UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Desenvolvimento e Avaliação de Desempenho de um *Cluster Raspberry Pi* e *Apache Hadoop* em Aplicações *Big Data*

Dissertação de Mestrado

Antônio José Alves Neto

São Cristóvão – Sergipe

2023

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Antônio José Alves Neto

Desenvolvimento e Avaliação de Desempenho de um *Cluster Raspberry Pi* e *Apache Hadoop* em Aplicações *Big Data*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Edward David M. Ordonez
Coorientador(a): Prof. Dr. José Aprígio Carneiro Neto

São Cristóvão – Sergipe

2023

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Alves Neto, Antônio José
A474p Desenvolvimento e avaliação de desempenho de um *cluster raspberry pi* e *apache hadoop* em aplicações *big data* / Antônio José Alves Neto ; orientador Edward David M. Ordonez. - São Cristóvão, 2023.
109 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2023.

1. Plataforma aberta da Web. 2. Benchmarking (Administração). 3. Big data. 4. Cluster (Sistema de computador) 5. Raspberry pi (Computador) 6. Zabbix (Software) I. Ordonez, Edward David M. orient. II. Título.

CDU 004

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Antônio José Alves Neto

Em 20 dias do mês de abril do ano de dois mil e vinte três, com início às 10h00min, realizou-se no Auditório do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Antônio José Alves Neto**, que desenvolveu o trabalho intitulado: ***" Desenvolvimento e Avaliação de Desempenho de um Cluster Raspberry Pi e Apache Hadoop em Aplicações Big Data",*** sob a orientação do Prof. Dr. **Edward David Moreno Ordonez**. A Sessão foi presidida pelo Prof. Dr. **Edward David Moreno Ordonez** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Victor Alves** (UMINHO) e, em seguida, o Prof. Dr. **Methanias Colaço Rodrigues Júnior** (PROCC/UFS) e o Prof. Dr. **Jose Aprigio Carneiro Neto** (IFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

gov.br Documento assinado digitalmente
EDWARD DAVID MORENO ORDONEZ
Data: 20/04/2023 12:23:27-0300
Verifique em https://validar.iti.gov.br

Cidade Universit gov.br Documento assinado digitalmente
METHANIAS COLACO RODRIGUES JUNIOR
Data: 27/04/2023 11:41:41-0300
Verifique em https://validar.iti.gov.br
", 20 de abril de 2023.

Prof. Dr. Edward David Moreno Ordonez
(PROCC/UFS)
Presidente

Prof. Dr. Methanias Colaço Rodrigues Júnior
(PROCC/UFS)
Examinador Interno

gov.br Documento assinado digitalmente
JOSE APRIGIO CARNEIRO NETO
Data: 20/04/2023 12:37:19-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Victor Manuel Rodrigues Alves
(UMinho)
Examinador Externo

Prof. Dr. José Aprigio Carneiro Neto
(IFS)
Coorientador

Assinado por: **Víctor Manuel Rodrigues Alves**
Num. de Identificação: 05910026

Antônio José Alves Neto
Candidato

gov.br Documento assinado digitalmente
ANTONIO JOSE ALVES NETO
Data: 27/04/2023 15:02:06-0300
Verifique em https://validar.iti.gov.br

Este trabalho é dedicado aos meus pais José Augusto e Vagna Erlir.

Agradecimentos

Aos meus pais José Augusto e Vagna Erlir por investirem tanto na minha educação, a meus irmãos Gustavo e Juliana por todo suporte que é me dado diariamente além dos meus familiares que longe ou perto me ajudam sempre que preciso.

Aos meus orientadores Dr. Edward Moreno e Dr. Aprígio Carneiro por toda paciência e parceria durante toda essa dura e árdua caminhada e ao Instituto Federal de Sergipe (IFS) - Itabaiana, por cederem os equipamentos utilizados neste trabalho.

A todos os meus professores do PROCC que me deram aula durante todo esse tempo, seja no período como aluno especial, quanto como aluno regular e a Elaine por sempre sanar qualquer dúvida referente ao mestrado.

Aos meus superiores Max Castor e Andrea Macedo que me incentivaram e possibilitaram conciliar, em muitas ocasiões, as atividades laborais com as atividade acadêmicas, além dos meu colegas de trabalho Yúri Faro, José Marcelo e Alysson Moura por toda ajuda quando se fez necessária.

Aos “Omilhados” que desde 2007 se fazem presente como bons amigos devem ser.

E a todos que me ajudaram diretamente ou indiretamente a finalizar mais uma etapa importante da minha vida.

“Dream on, dream on, dream on. Dream until your dreams come true.”
(Dream On - Aerosmith - 1973)
“So, understand. Don’t waste your time always searching for those wasted years.
Face up, make your stand. Realize you’re living in the golden years.”
(Wasted Years - Iron Maiden - 1986)
“Que sonhe por uma boa tarde, agora sonhe com os pés no chão.”
(Autor Desconhecido)

Resumo

Atualmente, com o exponencial avanço da tecnologia, uma grande quantidade dados é gerada diariamente. Dados esses que não são gerados apenas por pessoas. Uma gama de equipamentos eletrônicos também tornaram-se grandes geradores, dos quais esses grandes volumes de dados são conhecidos como *Big Data* e produzem informações valiosas e úteis para *business intelligence*, previsão, suporte à decisão, dentre outras possibilidades. Entretanto, o processamento desse grande volume de dados requer uma abordagem computacional diferente da tradicional, chamada de Computação de Alta Performance (*High Perfomance Computing* - HPC). Ao longo dos anos, a HPC vem sendo obtida graças à utilização de supercomputadores ou através de *clusters* computacionais. O primeiro deixou de ser uma opção pelo seu alto custo e difícil de manutenção, deixando a "clusterização" como a alternativa ideal. Os *clusters* são sistemas fracamente acoplados, formados por um conjunto de computadores que trabalham em colaboração uns com os outros, usando bibliotecas de troca de mensagens. Além disso, os *clusters* formados por Computadores de Placa Única (*Single Board Computer* - SBC) são uma alternativa viável para o desenvolvimento de pesquisas nessa área. Dentre os computadores de placa única, destaca se a *Raspberry Pi*, um SBC desenvolvido inicialmente para promover o ensino da ciência da computação. Sua variedade de modelos permite atender a diversas necessidades específicas e não requer grandes investimentos. Para operacionalização e processamento desse grande volume de dados em um *cluster*, faz-se necessário a instalação de uma plataforma de *big data*, sendo o *Apache Hadoop* uma das mais difundidas disponíveis atualmente. Desta forma, uma boa solução para se obter um *cluster big data* de baixo custo é utilizar a *Raspberry Pi* como estrutura de hardware e o *Apache Hadoop* como plataforma *Big Data*. No entanto, a falta de um material detalhado explicando todas as etapas da instalação, o processo de configuração e, por fim, a certificação de que o *cluster Hadoop* está funcionando corretamente é um problema pouco explorado pela comunidade acadêmica. Além disso, o monitoramento de recursos do *cluster* também é um problema que é pouco abordado pela academia. Partindo dessa problemática, este trabalho tem como objetivo, o desenvolvimento e avaliação de desempenho de um *cluster big data* de baixo custo utilizando *Raspberry Pi*, como estrutura de hardware de baixo custo e o *Apache Hadoop* como plataforma de *Big Data*. A avaliação do mesmo será feita utilizando *benchmarks* difundidos na área (*Terasort* e *TestDFSIO*), além de acompanhar e monitorar o uso dos seus recursos utilizando as ferramentas *Zabbix* e *Grafana*, provendo um material completo e detalhado de todo esse processo.

Palavras-chave: *Apache Hadoop*, *Benchmarks*, *Big Data*, *Cluster*, *Grafana*, *Raspberry Pi*, *Zabbix*

Abstract

Currently, with the exponential advancement of technology, a large amount of data is generated daily. These data aren't generated just by people. A range of electronic equipment has also become great generators. These large volumes of data are known as Big Data and produce valuable and helpful information for business intelligence, forecasting, and decision support, among other possibilities. However, processing this large volume of data requires a different computational approach from the traditional one, called High Performance Computing (HPC). Over the years, the HPC has been using supercomputers or computing clusters. The first one is no longer an option due to its high cost and difficulty to maintain, making clustering an ideal alternative. Clusters are loosely coupled systems, formed by a set of computers that work in collaboration with each other, using message exchange libraries. In addition, clusters formed by Single Board Computers (SBC) are a viable alternative for the development of research in this area. Among the SBCs, the Raspberry Pi stands out, a SBC initially developed to promote the teaching of computer science. Its variety of models allows it to meet several specific requirements and does not require large investments. To operate and to process this large volume of data in a cluster, it is necessary to have a big data platform, the Apache Hadoop being one of the most widely available today. Thus, a good solution to obtain a low-cost big data cluster is to combine the use the Raspberry Pi as the hardware structure and Apache Hadoop as Big Data platform. However, the lack of detailed material explaining all the installation steps, the configuration process, and, finally, the certification that the Hadoop cluster is working correctly is a problem little explored by the academic community. In addition, the monitoring of cluster resources is also a problem that is rarely addressed by the academy. In order to solve this problem, this work aims to develop and evaluate the performance of a low-cost big data cluster using Raspberry Pi as a low-cost hardware structure and Apache Hadoop as a Big Data platform. Its evaluation will be done using benchmarks widespread in the area (Terasort and TestDFSIO), in addition to accompanying and monitoring the use of its resources using the tools Zabbix and Grafana, providing a complete and detailed material of this entire process.

Keywords: Apache Hadoop, Benchmarks, Big Data, Cluster, Grafana, Raspberry Pi, Zabbix.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CE	Critérios de Exclusão
CI	Critérios de Inclusão
CPU	*Central Process Unit*
exFAT	*Extended File Allocation Table*
FAT	*File Allocation Table*
GB	*Gigabyte*
HA	*High Availability*
HC	*High Capacity*
HDFS	*Hadoop Distributed System File*
HDMI	*High-Definition Multimedia Interface*
HPC	*High Perfomance Computing*
I/O	*Input/Output*
JDK	*Java Development Kit*
MB	*Megabyte*
MSL	Mapeamento Sistemático da Literatura
OS	Operational System
PROCC	Programa de Pós-Graduação em Ciência da Computação
QP	Questão de Pesquisa
QRSL	Quasi-Revisão Sistemática da Literatura
RAM	*Random Access Memory*
SBC	*Single Board Computer*
SoC	*System on Chips*
SEFAZ/SE	Secretaria de Estado da Fazenda do Estado de Sergipe
UFS	*Universidade Federal de Sergipe*

UTP	*Unshielded Twisted Pair*
XC	*Extended Capacity*
YARN	*Yet Another Resource Negotiator*

Sumário

1 Introdução

Este capítulo tem como objetivo introduzir o presente trabalho no contexto de *clusters big data* de baixo custo. Na Seção 1.1 é dada primeiramente uma contextualização sobre *big data* e o uso de computação de alta *performance* para o processamento dessa grande quantidade de dados. Ainda nesta seção são apresentadas as dificuldades em se obter computadores com grande poder computacional e as possíveis soluções para contornar essa problemática, caracterizando a motivação e justificativa para o desenvolvimento deste trabalho. A Seção 1.2 apresenta o objetivo geral e objetivos específicos. A metodologia, explicando como o trabalho foi desenvolvido, é apresentada na Seção 1.3. Por fim, a Seção 1.4 discorre sobre a organização do trabalho, explicando como o mesmo está dividido e dando uma sinopse de cada capítulo.

1.1 Contextualização

Atualmente, com o exponencial avanço da tecnologia, uma grande quantidade de dados é gerada diariamente. Dados esses que não são produzidos apenas por pessoas, mas também uma gama de equipamentos eletrônicos tornaram-se grandes geradores, como: registros de *logs* de servidores, sensores, tecnologias *wearable*, dentre outras variedades de aplicações (RAYALA; KALLI, 2021). Mensurar o volume de todos estes registros eletrônicos não é uma tarefa fácil. Por exemplo, o Walmart tem aproximadamente um volume de dados de 2.5 *petabytes*, e a Meta armazena, aproximadamente, 30 *petabytes* de dados (BAROLLI, 2021), dos quais esses grandes volumes de dados são conhecidos como *Big Data* e produzem informações valiosas e úteis para *business intelligence*, previsão, suporte à decisão, dentre outras possibilidades.

A Internet de alta velocidade generalizada exacerbou a produção desses dados, impulsionando plataformas de processamento de *big data* eficientes em uma "era de *big data*". Na última década, *big data* tem sido uma palavra da moda e os vários avanços das tecnologias de *big data* tiveram um impacto crucial na vida diária das pessoas, bem como em vários setores da indústria

(KANITHAN et al., 2021). Sendo assim, o interesse por *Big Data* pode ser visto na busca pelo termo no Google[1] (Figura 1).

