicação das codificações implementadas por esse trabalho.

[2] https://sites.google.com/site/newerasurecodinginhadoop
[3] 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. O teorema da Codificação do Canal [1, 25] afirma 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 [32, 4].

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 [20].
- 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 [7, 10] é uma pesquisa bastante desafiadora e promissora.

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.

T. Anderson, M. Dahlin, J. Neefe, D. Roselli, R. Wang, and D. Patterson.
Serverless network file systems.
Technical report, University of California at Berkeley, Berkeley, CA, USA, 1998.

Referências Bibliográficas II

M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, TJ. Giuli, and P. Bungale.
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.

R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker.
Total recall: system support for automated availability management.
In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 25–25, Berkeley, CA, USA, 2004. USENIX Association.

Referências Bibliográficas III


A. R. Curtis.
Space today online - communicating with interplanetary spacecraft.
URL=http://www.spacetoday.org/SolSys/DeepSpaceNetwork/DeepSpaceNetwork.html. Acessado em 03 de maio de 2010.


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.


Marcelo Sampaio de Alencar.
Telefonia Celular Digital, chapter 6, pages 191–231.
Editora Érica, São Paulo, Brasil, 2004.

Referências Bibliográficas IV


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.


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.


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
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.


A. Haeberlen, A. Mislove, and P. Druschel.
Glacier: highly durable, decentralized storage despite massive correlated failures.
In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 143–158, Boston, MA, USA, 2005. USENIX Association.

Referências Bibliográficas VI

Y. Houri, M. Jobmann, and T. Fuhrmann.
Self-organized data redundancy management for peer-to-peer storage systems.
In *IWSOS '09: Proceedings of the 4th IFIP TC 6 International Workshop on Self-Organizing Systems*, pages 65–76, Berlin, Heidelberg, 2009. Springer-Verlag.

J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao.
Oceanstore: an architecture for global-scale persistent storage.
In *ASPLOS '00: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, ASPLOS-IX, pages 190–201, New York, NY, USA, 2000. ACM.

Referências Bibliográficas VII


C. T. Oliveira, M. D. D. Moreira, M. G. Rubinstein, L. H. M. K. Costa, and O. C. M. B. Duarte.
Mc05: Redes tolerantes a atrasos e desconexões.
In *Anais do 25o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Belém, Pará, Brasil, May 2007.


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.


J. S. Plank and M. G. Thomason.
A practical analysis of low-density parity-check erasure codes for wide-area storage applications.
In *DSN*, pages 115–124, 2004.

Referências Bibliográficas VIII


Wikipedia project.
Joint source and channel coding.
URL=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.


R. Rodrigues and B. Liskov.
High availability in dhts: Erasure coding vs. replication.
In *Peer-to-Peer Systems IV*, pages 226–239. LNCS, 2005.

Referências Bibliográficas IX

Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence.
Fab: building distributed enterprise disk arrays from commodity components.
In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, ASPLOS-XI, pages 48–58, New York, NY, USA, 2004. ACM.

F. Schmuck and R. Haskin.
Gpfs: A shared-disk file system for large computing clusters.
In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, Berkeley, CA, USA, 2002. USENIX Association.

Mischa Schwartz.
Information transmission, modulation and noise, chapter 3, 7, pages 95–200, 564–730.
McGraw-Hill, Singapure, 1990.

Referências Bibliográficas X


C. E. Shannon.
A mathematical theory of communication.
The Bell System Technical Journal, 27(3):379–423, 623–656, 1948.


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.


R. W. Sniffin.
Telemetry data decoding.
URL=http://deepspace.jpl.nasa.gov/dsndocs/810-005/208/208A.pdf. Acessado em 03 de maio de 2010.

Referências Bibliográficas XI


M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti.
Pergamum: replacing tape with energy efficient, reliable, disk-based archival storage.
In *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, pages 1:1–1:16, Berkeley, CA, USA, 2008. USENIX Association.


M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti.
Potshards: a secure, recoverable, long-term archival storage system.
Trans. Storage, 5:5:1–5:35, June 2009.


Z. Wilcox-O'Hearn B. Warner.
Tahoe: the least-authority filesystem.
In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, StorageSS '08, pages 21–26, New York, NY, USA, 2008. ACM.

Referências Bibliográficas XII


H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz.
Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems.
21st IEEE Symposium on Reliable Distributed Systems, 0:362–367, 2002.


T. White.
Hadoop: The Definive Guide, chapter 1, 2, 3, pages 1–74.
O'Reilly, Sebastopol, CA, USA, 2009.


H. Xia.
Robustore: a distributed storage architecture with robust and high performance.
PhD thesis, University of California at San Diego, La Jolla, CA, USA, 2006.
AAI3225997.