Proposta de Dissertação de Mestrado

Estudo e Implementação de Mecanismos de Codificação por Apagamento no Sistema de Arquivos Distribuído do Hadoop

Celina d' Ávila Samogin
Instituto de Computação — UNICAMP

22 de novembro de 2010

Orientadora: Profa. Dra. Islene Calciolari Garcia

Agenda

Motivação

- Esta proposta é uma contribuição para *software* livre em sistemas distribuídos.
- Armazenamento de arquivos é um componente essencial na computação de alto desempenho.
- Codificação por apagamento (*Erasure codes*) introduz redundância e tem sido utilizada em sistemas para alcançar confiabilidade e redução do custo de armazenamento.

Motivação

- Alguns sistemas que utilizam codificação por apagamento:
  - *NASA's Deep Space Network* para receber sinais e dados de telemetria (*downlinks*) vindos de veículos espaciais (*very distant spacecrafts*) e para enviar telecomandos (*uplinks*) para veículos espaciais [18, 19, 24];
  - *Delay and Disruption Tolerant Networks*, redes de sensores e redes *peer-to-peer* [3, 4, 22];
  - armazenamento de grande volume de dados [2, 5, 12].

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) [15, 8] e mais recentemente do algoritmo Reed-Solomon (RS) [14, 9, 10].

Objetivos desta proposta

- avaliar desempenho das camadas de codificação do hadoop, o tamanho do armazenamento;
- otimizar, estender o código da camada RAID, da camada RS;
- incluir novas codificações como a codificação Tornado e integrar o código atual com o HDFS.

Agenda

Codificação por Apagamento

- Shannon demonstrou essa teoria em artigo do Bell System Technical Journal [6] de 1948.
- Existem dois métodos básicos para tratar erros em comunicação e ambos envolvem a codificação de mensagens. A diferença está em como esses códigos são utilizados:
  - Em um *repeat request system*, os códigos são utilizados para detectar erros e se estes existirem, é feito um pedido de retransmissão.
  - Com *forward error correction*, os códigos são usados para detectar e corrigir erros.