Figura 1 – Histórico de busca pelo termo "*Big Data*" no Google

O Gartner (2011) define *big data* como ativos de informações de alto volume, alta velocidade e/ou alta variedade que exigem formas econômicas e inovadoras de processamento de informações que permitem *insights* aprimorados, tomada de decisões e automação de processos. Ao longo dos anos, outras características foram adicionadas ao conceito: Valor, obtido na tomada de decisão com base em análise de dados (GANTZ; REINSEL, 2011), e Veracidade, garantindo a confiabilidade da informação (IBM, 2016), formando, assim, os 5 Vs (Volume, Velocidade, Variedade, Valor e Veracidade) do *Big Data*.

Dentro desse contexto, se puderem ser analisados, esses dados são úteis, vistos hoje como uma oportunidade de negócios, que podem oferecer a tomada de decisões com base na análise de dados, visando extrair informações pertinentes e conhecimento competitivo. Assim, desbloquear o valor do *Big Data* em mercados complexos e em rápida mudança pode trazer vantagem competitiva e permitir uma melhor resposta das empresas. Entretanto, o processamento desse grande volume de dados requer uma abordagem computacional diferente da tradicional, chamada de Computação de Alta Performance (*High Performance Computing* - HPC).

A HPC refere-se à prática de agregar poder de computação de uma forma que oferece potência muito maior do que os computadores e servidores tradicionais. A HPC nada mais é do que a computação cotidiana, só que mais poderosa, visto que é uma forma de processar grandes volumes de dados em velocidades muito altas usando vários computadores e dispositivos de armazenamento como uma estrutura única e coesa (ORACLE, 2023).

Ao longo dos anos, a HPC vem sendo obtida graças a utilização de supercomputadores ou através de *clusters* computacionais. O primeiro está sendo menos optado, principalmente pelo seu alto custo e sua difícil manutenção, deixando a "clusterização" como a alternativa perfeita para o processamento desse grande volume de dados (BOURHNANE et al., 2020).

[1] <https://trends.google.com.br/trends/explore?date=all&q=big%20data&hl=pt>

Os *clusters* são sistemas fracamente acoplados, formados por um conjunto de computadores que trabalham em colaboração uns com os outros, utilizando bibliotecas de troca de mensagens. As empresas têm desenvolvido *clusters* poderosos para explorar *Big Data*, tornando-se o carro-chefe dessas indústrias para esse grande processamento de dados. No entanto, por conta dos altos custos e do seu alto consumo energético, nem todas as empresas e pesquisadores do âmbito acadêmico necessitam ou podem pagar por tais servidores poderosos (LEE; OH; PARK, 2021).

Além disso, os *clusters* formados por Computadores de Placa Única (*Single Board Computer* - SBC) são uma alternativa viável para o desenvolvimento de pesquisas, sendo uma solução escalonável, flexível, de alta disponibilidade e baixo custo, que permite o balanceamento de carga e é tolerante a falhas (HENNESSY; PATTERSON, 2011).

O SBC é um tipo de computador que é implementado em uma única placa, a qual utiliza as capacidades da tecnologia sistemas em *chips* (*System on Chips - SOC*) em sua modelagem, permitindo o armazenamento de dados em um dispositivo externo, geralmente um cartão de memória, além de oferecer diversas interfaces para interação com outros dispositivos eletrônicos (KUSS et al., 2018). Nessa perspectiva, a principal ideia de uso de um SBC era auxiliar no aprendizado da computação, porém, por conta do seu poder computacional semelhante a um computador tradicional, tornou-se uma das principais referências no desenvolvimento de protótipos de ferramentas.

Dentre os SBCs existentes, destaca-se a *Raspberry Pi*, desenvolvida inicialmente para promover o ensino da ciência da computação. Além disso, sua variedade de modelos permite atender às diversas necessidades específicas (GIGER; SRIKUGAN; PERSAUD, 2020). O desenvolvimento de um *cluster* formado por este tipo de equipamento não requer grandes investimentos, além de permitir o uso de diversas bibliotecas de paralelismo e *big data* (QURESHI; KOUBAA, 2020).

Para a operacionalização e processamento desse grande volume de dados em um *cluster*, faz-se necessário a instalação de uma plataforma de *big data*, sendo o *Apache Hadoop* uma das mais difundidas dentre as disponíveis atualmente, visto que permite o processamento distribuído de grandes volumesd de dados em *clusters*, usando modelos de programação simples. Ademais, ele foi projetado para escalar de servidores únicos para milhares de máquinas, cada uma oferecendo computação e armazenamento locais. Em vez de depender do hardware para fornecer alta disponibilidade, a própria biblioteca foi projetada para detectar e lidar com falhas na camada de aplicação, oferecendo um serviço de alta disponibilidade em um *cluster*, cada um dos quais pode estar propenso a falhas (Apache Software Foundation, 2023).

Vários autores (HAJJI; TSO, 2016; LEE; OH; PARK, 2021; BöTHER; RABL, 2021; NETO; NETO; ORDONEZ, 2022; NETO; NETO; MORENO, 2022; GIGER; SRIKUGAN; PERSAUD, 2020; SRINIVASAN et al., 2018) vêm convergindo para ideia de que uma boa solução para se obter um *cluster big data* de baixo custo é utilizar a *Raspberry Pi* como estrutura

de hardware, e o *Apache Hadoop* como plataforma *Big Data*.

No entanto, a falta de um material detalhado explicando todas as etapas da instalação, o processo de configuração e, por fim, a certificação de que o *cluster Hadoop* está funcionando corretamente é um problema pouco explorado pela comunidade acadêmica. Além disso, o monitoramento de recursos do *cluster* também é um problema que é pouco abordado pela academia. Partindo dessa problemática, surgiu a motivação para o desenvolvimento desta dissertação de mestrado.

1.2 Objetivos

Esta seção apresenta o objetivo geral deste trabalho e os seus respectivos objetivos específicos.

1.2.1 Objetivo Geral

Em suma, este trabalho tem como objetivo geral, o desenvolvimento e avaliação de desempenho de um *cluster big data* de baixo custo utilizando *Raspberry Pi*, como estrutura de hardware de baixo custo, e o *Apache Hadoop* como plataforma de *big data*, apresentando um material completo e detalhado de todo esse processo. A avaliação do mesmo é feita utilizando *benchmarks* difundidos na área, além de acompanhar e monitorar o uso dos seus recursos computacionais durante o processo de *benchmarking* do *cluster*.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral deste trabalho, foram definidos os seguintes objetivos específicos (Figura 2):

- Desenvolvimento de um *cluster big data* de baixo custo, utilizando *Apache Hadoop*, como plataforma *Big Data*, e nove (9) *Raspberry Pis 4B*, como estrutura de hardware de baixo custo;
- Avaliação do *cluster* desenvolvido utilizando os *benchmarks Big Data Terasort* e *TestDFSIO*, amplamente utilizados pela comunidade de big data;
- Desenvolvimento de um servidor de monitoramento de recursos do *cluster* utilizando 1 *Raspberry Pi 3B+*, utilizando as ferramentas *Zabbix* e *Grafana*;
- Fornecer um material completo e detalhado do desenvolvimento, testagem e monitoramento do *cluster big data*;

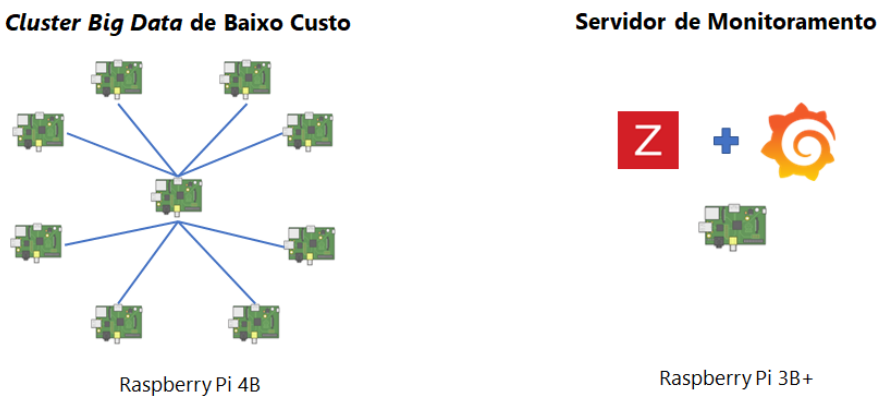

Figura 2 – Objetivo proposto deste trabalho.

1.3 Metodologia

A metodologia adotada no desenvolvimento deste trabalho deu-se inicialmente com o levantamento bibliográfico sobre *clusters big data* de baixo custo (Capítulo 3), visando identificar como conseguir o processamento de *big data* de forma mais simples e mais barata que as abordagens convencionais utilizadas na HPC.

Após a identificação que uma possível solução para o problema seria a combinação de *Raspberry Pi* com *Apache Hadoop*, foi feito um estudo qualitativo, através de uma Quasi-Revisão Sistemática da Literatura (Capítulo 4) para identificar como esses *clusters* estavam sendo desenvolvidos, testados e monitorados.

Em seguida, foi feita a aquisição dos dispositivos, sendo nove (9) *Raspberry Pis 4B* e iniciado o desenvolvimento do *cluster*. Em paralelo, foram encontradas e estudadas algumas soluções que pudessem validar o funcionamento do mesmo, sendo elencados os *benchmarks Terasort* e *TestDFSIO* para tal (Capítulo 4), tendo suas execuções sendo realizadas em diferentes configurações do *cluster*, mais especificamente, os testes foram realizados com o *cluster* contendo 2, 4 e 8 nós escravos e com diferentes tamanhos dos arquivos de dados (*datasets*), além de utilizar capacidades de armazenamentos diferentes, a fim de ser traçado um melhor comparativo entre os mesmos e garantir a possibilidade de escalabilidade do *cluster*.

Além disso, foi selecionada uma placa *Raspberry Pi 3B+* com o intuito de servir como servidor de monitoramentos dos recursos utilizados pelo *cluster* desenvolvido. Nessa placa foram instaladas as ferramentas *Zabbix* para a obtenção dos dados dos recursos do *cluster* (CPU, RAM, *I/O*, etc.), e o uso da ferramenta *Grafana*, para expor esses dados em forma de *dashboards*.

1.4 Organização do Trabalho

As demais partes deste trabalho estão organizadas em 8 capítulos da seguinte forma:

- Capítulo 2 - Enfatiza os conceitos básicos sobre os temas abordados neste trabalho;
- Capítulo 3 - Apresenta o Mapeamento Sistemático da Literatura (MSL) que serviu como base para o desenvolvimento deste trabalho, com o intuito de identificar a possibilidade de desenvolvimento de um *cluster big data* de baixo custo;
- Capítulo 4 - É apresentada a Quasi-Revisão Sistemática da Literatura (QRSL), desenvolvida com o objetivo de identificar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados, motivada como trabalho futuro do MSL, apresentado no Capítulo 3;
- Capítulo 5 - Oferece uma visão dos trabalhos encontrados, relacionados ao tema proposto neste trabalho e faz uma análise comparativa entre os trabalhos resultantes do levantamento bibliográfico e o trabalho proposto nesta dissertação de mestrado;
- Capítulo 6 - Detalha o processo de desenvolvimento do *cluster*, mostrando todos os recursos utilizados (softwares e hardwares), além de um passo a passo de instalação e configuração do mesmo e uma visão geral do *cluster* desenvolvido;
- Capítulo 7 - Mostra o processo de testagem e validação do *cluster*, utilizando os *benchmarks Terasort* e *TestDFSIO* (Capítulo 4), além do monitoramento dos recursos do *cluster* durante o processo de testagem;
- Capítulo 8 - São apresentadas as considerações finais deste trabalho, relatando o que se concluiu com o desenvolvimento do mesmo, além de mostrar as contribuições alcançadas, as limitações encontradas e as ideias para possíveis trabalhos futuros.

2

Referencial Teórico

Este capítulo tem como objetivo enfatizar os conceitos importantes que foram citados neste trabalho de forma sucinta, a fim de fornecer conhecimentos prévios aos leitores sobre as temáticas que foram abordadas neste estudo.

2.1 *Big Data*

A definição de "*Big Data*" é complexa e está em constante mudança. Entretanto, existe um consenso na literatura no que diz respeito às principais características de *Big Data*, como é descrita por um relatório amplamente citado do Gartner (2011), que diz o seguinte: "*Big Data* é o alto volume, alta velocidade e/ou alta variedade de ativos de informação que exigem formas inovadoras e econômicas de processamento que permitem uma visão aprimorada, tomada de decisão e automação de processos".

De forma similar, Gantz e Reinsel (2011) definiram *Big Data* usando 4V's (Volume, Variedade, Velocidade e Valor) em 2011. Já em 2012, a característica "Veracidade" foi adicionada como a 5ª característica que compõem o *Big Data* (IBM, 2016). Essas características são ilustradas na Figura 3.

