UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS EXATAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARIA TERESA KRAVETZ ANDRIOLI

TITULO

CURITIBA

2022

MARIA TERESA KRAVETZ ANDRIOLI

TITULO

Trabalho apresentado como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação no curso de Ciência da Computação, Setor de Ciências Exatas da Universidade Federal do Paraná.

Orientador: Prof. Dr. Luiz Carlos P. Albini

CURITIBA

2022

TERMO DE APROVAÇÃO

MARIA TERESA KRAVETZ ANDRIOLI

TITULO

Trabalho apresentado como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação no curso de Ciência da Computação, Setor de Ciências Exatas da Universidade Federal do Paraná, pela seguinte banca examinadora:

Prof. Dr. Luiz Carlos P. Albini
Orientador

Professora
UFPR

Professora

Professora

Curitiba, Maio de 2022.

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz[1] e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com LaTeX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação[2] da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*[3] e aos novos voluntários do grupo *abnTeX2*[4] que contribuíram e que ainda contribuirão para a evolução do abnTeX2.

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz[5] e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com LaTeX fosse possível.

[1] Os nomes dos integrantes do primeiro projeto abnTeX foram extraídos de http://codigolivre.org.br/projects/abntex/

[2] http://www.cpai.unb.br/

[3] http://groups.google.com/group/latex-br

[4] http://groups.google.com/group/abntex2 e http://abntex2.googlecode.com/

[5] Os nomes dos integrantes do primeiro projeto abnTeX foram extraídos de http://codigolivre.org.br/projects/abntex/

RESUMO

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.Ter no máximo 500 palavras!!! As palavras chave são separadas por ponto e vírgula.

Palavras-chaves: latex; abntex; editoração de texto.

ABSTRACT

This is the english abstract.

Key-words: latex. abntex. text editoration.

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

O uso, armazenamento e controle de dados é um tema muito discutido na área de computação desde seus primórdios até os dias de hoje. Por causa disso, muitos métodos e algoritmos e termos surgiram ao longo do tempo com o objetivo de gerenciar de forma eficiente esses dados. O surgimento dessas novas ferramentas computacionais e métodos de armazenamento foi importantíssimo para a evolução da área.

Atualmente, os métodos mais comuns são os bancos de dados relacionais e *datas warehouses* usando computação em nuvem (KUO; KUSIAK, 2019). Além disso, pesquisas nos campos de mineração de dados e aprendizagem de máquina cresceram bastante recentemente de modo a prover técnicas que permitissem analisar dados complexos e variados entre si (BELCASTRO et al., 2022). Um grande desafio é o fato de algoritmos sequenciais não serem otimizados o suficiente para lidar com dados em grande quantidade. Por causa disso, computadores de alta performance, com múltiplos *cores*, sistemas na nuvem e algoritmos paralelos e distribuídos são usados para lidar com esses empecilhos de *Big Data* (BELCASTRO et al., 2022).

Big Data refere-se a grandes conglomerados de dados complexos sobre os quais não é possível aplicar ferramentas tradicionais de processamento, armazenamento ou análise (KHALEEL; AL-RAWESHIDY, 2018). Estima-se que em 2025. os dados atuais criados, capturados ou replicados atinjam 175 Zettabytes, ou seja 175,000,000,000 Gigabytes (RYDNING, 2018).

A fim de lidar com essa enorme quantidade de dados, foi desenvolvido pelo Google o *MapReduce*, que é um modelo com uma implementação associada feito para processar e gerar grandes conglomerados de dados. Esse modelo é inspirado nos conceitos de mapear e reduzir, ou seja, aplicar uma operação que conecta cada item da base de dados a um determinado par de chaves e valores e então aplicar uma operação de reduzir, que junta os valores que compartilham chaves (DEAN; GHEMAWAT, 2008). Com essas operações é possível paralelizar dados em grandes quantidades e utilizar mecanismos de reutilização para facilitar a busca e manipulação destes.

Um dos *frameworks* mais populares que utiliza o *MapReduce* é o *Hadoop*, que foi desenvolvido pela Apache em 2006 e é capaz de armazenar e processar de giga a petabytes de dados eficientemente. Essa ferramenta é capaz de fazer isso optando por usar múltiplos computadores (*clusters*) em paralelo (WHITE, 2015).