Codificação por Apagamento

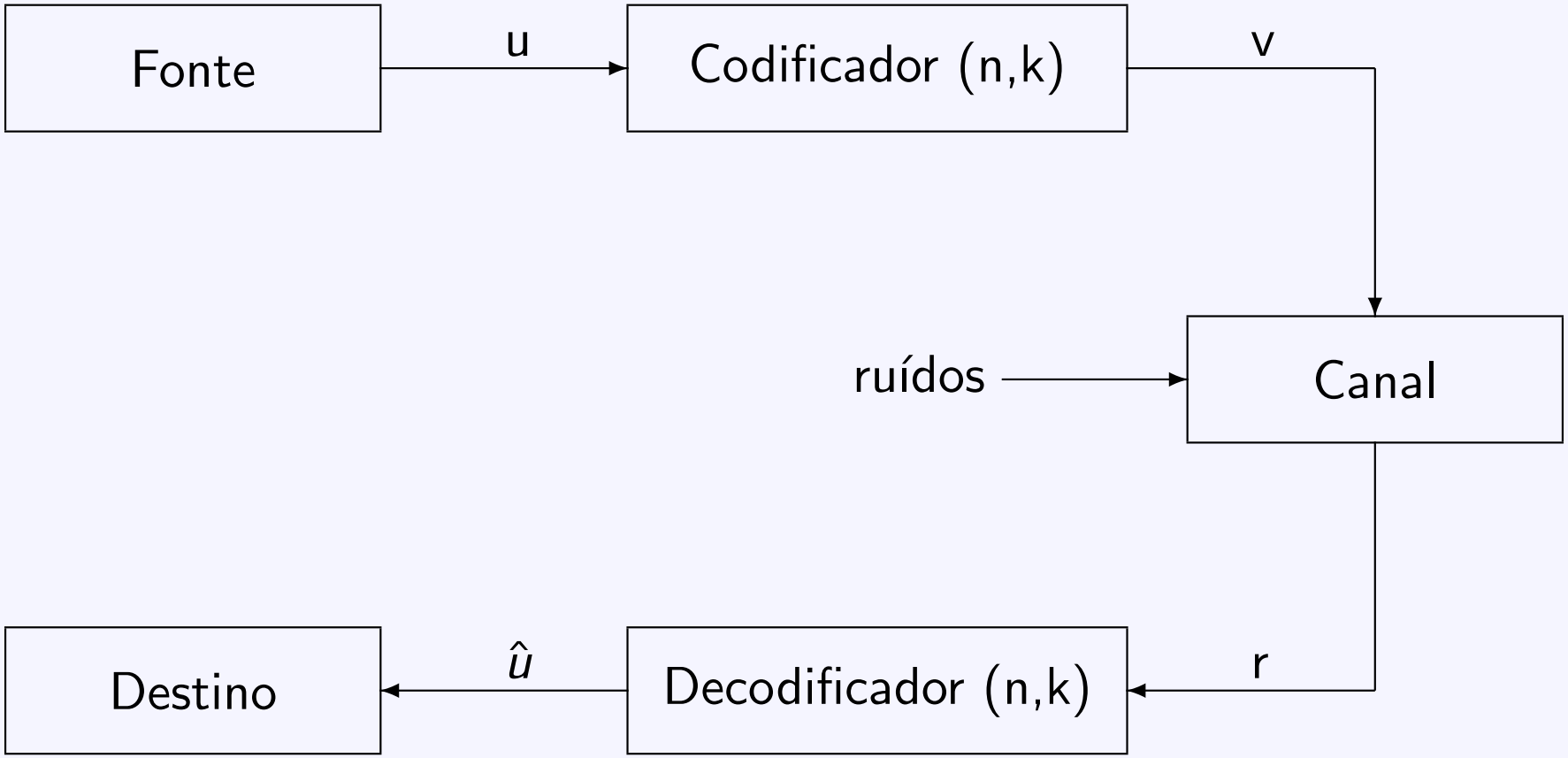

Figura: Códigos de bloco

Sistema de arquivos distribuídos armazenando um arquivo - Replicação Simples

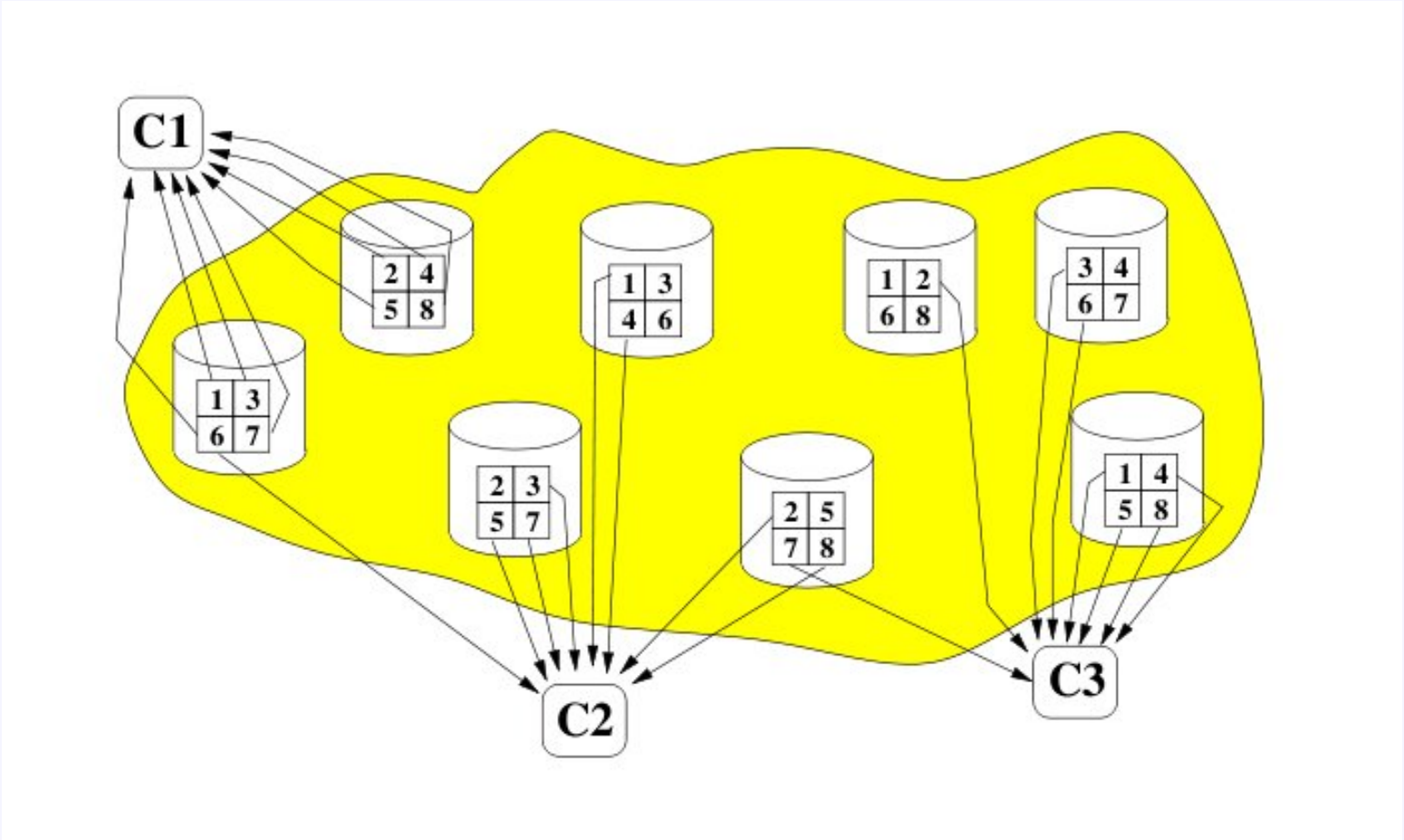

Figura: Sistema com replicação simples [13]

Sistema de arquivos distribuídos armazenando um arquivo - Codificação por Apagamento