O "Volume" se refere ao grande volume de dados que é gerado diariamente ao redor do mundo, oriundo das diferentes fontes como áudio, vídeos, mensagens, redes sociais, etc. (KHAN; UDDIN; GUPTA, 2014). A "Velocidade" é o quão rápido esses dados são gerados, armazenados e analisados para permitir uma melhor tomada de decisão. O quão rápido esses dados estão sendo gerados pode ser visto no site *Internet Live Stats*[1]. A "Variedade", assim como no volume, a variedade dos dados é muito grande. Esses dados estão estruturados em diferentes formas: em arquivos sequenciais, bancos de dados e em dados não estruturados, que são os conteúdos provenientes de diferentes tipos de fontes. O "Valor" é uma das mais importantes características

[1] <https://www.internetlivestats.com/>

Figura 3 – Os 5V's do *Big Data*. Adaptado de Trickdoid (2021).

do *Big Data* e está relacionada ao valor agregado que é obtido através das tomadas de decisões que foram baseadas em análises dos dados e feitas de forma prévia pelas organizações (KHAN; UDDIN; GUPTA, 2014). Por fim, a "Veracidade" lida com o fato de que com o alto volume, a alta velocidade e alta variedade dos dados, não é possível garantir 100% de informações corretas. Assim, a qualidade dos dados pode variar. A precisão da análise dos dados depende da veracidade dos dados de origem (OJOKOH et al., 2020).

Visto no passado como um problema técnico, o *Big Data* é hoje visto como uma oportunidade de negócios que pode oferecer a tomada de decisões com base na análise de dados. O principal desafio do *Big Data* é explorar grandes dados com o objetivo de extrair informações úteis e conhecimento competitivo. Dessa forma, desbloquear o valor do *Big Data* em mercados complexos e em rápida mudança pode trazer vantagem competitiva e permitir uma melhor resposta das empresas.

2.2 *Apache Hadoop*

O software *Apache Hadoop* é um *framework* que permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores usando modelos de programação simples. Ele foi projetado para funcionar desde em servidores únicos até em milhares de máquinas, cada uma delas oferecendo computação e armazenamento local. Em vez de depender de hardware para fornecer alta disponibilidade (*High Availability* - HA), a própria biblioteca é projetada para detectar e lidar com falhas na camada de aplicação, oferecendo, assim, um serviço altamente disponível em um *cluster* de computadores, cada um desses computadores pode estar sujeito a falhas (Apache Software Foundation, 2023).

Criado em 2005 por Goug Cutting e Mike Cafarella para dar suporte a um projeto de busca distribuída, o *Apache Hadoop* é um *framework* Java, de código aberto (*open-source*), que auxilia no armazenamento, no acesso e em obter grandes recursos de *Big Data* de forma distribuída com menor custo, alto grau de tolerância a falhas e alta escalabilidade (SARALADEVI et al., 2015). O *Apache Hadoop* é uma tecnologia que "casa" com a necessidade de *Big data*, uma vez que é baseado no modelo de programação simples desenvolvido pelo *Google*, o *MapReduce*. O *Apache Hadoop* faz parte do projeto Apache, composto de alguns outros projetos importantes (Figura 4). Grandes empresas como *IBM*, *Intel* e *Oracle* também contam com uma extensão do *Apache Hadoop* para suas soluções de *Big Data* (SARALADEVI et al., 2015).

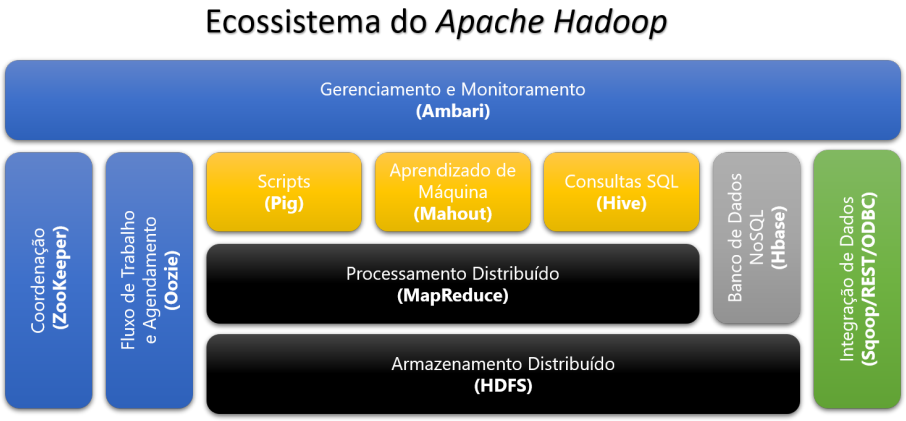

Figura 4 – Ecossistema do *Apache Hadoop*. Adaptado de (Apache Software Foundation, 2023).

A base do *framework Apache Hadoop* é composta pelos módulos que podem ser observados na Figura 5. Munde e Jahan (2007) descrevem cada parte do *framework* da seguinte forma:

- ***Hadoop Common Files*** - Contém as bibliotecas Java e utilitários, as quais fornecem sistemas de arquivos e abstrações no nível do sistema operacional. Além disso, também contém os arquivos Java e *scripts* necessários para iniciar o *Apache Hadoop*;
- ***Hadoop Distributed File System (HDFS)*** - É o sistema de arquivos distribuídos que armazena os dados em máquinas de *commodities*, fornecendo largura de banda muito alta em todo o *cluster*. Ele fornece alta tolerância a falhas e suporte nativo de grandes conjuntos de dados e os armazena em hardware comum;
- ***MapReduce*** - Implementação do modelo de programação *MapReduce* para o processamento de dados em larga escala, ele é usado para processamento de grande volume de dados de forma paralela;
- ***Yet Another Resource Negotiator (YARN)*** - Plataforma de gerenciamento de recursos que é responsável por gerenciar os recursos computacionais nos *clusters* e usá-los para agendar as tarefas dos usuários que serão executadas;

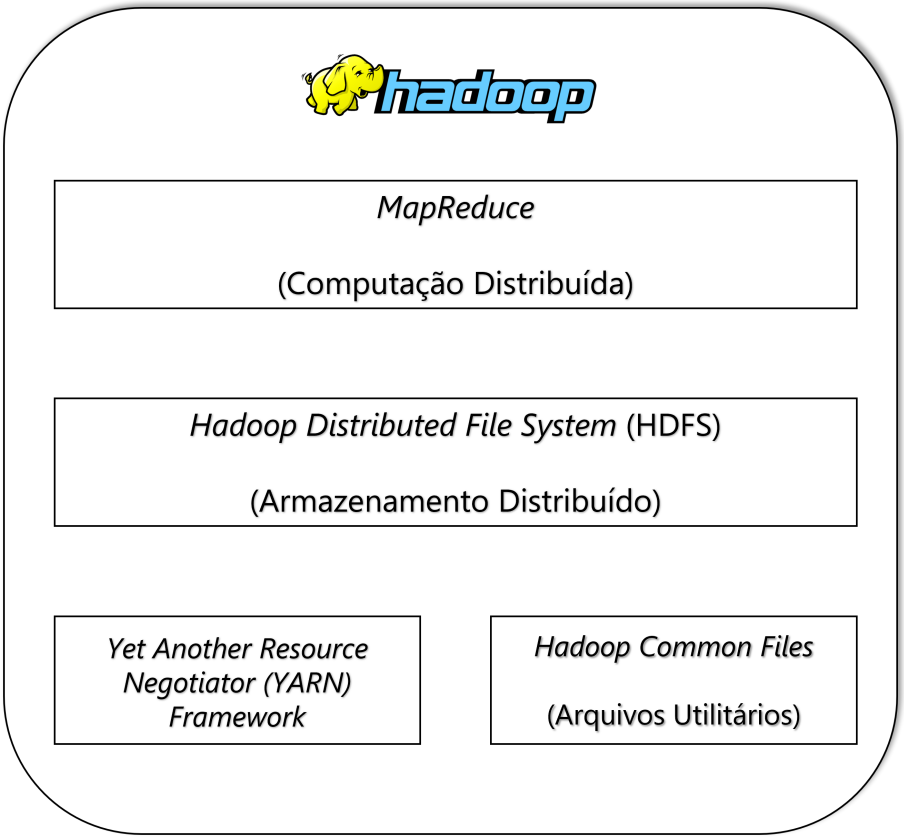

Figura 5 – Base do *framework Apache Hadoop* - Adaptado de (Apache Software Foundation, 2023).

2.3 *Big Data Benchmarks*

Um *benchmark* de *Big Data* visa gerar cargas de trabalho específicas e testes capazes de processar grandes conjuntos de dados para produzir resultados de avaliação significativa (TAY, 2011). Nesta Seção, são abordados os dois *benchmarks* que foram utilizados para testagem e avaliação do *cluster* desenvolvido: *Terasort* e *TestDFSIO*.

2.3.1 *Terasort Benchmark*

O *Terasort* é um dos *benchmarks* mais difundidos na comunidade *Big Data*. Ele mede o tempo para ordenar um número diferente de registros de 100 *bytes*, estimulando e estressando quase todas as partes do *framework MapReduce* do *Apache Hadoop*, assim como o seu sistema de arquivos distribuído (HDFS), tornando-o uma escolha ideal para ajustar a configuração de um *cluster Hadoop* (Apache Software Foundation, 2023).

Por exemplo: assumindo a necessidade de gerar dados de 1 GB, o número de linha de 100 *bytes* será 10^7, pois 1 GB = 10^7 * 100 *bytes* = 10^9 *bytes*.

Existem três etapas envolvidas no conjunto de execuções do *benchmark Terasort*. São elas:

- Gerar os dados de entrada via ***Teragen***;
- Executar o ***Terasort*** nos dados de entrada gerados pelo ***Teragen***;
- Validar os dados de entrada com os dados ordenados usando-se do ***Teravalidate***;

2.3.2 *TestDFSIO Benchmark*

O *TestDFSIO* é um *benchmark* utilizado para medir a taxa agregada de dados (*ratio* de um *cluster*) e a taxa de transferência de dados (*throughput*), comparando o seu desempenho durante os processos de escrita e leitura (*input and output* - I/O) em uma quantidade *N* de arquivos definidas no início da execução (Apache Software Foundation, 2023).

Seu processo de teste funciona da seguinte maneira: O programa de gravação inicia múltiplas tarefas de *Map*, onde cada trabalho é salvo em arquivos do HDFS de foma separada. O programa de leitura inicializa várias tarefas de *Map*, lendo os arquivos escritos no HDFS, com cada tarefa lida de forma sequencial. Em seguida, as medições das tarefas são realizadas.

O *TestDFSIO* utiliza uma única tarefa de *Reduce* para medir e calcular duas (2) métricas de performance para cada tarefa de *Map* executada: Taxa média de entrada e saída (I/O) e taxa de transferência (*throughput*). Desta forma, gargalos e problemas de desempenho na configuração da rede, hardware, sistema operacional e do próprio *Apache Hadoop* podem ser identificados e corrigidos.

2.4 *Single Board Computer* (SBC)

Single Board Computers - SCB ou Computadores de Placa Única são dispositivos ou um tipo de computador que é implementado em uma única placa impressa. Ele utiliza as capacidades da tecnologia sistemas em *chips* (*System on Chips - SoC*) em sua modelagem, permitindo também o armazenamento de dados em um dispositivo externo, geralmente um cartão de memória, além de oferecer várias interfaces para interagir com outros dispositivos eletrônicos (KUSS et al., 2018).

Os SBCs foram desenvolvidos com o intuito de serem uma ferramenta de apoio ao aprendizado de linguagens de programação, tornando-se uma das principais referências no desenvolvimento de protótipos de ferramentas que exigem um poder de processamento semelhante aos computadores tradicionais. Um dos principais e mais popular SBC é a *Raspberry Pi* (Seção 2.5).

2.5 *Raspberry Pi*

A *Raspberry Pi* é um SBC desenvolvido inicialmente para promover o ensino da ciência da computação. Por exemplo, o modelo “Zero” é ideal para pequenos projetos de robótica,

por causa do seu tamanho e o seu baixo custo, porém não tem entrada para *Ethernet* nem um processador poderoso. Já o modelo "4" é adequado para projetos mais sofisticados e vem com um processador *Quad-core Cortex-A72 (ARM v8)* de 64 *bits* e até 4 GB de *SDRAM* (GIGER; SRIKUGAN; PERSAUD, 2020).

Figura 6 – *Raspberry Pi* - Um tipo de SBC. Adaptado de Qureshi e Koubaa (2020).

O sistema operacional oficial é chamado *Raspberry Pi OS*, uma distribuição Linux baseada na distribuição do Debian e otimizada para *Raspberry Pis*. Além do *Raspberry Pi OS* com suporte oficial, existem vários sistemas operacionais de terceiros. Isso inclui, entre outros: *Ubuntu, OSMC, LibreElec, Mozilla WebThings, PiNet, RISC OS e Hypriot OS*. Em geral, os softwares e hardwares são compatíveis com versões anteriores, o que simplifica a atualização (GIGER; SRIKUGAN; PERSAUD, 2020).