1.2 OBJETIVO

O *Hadoop MapReduce* é um *framework* extremamente personalizável e adaptável. Por causa disso, frequentemente usa-se o processo de *tuning*, que consiste em modificar os mais de 190 parâmetros desse *framework* de modo a maximizar a eficiência de um *cluster Hadoop*. Esses parâmetros podem ser alterados em diversas combinações e podem ter efeitos tanto no *cluster* quantos nas tarefas (*jobs*) do processo.

Esse trabalho tem como objetivo avaliar o comportamento do *Hadoop MapReduce* antes e depois do *tuning* de alguns parâmetros de configuração, observando através de métricas de *benchmark* se houve melhora na performance considerando medidas como tempo e uso de memória.

1.3 ESTRUTURA DO TRABALHO

[TODO: ESTRTURA DO TRABALHO]

2 REFERENCIAL TEÓRICO

Esse capítulo tem como objetivo apresentar detalhadamente os conceitos técnicos que serão utilizados ao longo do trabalho. A seção 2.3 apresenta o *MapReduce*, o modelo de manipulação de dados feito pelo Google e a seção 2.4 trata do *Hadoop*, o *framework* desenvolvido pela Apache. A seção 2.5 introduz o *Hadoop MapReduce*.

2.1 CLUSTERS

Um *cluster* é um conjunto de computadores que trabalham juntos paralelamente em uma determinada aplicação. Cada computador desse conjunto é usualmente chamado de nodo. Além disso, existem diversas categorias de clusters dependendo do problema que eles buscam computar.

Algumas aplicações comuns de *clusters* são modelagem de clima, simulação de acidentes automotivos, mineração de dados e aplicações da área de astrofísica. Além disso, é comumente visto em aplicações comerciais como bancos e serviços de email (SADASHIV; KUMAR, 2011).

Uma das maiores vantagens desse tipo de instalação é a tolerância de falhas, pois os sistemas conseguem continuar suas tarefas caso um nodo pare de funcionar. Ainda, é altamente escalável com a adição de novos nodos, não precisa de manutenção frequente e tem um gerenciamento centralizado. Por fim, uma das suas maiores possíveis vantagens é o balanceamento de carga, que busca atingir o equilíbrio entre as tarefas de cada nodo de modo a otimizar os recursos.

2.2 CONTAINERS

[TODO: CONTAINERS]

2.3 MAPREDUCE

MapReduce é um modelo de programação associado a uma implementação que tem como objetivo processar, manipular e gerar grandes *datasets* de modo eficiente, escalável e com aplicações no mundo real. As computações acontecem de acordo com funções de mapeamento e redução e o sistema do *MapReduce* paraleliza essa computações entre grandes *clusters*, lidando com possíveis falhas, escalonamentos e uso eficiente de rede e discos (DEAN; GHEMAWAT, 2008).

As operações de mapeamento e redução são baseadas em conceitos presentes em linguagens funcionais e fazem com que seja possível fazer diversas reutilizações, assim lidando

com tolerância de falhas (DEAN; GHEMAWAT, 2008).

2.3.1 Terminologia

Para entender o *MapReduce*, antes é necessário que sejam estabelecidades algumas terminologias. Um *job* é uma unidade do que será processado: consiste nos dados de entrada, o programa *MapReduce* em si e as informações de configuração. O *Hadoop* executa essas unidade separando-a em tarefas (*tasks*), que podem ser do tipo de mapeamento (*map*) ou redção (*reduce*). O escalonamento dessas tarefas é feito automaticamento e cada uma roda em um nodo do *cluster*. Com existe a tolerância de falhas já implementada, se uma tarefa falha, ela é reescalonada. (WHITE, 2015)

2.3.2 Modelo de programação

A computação recebe um conjunto de pares (VALOR, CHAVE) e produz um conjunto de pares de (VALOR, CHAVE). O usuário cria as funções *Map* e *Reduce* de acordo com seu caso de uso. *Map* recebe um único par (VALOR, CHAVE) e produz um conjunto intermediário de pares. Em seguida, a biblioteca *MapReduce* agrupa os valores com a mesma chave e esses valores servirão de entrada para a função *Reduce*. A função *Reduce* então junta os valores com a mesma chave de modo a criar um conjunto menor, sendo possível dessa forma lidar com listas muito grandes para a memória disponível (DEAN; GHEMAWAT, 2008).