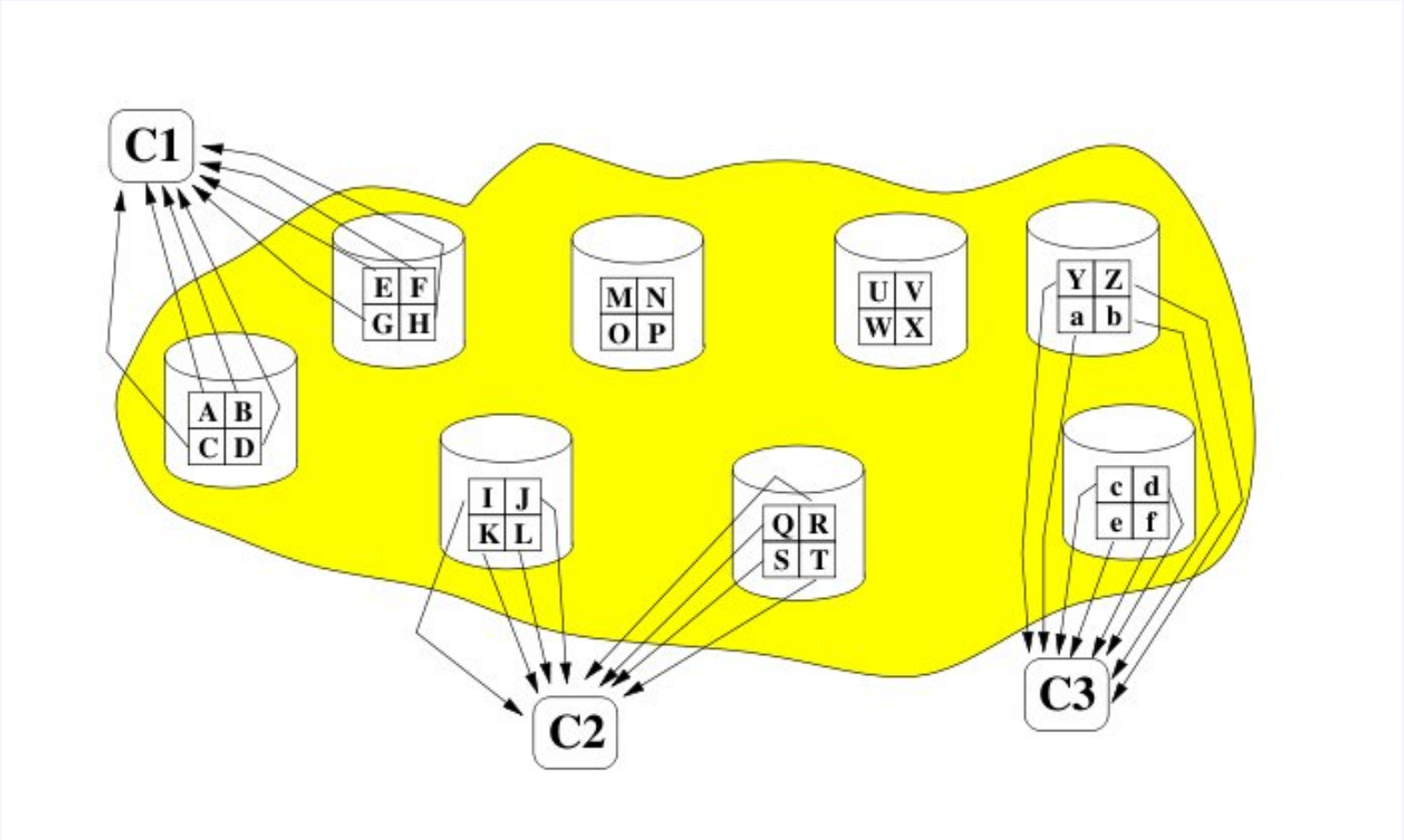

Figura: Sistema com codificação por apagamento [13]

RAID-5

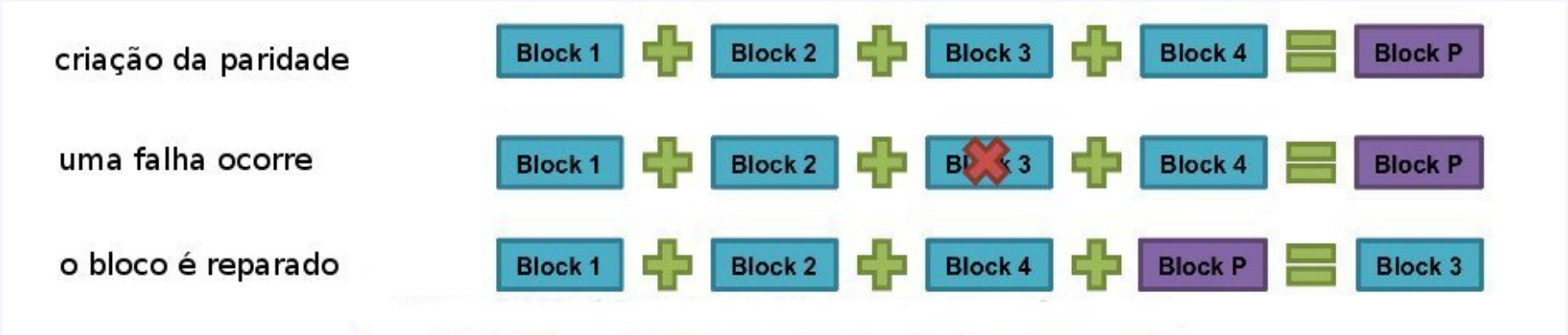

Figura: RAID-5: 1 bloco de paridade e *stripe* = 4 blocos [11]

RAID-6

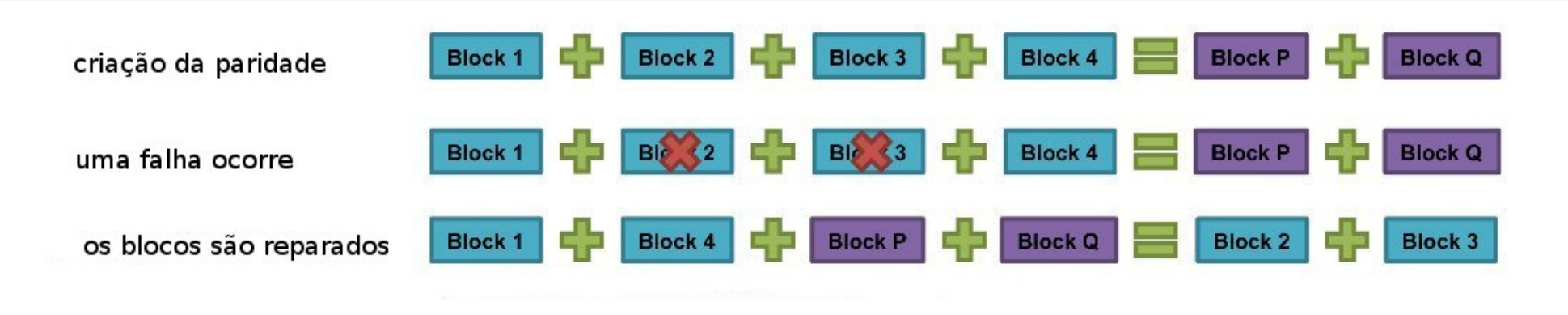

Figura: RAID-6: 2 blocos de paridade e *stripe* = 4 blocos [11]