O modelo "4B", lançado em Junho de 2019, demonstra uma ampla melhoria de desempenho em relação ao modelo anterior, devido ao seu redesenho completo de *chip*, sendo o primeiro na história da *Raspberry Pi*: núcleos de processamento mais poderosos, a primeira atualização do processador gráfico, memória e largura de banda de hardware externo amplamente melhoradas, incluindo as primeiras portas USB 3.0, conexão *Ethernet Gigabit*, entradas HDMI para monitores 4K, etc. (KANITHAN et al., 2021).

2.6 Computação de Alta Performance - *High Performance Computing* (HPC)

A computação de alta performance ou de alto desempenho, refere-se à prática de agregar poder computacional de uma forma que oferece potência muito maior do que os computadores e servidores tradicionais. A HPC ou supercomputação, é como a computação cotidiana, só que mais poderosa. É uma forma de processar grandes volumes de dados em velocidades muito altas usando vários computadores e dispositivos de armazenamento como uma estrutura coesa (ORACLE, 2023).

A utilização dessa variada quantidade de computadores ou supercomputadores é, na realidade, um *cluster* de computadores composto por vários processadores e, obviamente, o seu

custo, tal como o seu desempenho, é elevadíssimo (RAMOS, 2021).

Algumas cargas de trabalho, como sequenciamento de DNA, são simplesmente imensas para serem processadas por um único computador. Os ambientes de HPC ou de supercomputação abordam esses desafios, que são classificados como grandes e complexos, com nós individuais (computadores) trabalhando juntos em um *cluster* para realizar grandes quantidades de computação em um curto período de tempo. A HPC torna possível explorar e encontrar respostas para alguns dos maiores problemas mundiais em ciência, engenharia e negócios (ORACLE, 2023).

2.7 *Cluster*

Um *cluster* é um agrupamento de computadores conectados ou *hosts* que trabalham em conjunto para auxiliar aplicações e *middleware* (por exemplo, bancos de dados). Em um *cluster*, cada computador refere-se a um "nó". Ao contrário da computação em grade (*grid computing*), onde cada nó executa uma tarefa diferente, os *clusters* de computadores atribuem uma mesma tarefa a cada nó. Os nós em um *cluster*, geralmente estão conectados a uma rede local de alta velocidade e cada nó executa sua instância de um sistema operacional (VIRTANA, 2022).

Um *cluster* formado por computadores pode variar desde um simples sistema de dois nós conectando dois computadores pessoais, a um supercomputador com uma arquitetura de N computadores interconectados. Um exemplo de um *cluster* pode ser visto na Figura 7.

Os *clusters* são frequentemente usados para obter uma forma de computação de alto desempenho (HPC) e alta disponibilidade (*high availability* - HA) com uma boa relação de custo-benefício. Neste modelo, se um componente do *cluster* falhar, os demais podem continuam a prover o processamento de informações de forma ininterrupta. Pode fornecer velocidade de processamento mais rápida, maior capacidade de armazenamento, melhor integridade de dados, maior confiabilidade e maior disponibilidade de recursos. Geralmente são dedicados a funções específicas, como balanceamento de carga, alta disponibilidade, alto desempenho ou processamento de dados em larga escala (VIRTANA, 2022).

2.8 *Zabbix*

Zabbix é uma solução *open-source* de monitoramento distribuído, que possibilita monitorar diversos parâmetros de rede, saúde e integridade dos servidores, máquinas virtuais, aplicações, bancos de dados e etc. Ele utiliza um mecanismo flexível de notificação que permite aos usuários configurar alertas para, praticamente, qualquer tipo de evento, permitindo uma reação rápida a possíveis surgimento de problemas (Zabbix Company, 2023).

Por suportar tanto *polling* (coleta de dados), quanto *trapping* (envio de dados), o *Zabbix* é um software de fácil compatibilidade e integração com outras ferramentas, aumentando a

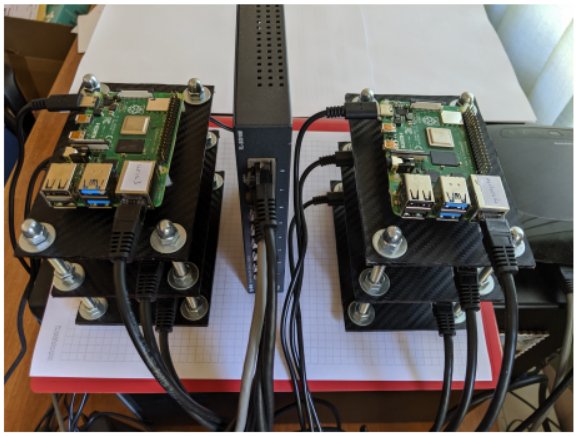
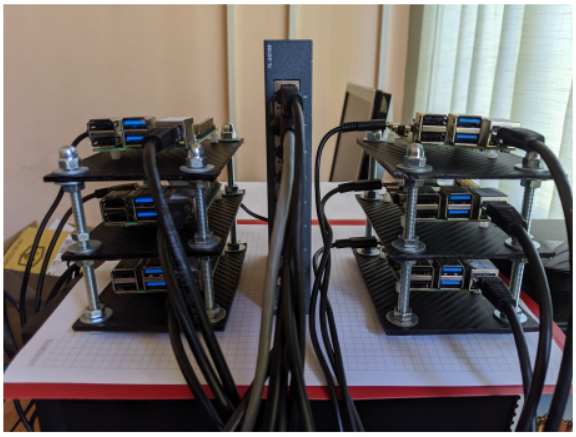

Figura 7 – *Cluster Raspberry Pi*. Adaptado de Komninos et al. (2019).

capacidade de monitoramento e alcance dos dispositivos. Os principais módulos do sistema de monitoramento *Zabbix* são: *Zabbix Server* e *Zabbix Agent*.

2.8.1 *Zabbix Server*

O *Zabbix Server* é o principal componente do *Zabbix*, e coleta dados dos agentes (*Zabbix Agent*) para o monitoramento. Quando alguma anormalidade é detectada, alertas são emitidos visualmente e através de uso de sistemas ou ferramentas de comunicação. Ele também mantém histórico dos dados coletados em banco de dados para a geração de gráficos, *dashboards*, que mostram as informações de forma alternada (Zabbix Company, 2023).

2.8.2 *Zabbix Agent*

O *Zabbix Agent* é instalado nos *hosts*, possibilitando coletar diversas métricas simples, como uso de CPU e memória RAM, temperatura, quantidade de processos em execução, etc. Além disso, o *Zabbix Agent* permite a coleta de métricas complexas ou personalizadas com o auxílio de *scripts* ou programas externos e até mesmo a tomada de ações diretamente no próprio *Zabbix Agent* (Zabbix Company, 2023).

2.9 *Grafana*

O *Grafana* é uma plataforma para visualizar e analisar métricas por meio de gráficos, tendo suporte a diversos tipos de bancos de dados, tanto gratuitos quanto pagos, e pode ser instalado em qualquer sistema operacional. Para facilitar a visualização dos gráficos, é possível criar *dashboards* dinâmicos que podem ser compartilhados com toda a organização. Além disso, a ferramenta permite configurar alertas com base nas métricas, que são analisadas de forma contínua para notificar o usuário sempre que for preciso, de acordo com as regras definidas

previamente. É bastante utilizado por sistemas de monitoramento para gerar gráficos *real-time* (Grafana Labs, 2023).

3

Mapeamento Sistemático da Literatura

Este capítulo apresenta, de forma breve, o Mapeamento Sistemático da Literatura (MSL) que serviu como base para o desenvolvimento deste trabalho. Esse MSL foi realizado seguindo a metodologia definida por Petersen et al. (2008) (Figura 8), com o intuito de identificar a possibilidade de desenvolvimento de um *cluster big data* de baixo custo. Para isso, foi definida a pergunta principal: "**Como é possível obter e validar um *cluster big data* de baixo custo?**", e para respondê-la, foram definidas as seguintes questões de pesquisa (QPs):

QP1: Quais são as estruturas de hardware utilizadas para obter um *cluster* de baixo custo?

QP2: Quais são as plataforma de *big data* que são utilizadas nos *clusters* ?

QP3: Quais são os algoritmos utilizados para validar os *clusters big data* encontrados nesses estudos?

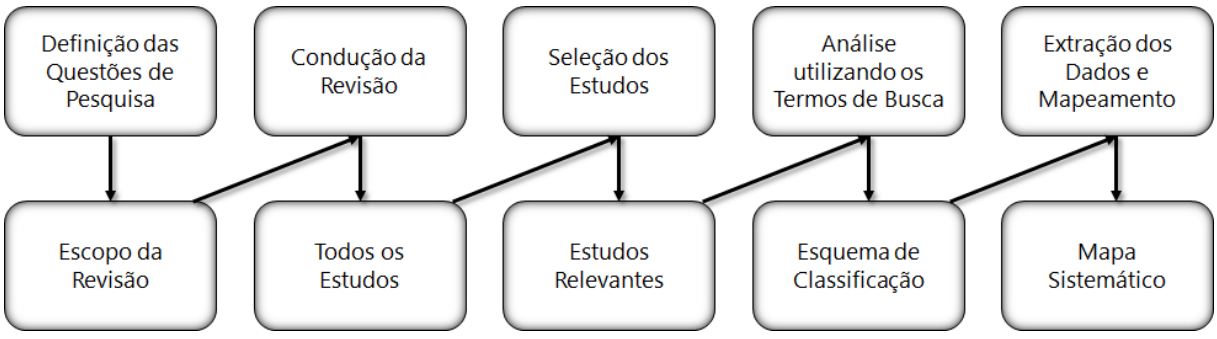

Figura 8 – Processo de MSL. Adaptado de Petersen et al. (2008).

A realização das buscas dos artigos foi feita nas principais bases de dados acadêmicas (*Scopus* e *Web of Science*). Foram encontrados 175 estudos, que após a aplicação de todos os

critérios de inclusão e exclusão presentes na metodologia, 14 artigos foram classificados como relevantes para responder as QPs.

O MSL identificou que a *Raspberry Pi* são as estruturas de hardware com maior incidência de citações nos artigos encontrados e que as plataformas de *big data* mais utilizadas são o *Apache Hadoop* e o *Apache Spark*. Além disso, o MSL também identificou que os algoritmos mais usados para a validação dos *clusters big data* foram os algoritmos *K-Means* e *Map Reduce*, esse último com base em sua abordagem original, ao contrário da abordagem do *Hadoop Map-Reduce*.

Esse MSL mostrou que existe a possibilidade de obter um *cluster big data* utilizando estruturas de baixo custo, mostrando resultados relevantes e fornecendo subsídios para um melhor entendimento sobre o desenvolvimento de *clusters de big data* de baixo custo.

Os detalhes deste capítulo são mostrados no artigo que foi aprovado como *Full Paper* e apresentado na *11ª Euro American Conference on Telematics and Information Systems - EATIS' 22*[1] (NETO; NETO; ORDONEZ, 2022) e pode ser visto no Apêndice A desta dissertação. Esse artigo pode contribuir para o desenvolvimento de outros estudos e ferramentas na temática abordada.

[1] <https://eatis.org/eatis2022/>

4

Quasi-Revisão Sistemática da Literatura

Este capítulo apresenta, de forma sucinta, a Quasi-Revisão Sistemática da Literatura (QRSL) desenvolvida, motivada como um trabalho futuro do MSL apresentado no Capítulo anterior (Capítulo 3). A QRSL foi desenvolvida a partir do protocolo de planejamento definido por Kitchenham (2004) (Figura 9), visando **identificar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados**. Para atingir esse objetivo, foram definidas as seguintes questões de pesquisa (QPs):

QP1: Quais são os modelos de *Raspberry Pis* que são utilizados para o desenvolvimento do *cluster big data*?

QP2: Quais são os *benchmarks*/algoritmos/técnicas que estão sendo utilizados no *cluster big data*?

QP3: Quais ferramentas são utilizadas para monitorar os recursos do *cluster*?

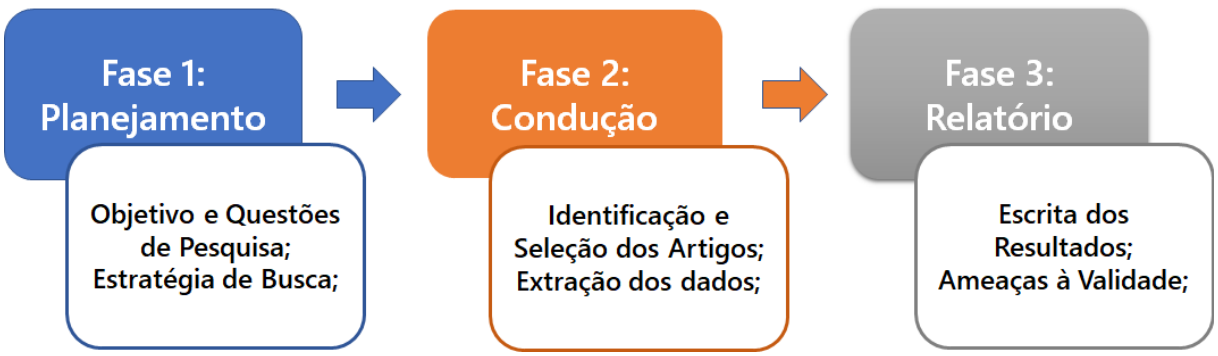

Figura 9 – Fases do Planejamento da QRSL - Adaptado de Kitchenham (2004).