Como exemplo, considere o problem de contar quantas vezes determinada palavra aparece em um documento. Nesse problema, as funções *Map* e *Reduce* seriam similares aos seguintes pseudocódigos (DEAN; GHEMAWAT, 2008):

```
1 map(String chave, String valor):
2 // chave: nome do documento
3 // valor: conteudo do documento
4
5   para cada palavra W em valor:
6     criaIntermediario(W, 1);
```

CÓDIGO 2.1 – Exemplo de função Map em pseudocódigo adaptado de (DEAN; GHEMAWAT, 2008)

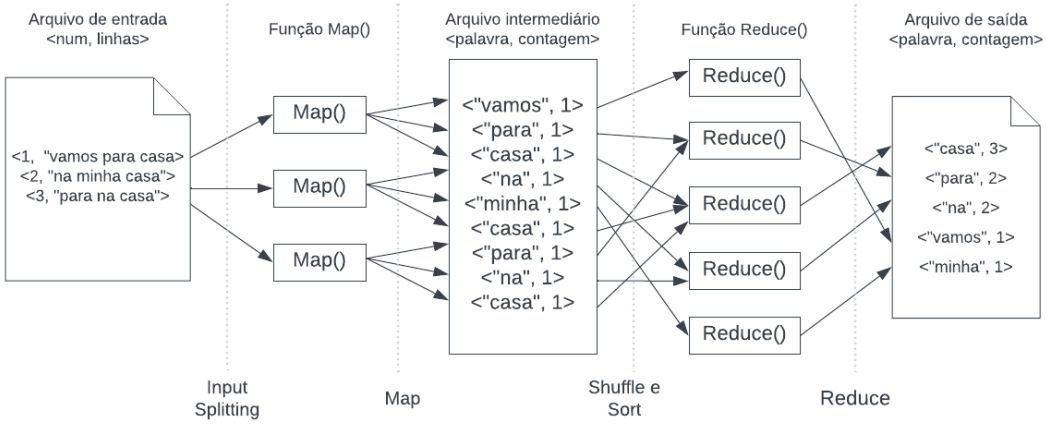
```
reduce(String chave, Iterador valores):
// chave: uma palavra
// valores: lista de contagens

  int resultado = 0;
  para cada V em valores:
    resultado = resultado + 1;
  cria(resultado);
```

CÓDIGO 2.2 – Exemplo de função Reduce em pseudocódigo adaptado de (DEAN; GHEMAWAT, 2008)

A função *Map* gera um objeto intermediário de cada palavra associada a uma lista do seu número de occorrências no texto e a função *Reduce* soma os valores até ter o total de ocorrências por palavra. Além disso, o usuário cria um configuração de *MapReduce* com os parâmetros de entrada e saída e eventuais parâmetros de *tuning*.

Para exemplificar ainda mais, considere um arquivo de texto com três linhas nas quais estão as seguintes frases, respectivamente, uma em cada linha: "vamos para casa", "na minha casa", "para na casa". Nesse exemplo, a função *Map* é chamada três vezes, uma para cada linha, gerando os pares (CHAVE, VALOR) intermediários, um para cada palavra encontrada no texto, como é exemplificado na FIGURA 1. Para cada palavra distinta ("vamos", "para", "casa", "na", "minha"), é executada a função *Reduce*, que soma quantas vezes cada uma dessas palavras apareceu no texto e gera um arquivo de saída.

FIGURA 1 – EXEMPLO DE EXECUÇÃO DO MAPREDUCE



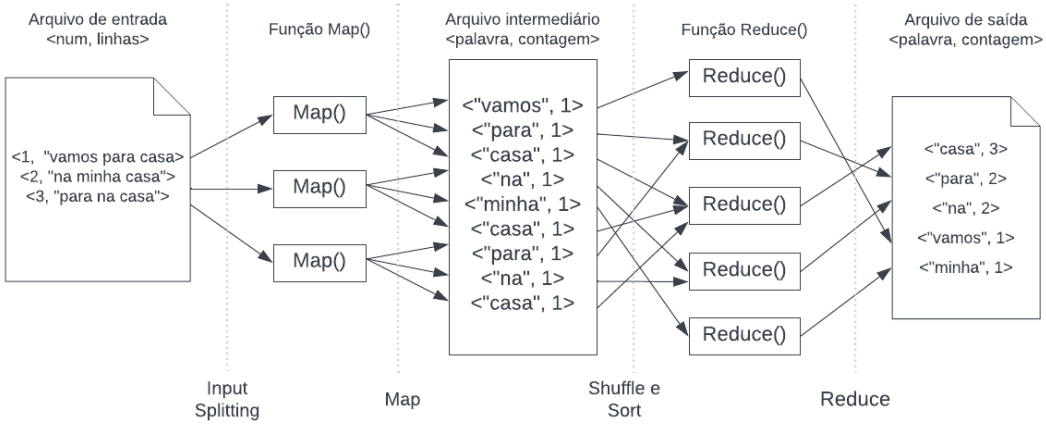
FONTE: A autora (2022)