Codificação RS e Tornado

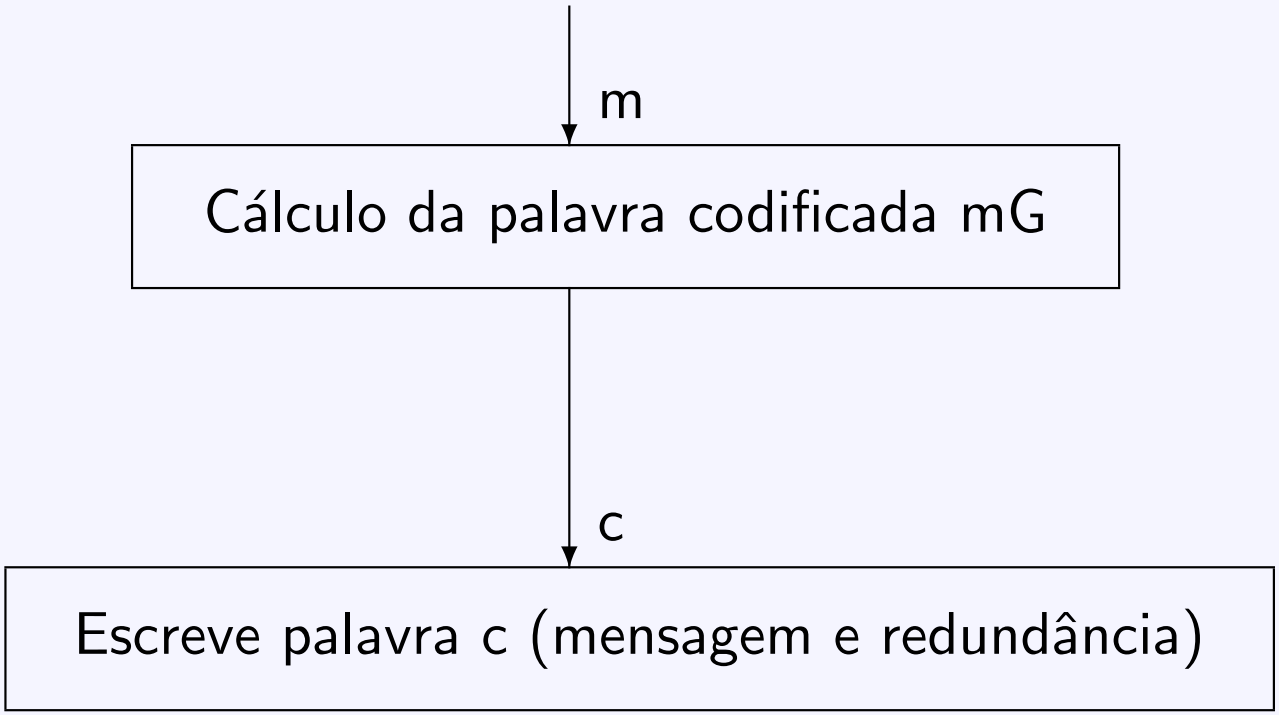

Figura: Algoritmo de codificação

Decodificação RS e Tornado

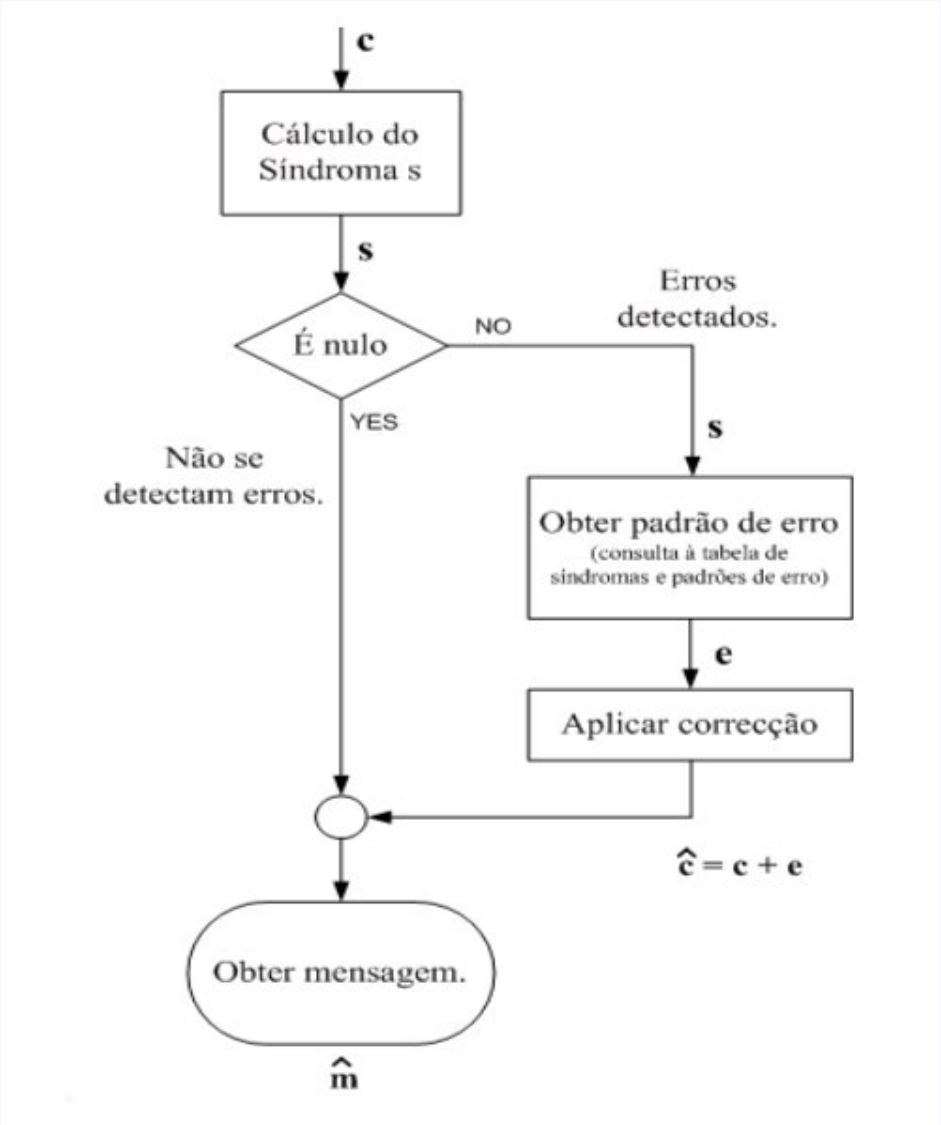

Figura: Algoritmo de decodificação [16]

Trabalho	Modelo	Objetivos	Codificação	Métricas
Byers,1998 [1]	*Digital Fountain*	Confiabilidade, Eficiência	Replicação, códigos Tornado e RS	tempo de codificação e decodificação de um bloco, *bandwidth*, perda de pacotes
Weatherspoon, 2002 [5]	Arquivador central, nós	Disponibilidade	Replicação, Codificação por Apagamento	tempo médio entre falhas, *overhead* de armazenamento, tempo de verificação do bloco
Dabek, 2004 [20]	DHT	Eficiência	Replicação, Codificação por Apagamento	latência
Camargo, 2009 [7]	OppStore	Confiabilidade, Disponibilidade	Replicação	*bandwidth*, número de mensagens trocadas
Fan, 2009 [12]	HDFS	Confiabilidade, Disponibilidade	Replicação, RAID	*overhead* de armazenamento
Houri, 2009 [22]	*Peer-to-peer*	Disponibilidade	Replicação, Codificação por Apagamento	*bandwidth*
Plank, 2009 [26]	k discos de dados e m discos de paridade	Eficiência	códigos Tornado, RS e RAID	tempo de codificação de um grande arquivo de vídeo e tempo de decodificação de um *drive* de dados
Esta proposta	HDFS e MapReduce	Redução do custo de armazenamento	Replicação, códigos RAID, RS e Tornado	*overhead* de armazenamento, tempo de latência de leitura de arquivos

Tabela: Comparação entre sistemas de codificação por apagamento

Agenda

Hadoop

- 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 grande volume de dados.