As buscas dos artigos foram realizadas nas principais bases de dados acadêmicas (*Scopus* e *Web of Science*). Foram encontrados 17 estudos que, após a aplicação de todos os critérios de inclusão e exclusão presentes na metodologia, resultou em 10 artigos relevantes. Posteriormente, os 10 artigos foram analisados por completo, avaliando a capacidade dos mesmos em responder as QPs dessa QRSL. Dentre todos os trabalhos, apenas 1 não atendeu aos requisitos, resultando em 9 artigos que foram classificados como aptos a atingir o objetivo da QRSL.

A QRSL apontou cinco diferentes modelos de *Raspberry Pi* utilizados para o desenvolvimento dos *clusters*, sendo que o *Rapsberry Pi 2B* e o *Rapsberry Pi 4B* foram os modelos mais utilizados nos experimentos. Alguns trabalhos utilizaram mais de um modelo de *Raspberry Pi*, geralmente para fazer algum comparativo entre os resultados encontrados.

O *Terasort* e o *WordCount*, dentre 9 encontrados, foram os *benchmarks*/algoritmos/técnicas mais utilizados, seguidos do *TestDFSIO* e o *Pi Quasi-Monte Carlo*. Alguns estudos mencionam a utilização de mais de um *benchmarks*/algoritmos/técnicas.

Por fim, para resposta da QP3, apenas três trabalhos mencionaram o uso de alguma ferramenta para monitorar os recursos do *cluster*, sendo elas: *Ganglia*[1], *Grafana*[2] e *Prometheus*[3]. Em alguns dos trabalhos, os autores mencionaram o fato de monitorar os recursos do *cluster* sem utilizar uma ferramenta para tal.

O desenvolvimento desta QRSL foi importante, pois resultou no estado da arte que foi utilizado durante esta dissertação, sendo melhor explanado no Seção 5.

O artigo completo foi aprovado como *Work in Progress* e apresentado no *XII Brazilian Symposium on Computing Systems Engineering - SBESC 2022*[4] (NETO et al., 2022) e pode ser encontrado no Apêndice C desta dissertação.

[1] <http://ganglia.sourceforge.net/>
[2] <https://grafana.com/>
[3] <https://prometheus.io/>
[4] <https://sbesc.lisha.ufsc.br/sbesc2022/>

5
Estado da Arte

Este capítulo busca oferecer uma visão dos trabalhos encontrados, relacionados ao tema proposto nesta dissertação, além de mostrar um comparativo entre os mesmos. Está dividido em 2 seções: A Seção 5.1 faz uma síntese de cada trabalho, mostrando o que foi feito em cada um desses estudos, e na Seção 5.2 são apontadas as principais características dos trabalhos encontrados, além de relacioná-las com as do trabalho proposto.

5.1 Trabalhos Relacionados

Nesta seção são apresentados cada um dos 9 trabalhos relacionados ao tema proposto nesta dissertação, fazendo uma síntese de cada um deles, apresentando as principais contribuições dos mesmos.

5.1.1 *Big Data Processing on Single Board Computer Clusters: Exploring Challenges and Possibilities*

Nesse trabalho, (LEE; OH; PARK, 2021), os autores exploraram os desafios das últimas gerações de *Raspberry Pis* utilizados em *clusters big data*. Eles construíram 1 *cluster big data* composto por 6 nós (5 escravos e 1 mestre), onde cada nó escravo utiliza 1 *Raspberry Pi 4B* e o nó mestre utiliza 1 *Desktop* (Processador i5 de 3.7GHz (6 *cores*), *Solid State Drive* (SSD) de 500 *Gigabytes* (GB) e Memória RAM de 8 GB) e *Apache Hadoop*.

Os autores realizaram uma bateria de testes, a fim de avaliar a performance de uma única *Raspberry Pi*, assim como a performance do *cluster* desenvolvido, utilizando alguns dos principais *benchmarks*: *Wordcount*, *Terasort*, *TestDFSIO* e o *Pi Quasi-Monte Carlo*.

Eles compararam o *cluster* desenvolvido com o *Desktop* e com 1 *cluster big data* composto por 6 *Raspberry Pis 3B* (1 nó mestre e 5 nós escravos), além de estudarem o impacto

da performance no armazenamento de dados utilizando 3 diferentes tipos de dispositivos: *Micro Storage Device Card* (MicroSD *Card*) de 32 GB; MicroSD *Card* 64 GB; *Universal Flash Storage - UFS* de 256 GB.

Os mesmos concluíram que, ao utilizar uma versão mais recente do *hardware Raspberry Pi*, o *cluster* tem uma melhor capacidade de processamento, bem como a utilização do dispositivo UFS, como unidade de armazenamento, tem um melhor *ratio* (*Input/Output*) em relação ao uso de MicroSD *Card*.

5.1.2 *Hadoop Performance Analysis on Raspberry Pi for DNA Sequence Alignment*

Nessa pesquisa, (TURANA; SUKOCO; KUSUMA, 2016), os autores buscaram analisar a sequência de alinhamento de DNA. Para isso, foi construído 1 *cluster Hadoop* na *Raspberry Pi B* usando o hardware da *Raspberry* como *commodity*. Foram utilizados 6 nós no *cluster* (1 nó mestre e 5 nós escravos). Eles usaram a biblioteca nativa *Biodoop* e o *benchmark Wordcount* para os testes, além de utilizar a ferramenta *Ganglia* para o monitoramento dos recursos do *cluster*.

Os autores realizaram uma comparação entre o *cluster Raspberry Pi B* e 6 *Desktops* (processador *dual core* de 1,86 GHz, *Hard Disk Drive (HDD)* de 160 GB e Memória RAM de 1 GB).

Foi concluído que é possível utilizar o *hardware* alternativo de baixo custo para implementar o *cluster Hadoop*, mesmo com a desvantagem do aumento de tempo na conclusão dos experimentos comparados.

5.1.3 *Understanding the Performance of Low Power Raspberry Pi Cloud for Dig Data*

Esse trabalho, (HAJJI; TSO, 2016), apresenta um conjunto de experimentos para testar a performance de 1 nó único e de 1 *cluster* composto por 12 *Raspberry Pis 2B* (1 mestre e 11 escravos) com e sem ambiente de virtualização, utilizando a ferramenta *Docker* para estudar a viabilidade do mesmo para *big data analyics* em tempo real.

Utilizando os *benchmarks Wordcount* e *Terasort*, os autores executaram uma bateria de testes com diversas configurações de tamanhos de arquivos, variando entre 1 GB a 6 GB. Os mesmos avaliaram que, em um ambiente virtualizado, o consumo de CPU e memória RAM torna-se maior, a taxa de transferência da rede diminui e os picos de acessos ocorrem com menos frequência e menos intensidade.

5.1.4 *Towards Green Data Centers*

No artigo (BOURHNANE et al., 2020), foi implementado um comparativo de *hardware* no uso do *Apache Hadoop*. Enquanto um experimento usou 1 *Desktop* (2 processadores *Core 2 Duo de 2.20 GHz, HDD de 160GB*), no outro foi utilizado 1 *cluster* com 5 *Raspberry Pis 3B+*, sendo 1 nó mestre e 4 nós escravos. Esse *cluster* tinha como diferencial, o uso de 5 HDDs de 1 TB (externos), além de 5 MicroSD *Card* (internos) de 8 GB.

A finalidade da pesquisa era agregar uma motivação no uso da prática de computação eficiente e ecológica, ou seja, na computação verde. Para tanto, foram executados testes com os *benchmarks TestDFSIO* e *Terasort* para a análise de desempenho e eficiência energética do *cluster*, utilizando a *Raspberry Pi*.

Os experimentos mostraram desempenho significativo com o *TestDFSIO* em relação ao *cluster* tradicional. No entanto, o *Terasort* forneceu um menor desempenho, que pode ser facilmente superado adicionando mais nós ao *cluster*. Nos testes de consumo energético no *cluster Raspberry Pi*, foi comprovado o baixo consumo de energia, provando ser esse, um experimento de computação verde.

5.1.5 *An Efficient Implementation of Mobile Raspberry Pi Hadoop Clusters for Robust and Augmented Computing Performance*

Nesse trabalho, (SRINIVASAN et al., 2018), os autores utilizaram 6 *clusters* diferentes, sendo eles configurados com 5 a 10 *Raspberry Pis 3B*, sempre utilizando o padrão de 1 nó mestre e os nós remanescentes como escravos. Também foi utilizado 1 *Desktop* (processador i5, 3.1 GHz, Memória RAM de 8 GB) para os mesmos experimentos.

Os autores exploraram a implementação do *Hadoop*, tanto para o *cluster* construído com *Raspberry Pi* quanto para o *Desktop*. Tais experimentos visavam testes de extração de pontos de recurso em uma imagem utilizando o algoritmo *Surf* e comparando os resultados obtidos.

Os pesquisadores verificaram que os *clusters* formados por *Raspberry Pi* têm melhor performance que o *Desktop* para grandes *datasets*.

5.1.6 *A containerized big data streaming architecture for edge cloud computing on clustered single-board devices | A Containerized Edge Cloud Architecture for Data Stream Processing*

Em ambos artigos, (SCOLATI et al., 2019; SCOLATI et al., 2020), foram realizados experimentos em 1 *cluster* de 8 *Raspberry Pis 2B* com o gerenciador *Docker Swarm*. Ao *cluster* foi conferida a seguinte configuração: 1 nó mestre, 4 nós escravos e 3 nós para coleta de dados do *cluster*.

Os autores implementaram o *cluster* utilizando o *Docker* para hospedar o *Apache Hadoop* e o *Apache Spark* como *containers* e, dessa forma, aprimorar o processamento de *streaming de dados*. Para monitorar os recursos do *cluster*, os mesmos usaram as ferramentas *Prometheus* e *Grafana*.

Eles avaliaram o desempenho dos mesmos, mostrando que o uso da "conteinerização" aumenta a tolerância a falhas e a facilidade de manutenção.

5.1.7 *Scale-down Experiments on TPCx-HS*

Nesse estudo, (BöTHER; RABL, 2021), os autores fizeram a execução do *benchmark* TPCx-HS em 2 *clusters* utilizando *Raspberry Pi*: O primeiro *cluster* composto por 4 *Raspberry Pis 3B+* trabalhando como escravos e 1 *Raspberry Pi 4B* como mestre, utilizando *Micro SD Cards* de 32 GB em cada dispositivo. Já o segundo *cluster*, composto por 5 *Raspberry Pis 4B* (1 mestre e 4 escravos), em que cada *Raspberry* utilizou MicroSD *Card* de 128 GB para armazenamento.

Nos experimentos, foram utilizadas 2 configurações diferentes nos tamanhos dos arquivos usados pelo *benchmark* TPCx-HS: 1 GB e 10 GB. Em seguida, foi feita a comparação dos resultados obtidos com os disponíveis no site[1] do *benchmark*.

Os autores concluíram que a geração de *Raspberry Pi 4B* resolve o gargalo de memória RAM que existia na geração de *Raspberry Pi 3B+*. Além disso, os mesmos defendem o uso de *clusters Raspberry Pi* devido ao seu custo-benefício.

5.1.8 *Performance of Raspberry Pi microclusters for Edge Machine Learning in Tourism*

No desenvolvimento dessa pesquisa, (KOMNINOS et al., 2019), foram utilizadas 6 *Raspberry Pis 4B*, com 4 GB de RAM e cartão MicroSD *Card* de 64 GB em cada nó. No *cluster* foram instalados o *Hadoop* para armazenamento distribuído de arquivos e o *Spark* para aprendizado de máquina.

Tal estudo tinha como objetivo utilizar recursos computacionais em uma arquitetura distribuída de baixo custo, para atender aplicações turísticas através da análise de *big data*.

Após experimentos e testes, foi demonstrado que o desempenho do *cluster* foi suficiente para fins de treinamento e execução de modelos de aprendizado de máquina em um contexto de computação de borda.