2.3.3 Execução do MapReduce

[TODO: AQUI COLOCAR ESTRUTURA CLIENTE SERVIDOR MASTER WORKER]

Na FIGURA 1 foi possível ver como o *MapReduce* funcionaria em pequena escala. Uma das maiores vantagens do *MapReduce* é, no entanto, sua escalabilidade visto que ele permite uma execução distribuída entre uma grande quantidade de nodos. A FIGURA 2 representa uma execução genérica do *MapReduce*, na qual os dados de entrada são separados e para cada uma dessas separações o nodo *master* designa um *worker map*. Cada *worker* processa sua entrada e cria um objeto intermediário. Depois, o *master* utiliza esses objetos intermediários como entrada de um *worker reduce* e estes geram os arquivos finais de saída.

FIGURA 2 – EXEMPLO DE EXECUÇÃO DO MAPREDUCE



FONTE: A autora (2022)

2.4 HADOOP

Hadoop é um *framework* desenvolvido na linguagem Java pela Apache com os seguintes princípios arquiteturais, segundo Navarro Belmonte (2018):

- A possibilidade de escalar o sistema ao adicionar nodos no *cluster*.
- Possibilidade de funcionar bem em *hardware* não necessariamente caro e de luxo.
- Tolerância a falhas, com implementações que identificam estas e permitem que o sistema funciona independente delas acontecerem.
- Fornecer serviçoes transparentes de modo que o usuário possa focar no problema que ele quer resolver.

Esse *framework* disponibiliza ferramentas para que o usuário possa escrever as funções necessárias em diversas linguagens de programação, conforme a necessidade do programador. O *framework* funciona na mesma estrutura de Cliente/Servidor apresentada anteriormente e que é usada pelo *MapReduce*. Além disso, oferece ao programador um sistema complexo paralelo e distribuído (*Hadoop HDFS*), com os recursos ocultos do usuário, mas capaz de lidar com a

comunicação entre as máquinas, quaisquer falhas que possam vir a ocorrer e o escalonamento das tarefas.

2.4.1 Arquitetura do Hadoop

2.4.2 Hadoop HDFS

2.5 HADOOP MAPREDUCE

2.5.1 Componentes do Hadoop MapReduce

2.5.2 Fluxo de Execução

2.6 DOCKER

REFERÊNCIAS

BELCASTRO, L.; CANTINI, R.; MAROZZO, F.; ORSINO, A.; TALIA, D.; TRUNFIO, P. Programming big data analysis: principles and solutions. **Journal of Big Data**, SpringerOpen, v. 9, n. 1, p. 1–50, 2022. Citado 2 vez na página 8.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. **Communications of the ACM**, ACM New York, NY, USA, v. 51, n. 1, p. 107–113, 2008. Citado 9 vezes nas páginas 8, 10–12.

KHALEEL, A.; AL-RAWESHIDY, H. Optimization of computing and networking resources of a Hadoop cluster based on software defined network. **IEEE Access**, IEEE, v. 6, p. 61351–61365, 2018. Citado 1 vez na página 8.

KUO, Y.-H.; KUSIAK, A. From data to big data in production research: the past and future trends. **International Journal of Production Research**, Taylor & Francis, v. 57, n. 15-16, p. 4828–4853, 2019. DOI: 10.1080/00207543.2018.1443230. eprint: https://doi.org/10.1080/00207543.2018.1443230. Disponível em: https://doi.org/10.1080/00207543.2018.1443230. Citado 1 vez na página 8.

NAVARRO BELMONTE, V. P. **Improving Real Time Tuning on YARN**. 2018. Tese (Doutorado) – Carleton University. Citado 1 vez na página 13.

RYDNING, D. R.-J. G.-J. The digitization of the world from edge to core. **Framingham: International Data Corporation**, p. 16, 2018. Citado 1 vez na página 8.

SADASHIV, N.; KUMAR, S. D. Cluster, grid and cloud computing: A detailed comparison. In: IEEE. 2011 6th international conference on computer science & education (ICCSE). [S.l.: s.n.], 2011. P. 477–482. Citado 1 vez na página 10.

WHITE, T. **Hadoop: The definitive guide**. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado 2 vezes nas páginas 8, 11.