- 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} - dados processados em uma hora pela Google
  - exabyte - 10^{18}
  - zettabyte - 10^{21} - um disco de 140 GB para cada pessoa no mundo

Hadoop

- O Hadoop é um *framework* que foi criado por Doug Cutting para ser uma implementação *open source* de algoritmos de motores de busca.
- Facebook, Yahoo, Twitter, Microsoft e IBM e por laboratórios de Universidades: University of Maryland, Cornell University, University of Edinburg, Unicamp [25]
- Apache *Software Foundation*

Hadoop

- O Hadoop não é um *framework* canônico:
  - Arquitetura é mestre/escravo.
  - Projetado para aplicações que atualizam dados da seguinte forma:
    - ★ uma escrita e muitas leituras através de acessos por *batch*
    - ★ dados com tamanho da ordem de petabytes, organizados de forma não estruturada, com esquema dinâmico e integridade baixa.
  - As escritas são feitas somente no final do arquivo.
- Kernel do Hadoop: um armazenamento compartilhado (HDFS) e um sistema de análise (MapReduce)

Mapreduce

- O MapReduce pode resolver problemas genéricos, cujos dados podem ser divididos em matrizes de dados, para cada matriz a mesma computação necessária (sub-problema) e não existe comunicação entre as tarefas (sub-problemas).
- Problemas como empacotamento, linha de fábrica, otimização não são resolvidos pelo modelo de computação do MapReduce.
- A execução de um típico *job* do MapReduce pode ser assim descrita:
  - 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