[1] <https://www.tpc.org/tpcx-hs/results/tpcxhs_perf_results5.asp?version=2>

5.2 Análise Comparativa

Na Tabela 1 é feito um comparativo entre as características presentes nos trabalhos descritos anteriormente, com o trabalho proposto nesta dissertação. Nela é possível comparar o uso das plataformas de hardware utilizadas, o número de nós escravos utilizados nos experimentos, os *benchmarks* utilizados para validar a performance e o comportamento do *cluster*, além das ferramentas (monitoramento e auxiliares) em cada um dos artigos e principais referências encontradas neste trabalho.

Por meio desse comparativo é possível observar que este trabalho utiliza um hardware mais atualizado, além de variar no número de nós escravos usados no *cluster* (2, 4 e 8 nós). Também apresenta um modelo de passo a passo de como desenvolver e configurar o *cluster* do início ao fim. Esse modelo pode ajudar à comunidade e facilitar a montagem e utilização desses *clusters*.

Os *benchmarks* utilizados (*Terasort* e *TestDFSIO*) são bem difundidos na comunidade *Big Data*. Também foram utilizadas ferramentas que coletam, armazenam e disponibilizam as informações do *cluster* em *dashboards*, desde um nó único, até o mesmo como um todo, em tempo real usando (*Zabbix* e *Grafana*).

Por fim, neste trabalho, o tamanho dos arquivos executados nos *benchmarks* foram alterados para ser observado o impacto de utilizar arquivos grandes nas aplicações e mostrar métricas diferentes (como uso de CPU, memória usada, estatísticas de tráfego de rede, temperatura do nó, etc.) durante sua execução em tempo real, além de utilizar 2 configurações diferentes na capacidade de armazenamento dos nós do *cluster* (16 GB e 128 GB).

Tabela 1 – Características entre os trabalhos relacionados

Referência	Hardware	Nós Escravos	*Passo a Passo*	*Benchmark*	Ferramenta Monitoramento	Ferramenta Auxiliar
Lee, Oh e Park (2021)	*Raspberry Pi 4B*, *Raspberry Pi 3B*	5	Não	*Terasort*, *TestDFSIO*, *Pi Quasi-Monte Carlo*, *Wordcount*	X	X
Scolati et al. (2019), Scolati et al. (2020)	*Raspberry Pi 2B*	4	Não	X	*Prometheus*, *Grafana*	*Docker*
Hajji e Tso (2016)	*Raspberry Pi 2B*	11	Não	*Wordcount*, *Terasort*	X	*Docker*
Srinivasan et al. (2018)	*Raspberry Pi 3B*	4, 5, 6, 7, 8, 9	Não	*Surf*	X	X
Turana, Sukoco e Kusuma (2016)	*Raspberry Pi B*	5	Não	*Wordcount*	*Ganglia*	X
Bourhnane et al. (2020)	*Raspberry Pi* 3B+	4	Não	*TestDFSIO*, *Terasort*	X	X
Böther e Rabl (2021)	*Raspberry Pi 4B*, *Raspberry Pi 3B+*	4	Não	*TPCx-HS*	X	X
Komninos et al. (2019)	*Raspberry Pi 4B*	5	Não	Árvore de Decisão, Regressão Linear	X	X
Este Trabalho	*Raspberry Pi 4B*, *Raspberry Pi 3B+*’	2, 4, 8	Sim	*TestDFSIO*, *Terasort*	*Zabbix*, *Grafana*	X

6

Desenvolvimento do *Cluster Big Data* de Baixo Custo usando *Apache Hadoop* e *Raspberry Pi*

O presente capítulo apresenta brevemente o processo de desenvolvimento e montagem do *cluster big data* de baixo custo. Todo esse processo completo e detalhado pode ser visualizado no artigo que foi aprovado e publicado no periódico ***Computers and Electrical Engineering***[1] (NETO; NETO; MORENO, 2022) e está disponível no Apêndice D desta dissertação.

6.1 *Cluster Setup*

Nesta Seção, todos os recursos (hardware e software) utilizados para o desenvolvimento do *cluster* de baixo custos são apresentados.

6.1.1 Estruturas de Hardware Utilizadas

O *cluster* desenvolvido é composto por 9 *Raspberry Pis* 4B, sendo 1 *master* e os demais escravos.

Cada *Raspberry Pi* utiliza um cartão de memória de 16/128 GB (*)[2] para melhorar a sua capacidade de armazenamento. Todas as *Raspberry Pis* estão conectadas em uma mesma rede através de um *Switch* de 16 portas. De forma a ter um melhor monitoramento do uso dos recursos do *cluster*, foi utilizada 1 *Raspberry Pi* 3B+ como servidor de monitoramento.

Na Tabela 2 todos os hardwares utilizados são apresentados de forma detalhada, além do preço de cada um. Os preços foram obtidos em um *e-commerce*, em Abril de 2022. Os preços foram orçados em Dólar ($).

[1] <https://www.sciencedirect.com/journal/computers-and-electrical-engineering/>

[2] No Capítulo 7 são executadas 2 baterias de testes no *cluster*. Em cada uma delas, cada nó utiliza uma capacidade de armazenamento diferente.

Tabela 2 – Hardwares usados para o desenvolvimento do *cluster* de baixo custo.

Hardware	Quantidade	Preço Unitário	Preço Total
Case	10	$3	$30
Cartão de Memória MicroSD Classe 10 16 GB HC-I	10	$5	$50
Cartão de Memória MicroSD Classe 10 128 GB XC-I	10	$25	$250
Raspberry Pi 3B+	1	$35	$35
Raspberry Pi 4B	9	$55	$495
Switch	1	$35	$35
Cabo UTP	10	$1	$10
Total	-	-	**$905**

Cada *Raspberry Pi* utilizada para o desenvolvimento deste trabalho tem a seguinte configuração, conforme pode ser observado na Tabela 3.

Tabela 3 – Configuração das *Raspberry* Pis.

Feature	***Raspberry Pi 4B***	***Raspberry Pi 3B+***
Ethernet	1 GB	1 GB
Mémoria RAM	4 GB	1 GB
CPU	Cortex A-72 Quad Core x64 (1.5 GHz)	Cortex A-53 Quad Core x64 (1.2 GHz)
Capacidade de Armazenamento	16/128 GB	16/128 GB

6.1.2 Softwares Utilizados

O sistema operacional utilizado em todas as *Raspberry Pis* foi o *Raspberry Pi OS*[3].

A plataforma *big data* escolhida para ser instalada no *cluster* foi a *Apache Hadoop*[4]. De forma a se obter um melhor monitoramento do *cluster* e dos seus recursos, foi usado o Zabbix[5] para coletar e armazenar todas as métricas e para exibí-las foram utilizadas as interfaces gráficas providas pelo Grafana[6].

A Tabela 4 apresenta todos os softwares utilizados neste trabalho de forma detalhada, sendo importante salientar que todos eles são *open-source* ou na versão gratuita.

Tabela 4 – Softwares utilizados no desenvolvimento do *cluster*.

Software	**Versão**
Apache Hadoop	3.2.0
Grafana	8.1.5
Java Development Kit (JDK)	1.8.0_212
Rasperry Pi OS	5.10
Zabbix Agent	5.4.6
Zabbix Zerver	5.4.4

[3] <https://www.raspberrypi.com/software/>
[4] <https://hadoop.apache.org/>
[5] <https://www.zabbix.com/>
[6] <https://grafana.com/>

6.2 Passo a Passo de Instalação

Vide Seção 5 - *Cluster Development - Step-by-Step* em Neto, Neto e Moreno (2022).

6.3 *Cluster* Desenvolvido

O *cluster* desenvolvido durante este trabalho, pode ser visualizado na Figura 10. As *Raspberry Pis* que estão com *cases* de cor vermelha são as do modelo "4B", utilizadas como nós do *cluster*. Já a *Raspberry Pi* 3B+, que foi utilizada como servidor de monitoramento, utiliza a *case* verde.

Figura 10 – *Cluster* desenvolvido.

O *Apache Hadoop* tem uma interface gráfica que possibilita ao usuário ter uma visão geral do *cluster*, disponibilizando informações relevantes do mesmo, como capacidade total de armazenamento usada e ainda disponível, quantidade de nós ativos, etc. Algumas dessas informações podem ser visualizadas na Figura 11.

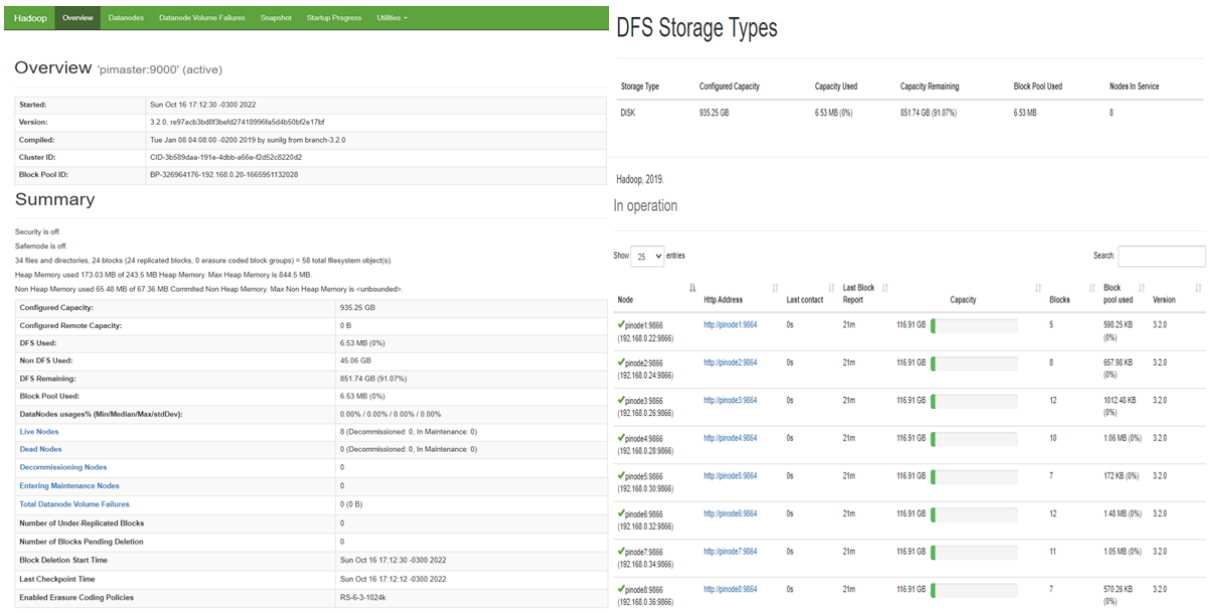

Figura 11 – Interface gráfica do *Apache Hadoop*.

Entretanto essa interface não possibilita visualizar informações inerentes aos hardwares que compõem o *cluster*, mostrando o consumo de recursos importantes como: CPU, memória RAM, temperatura, etc. Para isso foi desenvolvido um *dashboard* no *Grafana*, com as informações que foram coletadas pelo *Zabbix*. Esse *dashboard* pode ser visto na Figura 12.

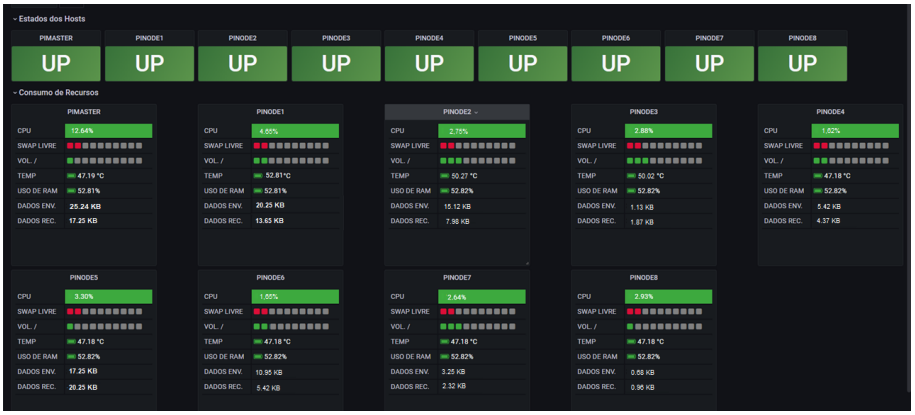

Figura 12 – *Dashboard* geral dos nós do *cluster* - *Grafana*

Além desse *dashboard* com um *overview* (visão geral), foi desenvolvido um *dashboard* específico para cada componente do *cluster*, contendo as mesmas informações do *dashboard* mencionado na Figura 12, porém de forma mais detalhada. Um desses *dashboards* implementados pode ser visualizado na Figura 13.

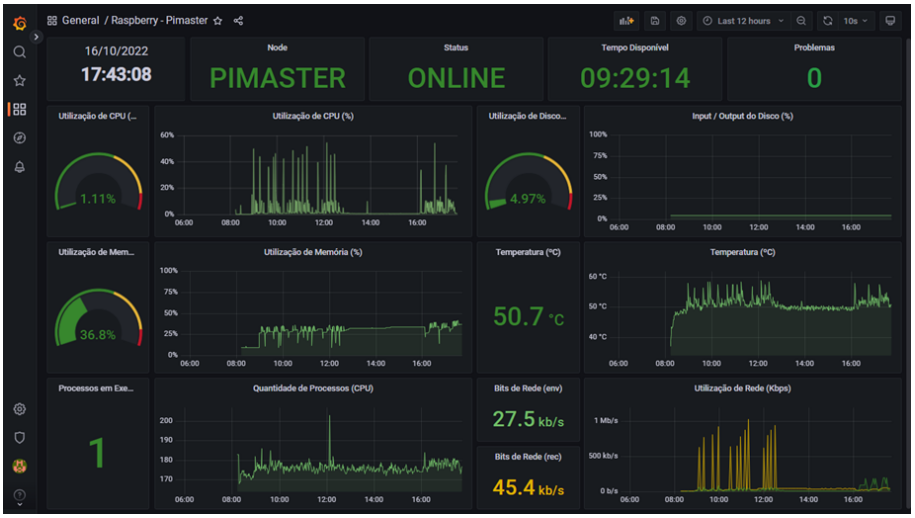

Figura 13 – *Dashboard* detalhado dos nós do *cluster* - *Grafana*

7

Benchmarking do *Cluster*

Este capítulo mostra todo o processo de testagem e validação do *cluster* desenvolvido e montado no capítulo anterior (Capítulo 6). Para isso, foram executados os *benchmarks Terasort* e *TestDFSIO* no *cluster* em 3 configurações diferentes: a primeira com 1 nó mestre e 2 nós escravos; a segunda com 1 nó mestre e 4 escravos; e na última com 1 nó mestre e todos os demais nós como escravos, totalizando 8 nós escravos.

Além disso, todo esse processo de testagem e validação foi feito com 2 capacidades de armazenamento diferentes: a 1ª, onde cada nó do *cluster* utilizou um cartão de memória de 16 GB e na 2ª, cada nó estava com um cartão de memória de 128 GB. Houve também uma variação entre o tamanho dos arquivos testados (250 MB, 500 MB, 750 MB e 1 GB), tendo seu limite máximo de 1 GB devido a limitação dos cartões de memória de 16 GB de armazenamento.

Também são apresentados neste capítulo, através dos *dashboards* desenvolvidos, o monitoramento dos recursos do *cluster* durante a execução dos *benchmarks*. Toda essa gama de opções permite observar o comportamento do *cluster* em vários tipos de situações, verificando qual delas obteve um melhor desempenho.

7.1 *Benchmark Terasort*

Nesta seção são apresentados os resultados do *benchmark Terasort*, mostrando a performance do *cluster* em cada uma das etapas, são elas: *Teragen*, *Terasort* e *Teravalidate*. Além dos resultados, ela também apresenta os *dashboards* contendo as informações dos recursos do *cluster* durante a bateria de testes.

7.1.1 Resultados - *Terasort*

A execução e coleta completa dos resultados do *benchmark Terasort* teve um total de aproximadamente 8 horas de duração, sendo que a primeira bateria de testes demorou cerca de 5

horas (16 GB) e a última aproximadamente 3 horas (128 GB). Durante a realização dos testes para este *benchmark*, foram coletadas as seguintes métricas: tempo total da execução do teste (Tempo Real), tempo total gasto na realização das tarefas de *Map* (Tempo de *Map*), tempo total gasto na realização das tarefas de *Reduce* (Tempo de *Reduce*) e tempo total gasto no processamento realizado pela CPU (Tempo de CPU).

As Tabelas 5, 6 e 7 mostram a performance do *cluster* durante a execução dos testes envolvendo as etapas *Teragen, Terasort* e *Teravalidate* respectivamente. A etapa do *Teragen* realiza apenas tarefas de *Map* para a geração dos dados, desta forma, a Tabela 5 não contempla nenhum valor para a métrica de Tempo de *Reduce*. Uma síntese de forma gráfica desses resultados pode ser visualizada na Figura 14.

Durante a etapa de geração dos dados (*Teragen* - Tabela 5), para os cartões de memória de 16 GB, o *cluster* com 2 nós escravos apresentou uma média de 1,51 minutos gastos para a finalização da tarefa (Tempo Real). Quando sua capacidade de processamento foi aumentada (4 nós), a média do tempo total de execução diminuiu para 1,29 minutos gastos. E por fim, incrementando ainda mais a capacidade de processamento do *cluster* (8 nós), o mesmo concluiu a etapa *Teragen* em 1,23 minutos. Com a capacidade de armazenamento de 128 GB, o *cluster* com 2 nós escravos apresentou uma média de 1,18 minutos gastos para a finalização da tarefa. Incrementando sua quantidade de nós para 4, a média do tempo total de execução diminuiu para 1,13 minutos gastos. E por fim, com 8 nós, o *cluster* concluiu a etapa *Teragen* em aproximadamente 1 minuto. Isso mostra o ganho em ter *clusters* com mais nós processando informações.

Os dados coletados e apresentados para a etapa de ordenação dos dados (*Terasort* - Tabela 6) mostram que com a capacidade de armazenamento de 16 GB, o *cluster* com 2 nós escravos apresentou uma média de 4,87 minutos gastos para a finalização da tarefa (Tempo Real). Quando sua capacidade de processamento foi aumentada (4 nós), a média do tempo total de execução diminuiu para 4,30 minutos gastos. E por fim, incrementando ainda mais a capacidade de processamento do *cluster* (8 nós), o mesmo concluiu a etapa *Terasort* em 3,63 minutos. Com uso dos cartões de 128 GB, o *cluster* com 2 nós escravos apresentou uma média de 2,31 minutos gastos para a finalização da tarefa. Incrementando sua quantidade de nós para 4, a média do tempo total de execução diminuiu para 1,79 minutos gastos. E com 8 nós, o *cluster* concluiu esta etapa em 1,60 minutos.

Por fim a Tabela 7, com as informações da etapa de validação dos dados ordenados (*Teravalidate*), utilizando os cartões de memória de 16 GB, mostra que o *cluster* concluiu a etapa *Teravalidate* em aproximadamente 1 minuto, independentemente de sua quantidade de nós escravos. Já para a utilização dos cartões de 128 GB, o *cluster* com 2 e 4 nós escravos apresentou uma média de 0,65 minutos gastos para a finalização da tarefa. Com 8 nós, foi obtida uma média de 0,54 minutos para a conclusão da etapa de validação dos dados.

A partir dos dados expostos nessas tabelas, é possível observar que a medida que o tamanho do arquivo aumenta, há um esforço maior por parte do *cluster* com uma menor

quantidade de nós para realizar o processamento dessas tarefas. Esse esforço pode ser observado através do tempo total gasto pelo mesmo para executa-las. Isso acontece pelo fato do *cluster* com uma menor capacidade (quantidade de nós) ter poucos recursos de hardware agregados, isto é, menos máquinas para processar um maior volume de tarefas (blocos), exigindo assim um maior consumo de recursos computacionais, aumentando assim o tempo de execução dessas tarefas.

A medida que a capacidade de processamento do *cluster* cresce, aumentando a quantidade nós, observa-se que as tarefas são decompostas e distribuídas de forma mais eficiente no *cluster*, reduzindo o tempo gasto na execução e no processamento das mesmas. Dessa forma, conclui-se que é necessário aumentar o número de nós no *cluster* à medida que o tamanho dos arquivos a serem processados aumenta, até que seja alcançado um cenário ideal para um bom desempenho do *cluster* e esse problema seja sanado, com exceção da etapa de *Teravalidate*, onde notou-se uma homogeneidade para a capacidade de armazenamento de 16 GB e uma leve diferença para o uso dos cartões de memória de 128 GB.

Também foi possível observar que há uma grande diferença de performance do *cluster* quando se tem uma maior capacidade de armazenamento em seus nós. Isso se dá pelo fato de que os cartões de memória de 16 GB usa a tecnologia *High Capacity* (HC), diferentemente da tecnologia usada nos cartões de memória de 128 GB, que usa a tecnologia XC (*Extended Capacity*). Em suma, a tecnologia XC trabalha com arquivos em formato exFAT (*Extended File Allocation Table*), que possui uma maior velocidade de escrita/leitura (*I/O*) e permite a gravação de arquivos muito grandes, diferentemente do formato de arquivos FAT (*File Allocation Table*) que é utilizado para os cartões de memória com tecnologia HC (Kingston Technology, 2020).

Em paralelo à realização do *benchmarking* do *cluster*, foram coletadas informações inerentes ao comportamento do mesmo, a partir de um monitoramento ativo. Todo esse estágio será melhor explanado na seção 7.1.2.

Tabela 5 – Resultados usando o *benchmark Terasort: Teragen*.

Capacidade	Nós	Tamanho Arquivo (MB)	Tempo Real (s)	Tempo de *Map* (ms)	Tempo de *Reduce* (ms)	Tempo de CPU (ms)
16 GB	2	250	1m02,225s	61.275	X	22.050
		500	1m32,724s	90.705	X	41.060
		750	1m39,472s	106.762	X	60.180
		1000	2m29,327s	174.796	X	79.770
	4	250	1m01,723s	61.050	X	22.050
		500	1m28,652s	95.130	X	41.160
		750	1m26,708s	97.168	X	60.150
		1000	1m57,575s	150.444	X	79.120
	8	250	1m00,426s	60.813	X	22.180
		500	1m17,605s	80.538	X	40.960
		750	1m22,929s	87.454	X	59.360
		1000	1m49,702s	127.097	X	78.490
128 GB	2	250	1m00,789s	44.814	X	18.540
		500	1m14,424s	59.767	X	36.860
		750	1m28,662s	75.824	X	50.310
		1000	1m29,593s	103.377	X	61.740
	4	250	1m00,037s	48.983	X	19.460
		500	1m09,699s	72.502	X	34.600
		750	1m16,064s	93.752	X	44.110
		1000	1m35,080s	105.786	X	54.980
	8	250	0m58,425s	39.420	X	18.790
		500	1m05,929s	51.724	X	35.900
		750	1m14,047s	75.772	X	47.570
		1000	1m24,715s	89.750	X	58.180

Tabela 6 – Resultados usando o *benchmark Terasort: Terasort*.

Capacidade	Nós	Tamanho Arquivo (MB)	Tempo Real (s)	Tempo de *Map* (ms)	Tempo de *Reduce* (ms)	Tempo de CPU (ms)
16 GB	2	250	3m18,552s	212.925	76.268	93.410
		500	4m25,012s	197.384	195.589	183.980
		750	5m58,264s	379.467	290.146	275.400
		1000	6m46,948s	1.323.027	280.617	380.530
	4	250	2m51,876s	149.442	69.583	92.080
		500	4m09,103s	372.285	170.074	187.790
		750	5m25,683s	749.372	218.505	279.150
		1000	5m33,831s	1.015.007	249.850	382.190
	8	250	2m09,256s	101.726	63.221	89.580
		500	3m13,397s	274.274	103.761	182.740
		750	4m22,970s	327.033	197.895	275.320
		1000	5m08,844s	683.331	254.715	381.930
128 GB	2	250	1m35,297s	57.308	34.318	63.530
		500	2m29,127s	231.680	57.590	117.400
		750	2m49,262s	307.732	110.490	166.710
		1000	3m10,745s	442.866	98.317	216.370
	4	250	1m27,252s	40.948	35.518	61.430
		500	1m48,494s	106.581	59.072	110.340
		750	2m06,049s	140.392	73.646	158.010
		1000	2m36,341s	305.000	96.663	209.780
	8	250	1m22,147s	36.777	33.160	57.140
		500	1m40,770s	83.958	49.944	108.790
		750	1m57,645s	155.445	69.150	158.360
		1000	2m21,481s	255.818	93.636	208.630

Tabela 7 – Resultados usando o *benchmark Terasort: Teravalidate*.