Arquitetura do HDFS

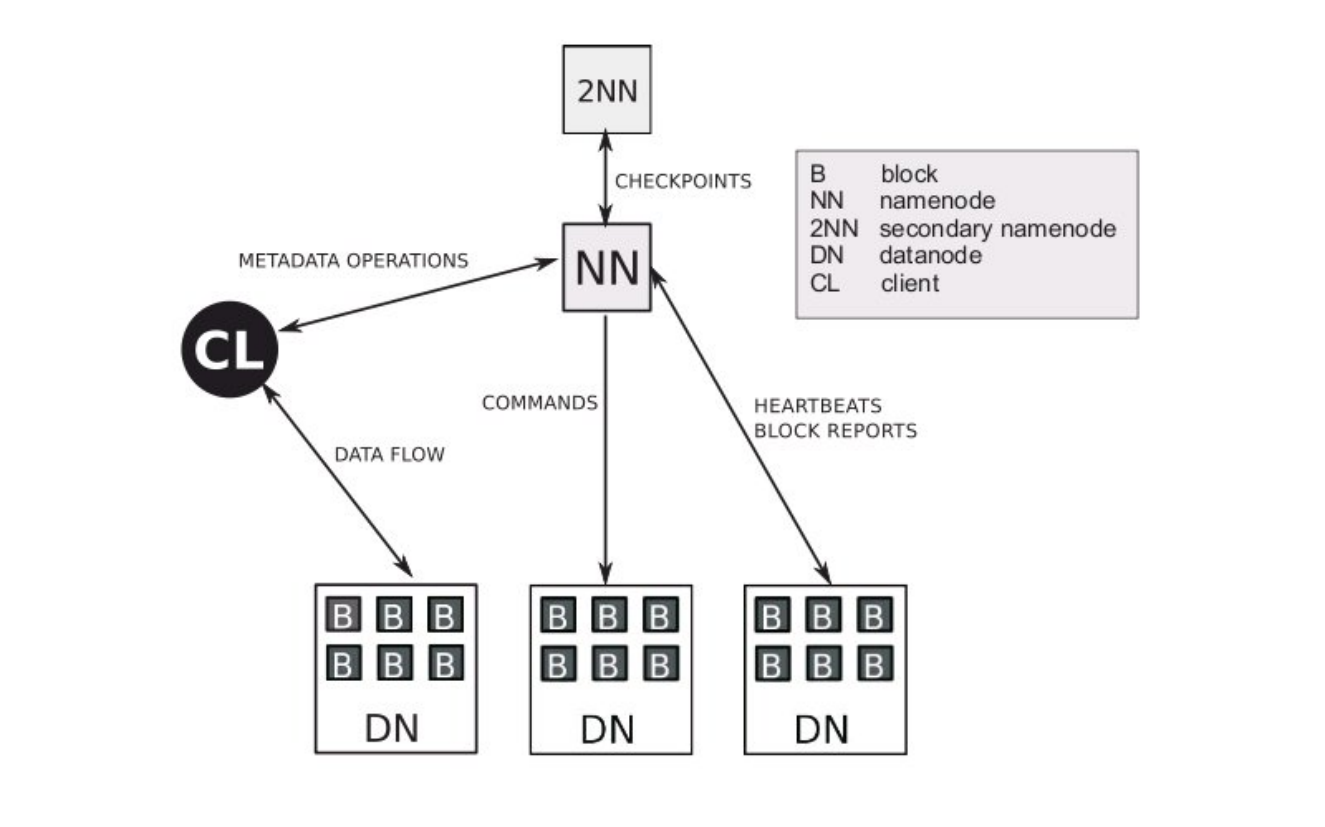

Figura: Arquitetura do HDFS [21]

Arquitetura do HDFS

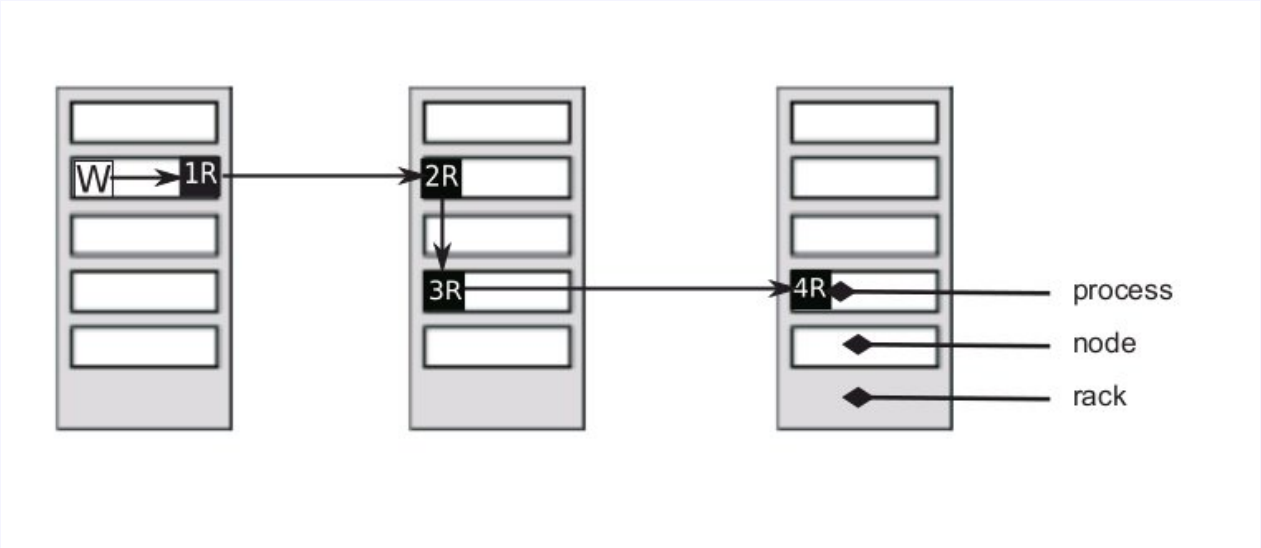

Figura: Arquitetura do HFS - *Pipeline* dos Datanodes e Blocos [23]

Codificação por Apagamento no HDFS

- Em 2009, foi proposta uma implementação de camada de codificação por apagamento no Hadoop utilizando RAID [8] e uma mais recentemente utilizando códigos RS [9].
- A versão atual do Hadoop utiliza apenas a técnica de replicação [23] para obter disponibilidade e confiabilidade de dados. A inclusão da codificação por apagamento será feita com o objetivo de reduzir o tamanho do armazenamento do HDFS.

Camada RAID - Discussões

- Existem discussões sobre o número de blocos por grupo de paridade [11].
- O número de blocos por grupo de paridade poderia ser diferente para arquivos pequenos e para arquivos grandes: 80% dos 7 *clusters* do Yahoo! são arquivos com menos de $8GB$ ($64x128MB$). Enquanto que mais de 75% do espaço é usado para armazenar diretórios, cujo tamanho (excluindo os sub-diretórios) é mais do que $8GB$ ($64x128MB$).
- O agrupamento poderia ser feito por diretórios de arquivos.

Camada RAID - Discussões

- Sobre o arquivo de paridade, ele poderia ser:
  - 1 - um arquivo de paridade para cada grupo (o uso da memória do namenode aumentaria, pois o número de grupos pode ser grande)
  - 2 - um arquivo de paridade é gerado para cada *map* de cada *job* do MapReduce (muitos blocos de paridade poderão ser regerados desnecessariamente em caso de falha)
  - 3 - um arquivo de paridade é gerado como em 2, esse arquivo de paridade é divido em vários fragmentos, um para cada bloco de paridade (pode usar muito espaço se o bloco de paridade for pequeno, pois o HDFS ainda não suporta arquivo espalhado)

Camadas RS e Tornado - Discussões

- $n = 4m$ é número de blocos codificados a partir a um bloco inicial
- armazenamento dos blocos codificados como no *pipeline* dos blocos replicados

Agenda

Objetivos

- avaliação de desempenho, ganhos, e custos de diferentes estratégias de codificação por apagamento;
- implementação de otimizações ou extensões para o código que atualmente implementa a codificação RS, tentando melhorar, principalmente, a parte de distribuição de blocos;
- implementação de novos algoritmos (codificação Tornado) e extensão da interface atual para aceitá-los;
- integração do código atual com o HDFS.

Métodos

Os testes poderão utilizar:

- máquinas do Instituto de Computação da Unicamp, principalmente do LSD (Laboratório de Sistemas Distribuídos);
- máquinas do ambiente computacional do CENAPAD-SP (Centro Nacional de Processamento de Alto Desempenho em São Paulo);
- a nuvem do AWS (Amazon *Web Services*).

Métodos

1	sem codificação, fator replicação $= r$	baixa em relação ao espaço de armazenamento	$O(p^r)$	rx
2	Códigos RAID, 1 bloco de paridade, *stripe* $= 10$ blocos, fator replicação $= 2$	permite falha em 1 máquina	$O(p^4)$	$2.2x$
3	Códigos RS RAID, 4 blocos de paridade, *stripe* $= 10$ blocos, fator replicação $= 1$	permite falha em 4 máquinas	$O(p^5)$	$1.4x$
4	Códigos RS, fator replicação $= 4$, com n máquinas	permite falha em até $3m$ máquinas	$\Omega(p^{3m+1})$	$4x$
5	Códigos Tornado, fator replicação $= 4$, com n máquinas	permite falha em até $3m$ máquinas	$\Omega(p^{3m+1})$	$4x$

Tabela: Comparação entre algoritmos de codificação por apagamento e replicação

onde:
p = probabilidade de perda do bloco, $0 < p < 1$
x = tamanho do armazenamento em disco de um bloco
m = número de fragmentos do bloco inicial antes da codificação
$n = 4m$ é número de blocos codificados a partir a um bloco inicial
o bloco codificado b_i está armazenado na máquina d_i, para $1 \leq i \leq n$; a máquina d_i e a d_{i+1} são distintas e estão no mesmo rack

stripe = número de blocos de um arquivo que são combinados em um único bloco de paridade

Métodos - Camada RAID - Exemplo do algoritmo de codificação

O tamanho da *stripe* é 10 blocos e existe um arquivo */a/arquivo.txt* com exatamente 10 blocos. Nesse caso, o algoritmo de codificação da camada RAID faz o seguinte:

- bloco[0] = primeiro bloco
- bloco[1] = segundo bloco
- ...
- bloco[9] = último bloco
- bloco_paridade = iniciado com 0 em todos os bytes
- para i de 0 até número de bytes em um bloco:
  - para j de 0 até 9:
    - bloco_paridade = bloco_paridade *xor* bloco[*j*][*i*]
- para i de 0 até 9:
  - escreva bloco_paridade no arquivo */raid/a/arquivo.txt*

Métodos - Camada RAID - Exemplo do algoritmo de codificação

- O número de blocos da *stripe* é parametrizável.
- As operações entre os *bits* de cada bloco em uma *stripe* são realizadas em ordem e os *bits* de cada operação são gravados em blocos diferentes de paridade.
- Os blocos de paridade ficam armazenados em um arquivo de paridade.
- Existe um mapeamento um-para-um entre o arquivo e seu arquivo de paridade.

Métodos

- *Code Review Checklist* segue Java *Code Conventions* de 1997
- Após revisão, é sugerido marcar a *Reviewed flag* na discussão do Jira

Forma de Análise dos Resultados

Nós poderemos utilizar alguns *e-books* do Projeto Gutenberg e do Portal Domínio Público como entrada de dados de alguns dos testes:

Teste de funcionalidade dos algoritmos de codificação e de decodificação RS e da codificação Tornado

Teste *cluster* Hadoop 0.21.0 que utiliza apenas replicação

Teste *cluster* Hadoop 0.21.0 com a camada RAID que utiliza replicação e codificação por apagamento [8]

Teste *cluster* Hadoop 0.22.0 com a camada RS que utiliza replicação e codificação por apagamento [9, 10]

Teste *cluster* Hadoop com a camada Tornado que utiliza replicação e codificação por apagamento

Forma de Análise dos Resultados

Estamos prevendo duas fases de teste:

testes de funcionalidade e de injeção de falhas testar os algoritmos que criam os blocos codificados (dados e redundância) e os mantêm; testar os algoritmos que atendem os pedidos de leitura de blocos codificados em diferentes codificações; testar os algoritmos que percebem réplicas indisponíveis e as reconstroem a partir dos blocos codificados (para isso utilizar possivelmente o Zookeeper, um serviço de coordenação de processos de aplicações em sistemas distribuídos); testar os algoritmos que percebem blocos indisponíveis e reconstroem as réplicas (se indisponíveis); esta fase será executada em ambiente virtualizado;

Forma de Análise dos Resultados

testes de desempenho, de tamanho do armazenamento e de injeção de falhas

obter uma aproximação do tamanho do armazenamento (dados e paridade) para conjuntos de arquivos que ocupem espaço original do tamanho de alguns gigabytes, terabytes e petabytes; medir o tempo de latência de leitura de arquivos; esta fase será executada em ambiente o mais real possível.

Forma de Análise dos Resultados

Os algoritmos de codificação e de decodificação poderão permitir parametrizar:

- número de blocos de paridade (redundância);
- fator de replicação.

Plano de Trabalho e Cronograma

1. Créditos do mestrado
2. Exame de qualificação do mestrado
3. Revisão bibliográfica
4. Implementação
5. Realização de testes
6. Escrita da dissertação de mestrado
7. Preparação de artigo para congresso ou revista
8. Defesa da dissertação

Atividade	2010					2011						2012
	3–4	5–6	7–8	9–10	11–12	1–2	3–4	5–6	7–8	9–10	11–12	1–2
1	o	o	o	o	o							
2				o								
3	o	oo	oo	oo	oo	oo	oo	o	o	o	o	
4			o	oo	oo	oo	oo	oo	oo	oo	oo	
5		o	o	oo	oo	oo	oo	oo	oo	oo	oo	
6		oo	oo	oo	oo	oo	o	o	o	o	o	
7								o	o	o		
8												o

Contribuições esperadas

- *Overview* de Codificação por Apagamento - são poucas as pesquisas experimentais publicadas sobre o tema; não existe uma nomenclatura unificada; poucos pesquisadores, que são programadores de sistemas, fazem propostas neste tema.
- Submeter as alterações e sugestões como contribuição ao Hadoop através de pequenos *patchs*.
- Algumas das ferramentas que a comunidade de mantenedores usa:
  - jira, *issue tracking and project tracking for software development teams*, sistema proprietário da Atlassian;
  - git, sistema de controle de versão, licença gpl, histórico de *commits* ;
  - svn (subversion), sistema de controle de versão, licença apache, repositório oficial 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
- Buscas no jira 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: 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 deste Instituto, que sugeriu o tema deste trabalho e que tem contribuído com várias idéias para a realização deste trabalho.
- Esperamos interação e colaboração com os desenvolvedores.

Agenda

Referências Bibliográficas I


Byers, J. W. Luby, M. Mitzenmacher, M. Rege, A.
A digital fountain approach to reliable distribution of bulk data.
SIGCOMM Computer Communication Rev., 28(4):56–67, 1998.


Kubiatowicz, J. Bindel, D. Chen, Y. Czerwinski, S. Eaton, P. Geels, D. Gummadi, R. Rhea, S. Weatherspoon, H. Weimer, W. Wells, C. Zhao, B.
Oceanstore: an architecture for global-scale persistent storage.
SIGPLAN Not., 35(11):190–201, 2000.


Oliveira, C. T. Moreira, M. D. D. Rubinstein, M. G. Costa, L. H. M. K. Duarte, O. C. M. B.
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.

Referências Bibliográficas II


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


Weatherspoon, H. Kubiatowicz, H. J. D.
Erasure coding vs. replication: A quantitative comparison.
In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 328–338, London, UK, 2002. Springer-Verlag.


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

Referências Bibliográficas III


Camargo, R. Y. Filho, F. C. Kon, F.
Efficient maintenance of distributed data in highly dynamic opportunistic grids.
In *SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1067–1071, New York, NY, USA, 2009. ACM.


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.


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.

Referências Bibliográficas IV


Apache Software Foundation.
Mapreduce-1970 - reed-solomon code implementation to be used in raid.
URL=https://issues.apache.org/jira/browse/MAPREDUCE-1970.
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.


Fan, B. Tantisiriroj, W. Xiao, L. Gibson, G.
Diskreduce: Raid for data-intensive scalable computing.
In *PDSW '09: Proceedings of the 4th Annual Workshop on Petascale Data Storage*, pages 6–10, New York, NY, USA, 2009. ACM.

Referências Bibliográficas V


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


Reed, I. S. Solomon, G.
Polynomial codes over certain finite fields.
Journal of the Society for Industrial and Applied Mathematics, 8(2):300–304, 1960.


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

Ferreira, A. J.
material de apoio - códigos detectores e correctores de erros, introdução à codificação de canal e aos códigos detectores e correctores de erros, cíclicos e não cíclicos.
URL=http://www.deetc.isel.ipl.pt/sistemastele/docentes/AF/AF.htm.
Acessado em 24 de julho de 2010.

Lin, S. Costello, D. J. J.
Error Control Coding: Fundamentals and Applications, chapter 1, pages 1–14.
Prentice-Hall Press, Englewood Cliffs, New Jersey, USA, 1983.

Almeida, G. M.
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.

Referências Bibliográficas VII


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


Dabek, F. Li, J. Sit, E. Robertson, J. Kaashoek, M. F. Morris, R.
Designing a dht for low latency and high throughput.
In *NSDI'04: Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation*, pages 85–98, Berkeley, CA, USA, 2004. USENIX Association.


Oriani, A. Garcia, I. C. Schmidt, R.
The Search for a Highly-Available Hadoop Distributed Filesystem.
Technical Report IC-10-24, Institute of Computing, University of Campinas, August 2010.

Referências Bibliográficas VIII


Houri, Y. Jobmann, M. Fuhrmann, T.
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.


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


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

Referências Bibliográficas IX

Hadoop Wiki.
Hadoop wiki - poweredby.
URL=http://wiki.apache.org/hadoop/PoweredBy. Acessado em 08 de maio de 2010.

Plank, J. S. Xu, L. Luo, J. Schuman, C. D. Wilcox-O'Hearn, Z.
A performance evaluation and examination of open-source erasure coding libraries for storage.
In *FAST '09: Proceedings of the 7th Conference on File and Storage Technologies*, pages 253–265, Berkeley, CA, USA, 2009. USENIX Association.