Capacidade	Nós	Tamanho Arquivo (MB)	Tempo Real (s)	Tempo de *Map* (ms)	Tempo de *Reduce* (ms)	Tempo de CPU (ms)
16 GB	2	250	0m52,880s	18.837	5.875	14.340
		500	1m03,795s	30.559	5.936	26.170
		750	1m16,912s	43.967	5.946	39.470
		1000	1m29,995s	56.603	5.987	52.160
	4	250	0m51,536s	18.397	6.592	14.080
		500	1m03,624s	30.656	6.277	26.490
		750	1m16,872s	42.708	6.299	38.610
		1000	1m28,086s	55.741	6.621	51.790
	8	250	0m50,625s	18.408	5.973	13.820
		500	1m03,012s	31.771	6.628	27.210
		750	1m14,975s	41.814	5.874	37.330
		1000	1m26,198s	55.197	5.999	50.860
128 GB	2	250	0m48,705s	11.640	7.605	9.410
		500	0m53,805s	14.887	7.394	12.820
		750	0m58,925s	17.927	7.242	15.730
		1000	1m02,624s	23.699	7.146	22.090
	4	250	0m50,200s	11.145	10.300	9.180
		500	0m52,923s	14.771	7.639	13.040
		750	0m58,808s	19.904	7.726	18.800
		1000	1m00,706s	22.802	7.827	20.550
	8	250	0m50,802s	12.621	7.711	10.030
		500	0m52,955s	15.451	7.547	13.440
		750	0m56,144s	20.037	7.567	18.070
		1000	0m59,710s	21.866	7.818	20.260

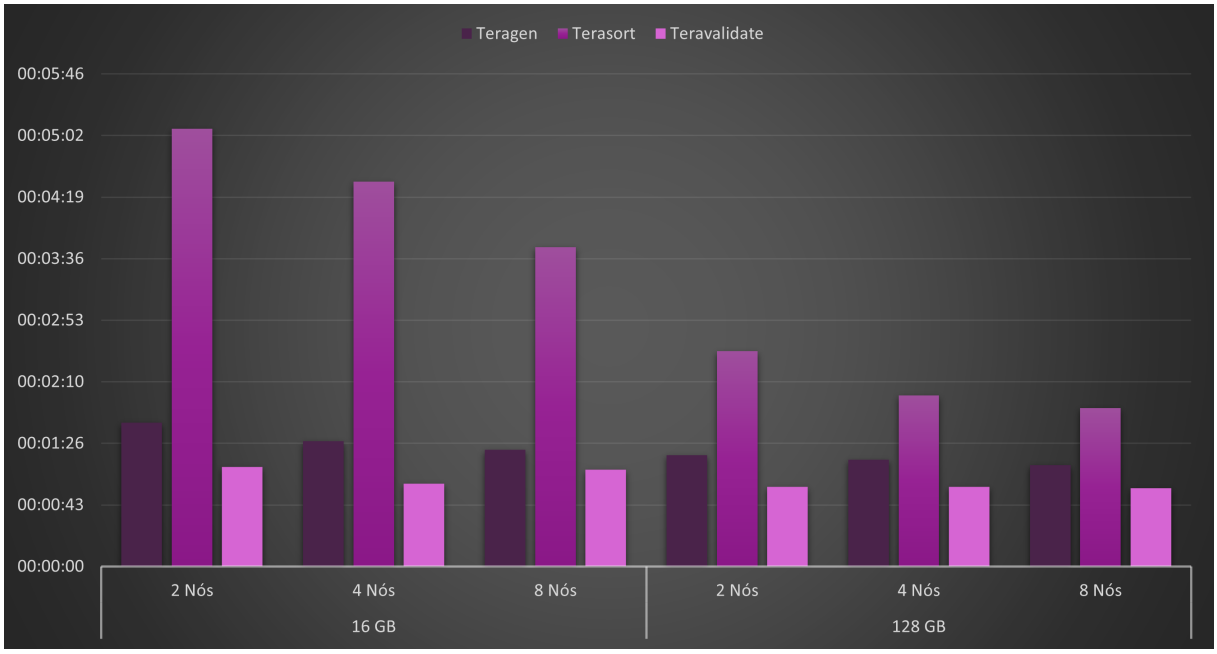

Figura 14 – Síntese dos resultados do *benchmark Terasort*.

7.1.2 Monitoramento - *Terasort*

O monitoramento dos recursos do *cluster* durante o processo de *benchmarking* demonstrado na subseção anterior são apresentados nesta subseção. O mesmo está dividido em dois cenários de forma análoga ao processo de testagem do *cluster*: o primeiro com os nós com capacidade de armazenamento de 16 GB e o segundo com o *cluster* utilizando os cartões de memória de 128 GB em cada um dos seus nós.

Todas as métricas coletadas durante a testagem utilizando o *benchmark Terasort* podem ser visualizadas nas Figuras 15, 16, 17, 18, 19 e 20. Nas 3 primeiras Figuras utilizando os cartões de 16 GB e as 3 últimas com armazenamento de 128 GB.

Em ambos os cenários é possível observar a grande quantidade processos que são executados no *cluster*, com picos variando entre 140 e 160 processos sendo executados para os cartões de 16 GB e chegando a 190 processos em execução quando se utilizou os cartões de 128 GB. Essa grande quantidade de processos necessita de um maior esforço para processa-los. Esse esforço pode ser visto através da análise dos dados de utilização do processador (CPU), contendo picos durante a bateria de testes, chegando a quase 100% de uso, e que consequentemente necessita de mais energia, aumentando assim a temperatura dos dispositivos utilizados, na qual os nós analisados, em modo ocioso, trabalham em uma temperatura que varia entre 42 $^{\circ}C$ e 45 $^{\circ}C$ e quando requisitados há um aumento da temperatura, chegando a picos de até 60 $^{\circ}C$.

A utilização de disco *(input/output)* permaneceu de forma constante em aproximadamente 64% para todos os nós do *cluster* utilizando os cartões de 16 GB, com exceção do **Pinode2**,

onde o uso de seu disco chegou a quase 75%. Diferentemente de quando o *cluster* trabalhou com uma maior capacidade de armazenamento em seus nós (cartões de 128 GB), quando a sua utilização de disco *(input/output)* caiu de forma drástica, operando em aproximadamente 5% da sua capacidade total.

Ao se analisar a utilização da memória RAM, foi visto que não houve um consumo excessivo das mesmas nos dois cenários de testes. Enquanto para o uso dos cartões de memória de 16 GB os nós do *cluster* não ultrapassaram o valor de 25% de uso de RAM, na utilização dos cartões de 128 GB os nós do *cluster* utilizaram mais que 25% de memória RAM, chegando a um valor máximo de aproximadamente 60%.

Também é possível verificar um tráfego intenso na rede, tanto no envio, quanto no recebimento de pacotes. Vale ressaltar que um *cluster* composto por um maior número de nós, requer uma maior quantidade de troca de informações através da rede de interconexão. Sendo necessário o uso de equipamentos de alta velocidade para que a simultaneidade e os atrasos na rede (*delay*) possam ser reduzidos ou para diminuir o congestionamento potencial ao trocar mensagens entre os nós.

7.1.2.1 Armazenamento de 16 GB

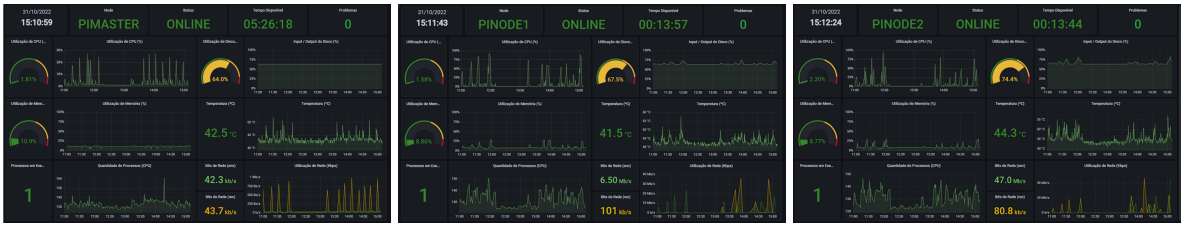

Figura 15 – Comportamento do *cluster* com armazenamento de 16 GB - *Terasort* - Parte 1.

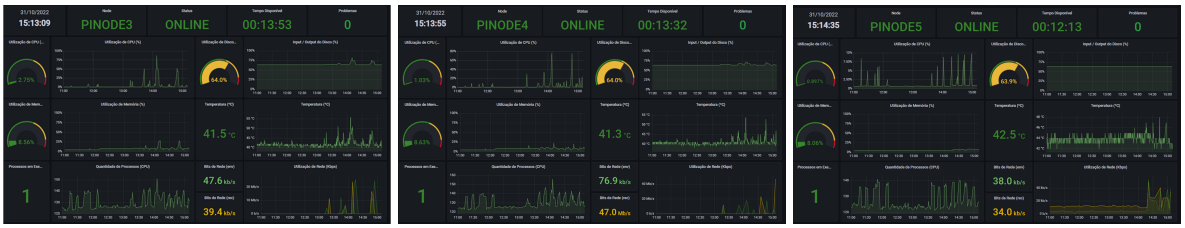

Figura 16 – Comportamento do *cluster* com armazenamento de 16 GB - *Terasort* - Parte 2.

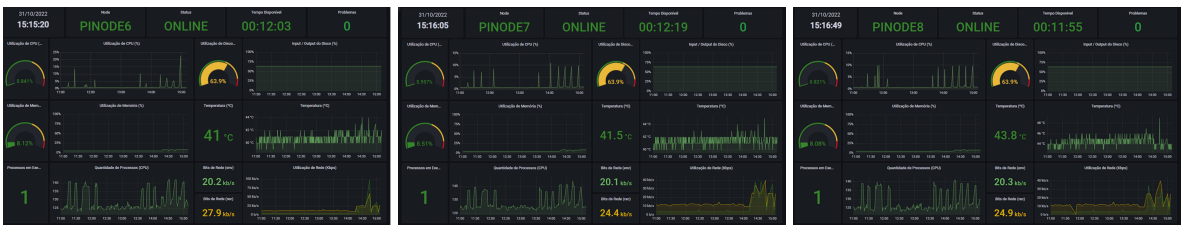

Figura 17 – Comportamento do *cluster* com armazenamento de 16 GB - *Terasort* - Parte 3.

7.1.2.2 Armazenamento de 128 GB

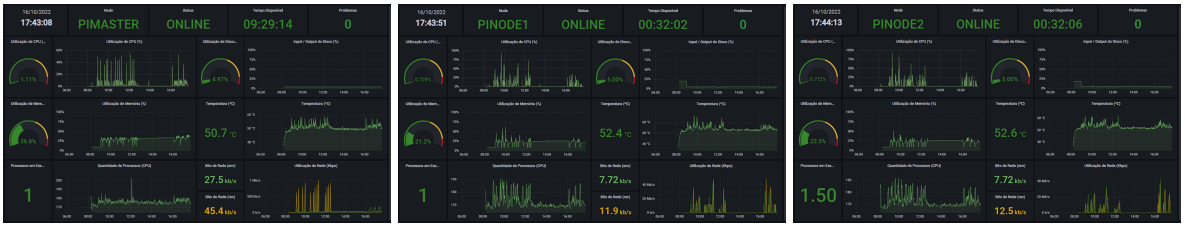

Figura 18 – Comportamento do *cluster* com armazenamento de 128 GB - *Terasort* - Parte 1.

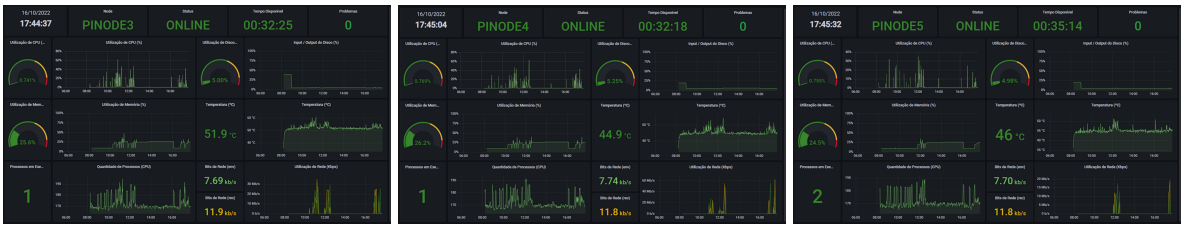

Figura 19 – Comportamento do *cluster* com armazenamento de 128 GB - *Terasort* - Parte 2.

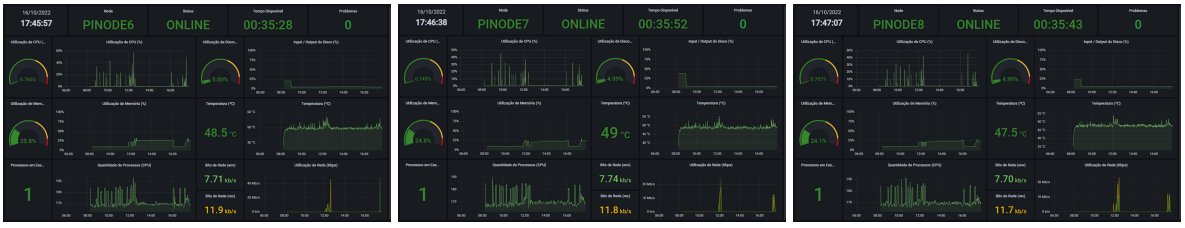

Figura 20 – Comportamento do *cluster* com armazenamento de 128 GB - *Terasort* - Parte 3.

7.2 *Benchmark TestDFSIO*

Nesta seção são apresentados os resultados do *benchmark TestDFSIO*. Para esse cenário de testes, além das configurações citadas no início deste capítulo, os testes de escrita e leitura foram realizados em 2 quantidades de arquivos diferentes: 1 e 10 arquivos respectivamente.

7.2.1 Resultados - *TestDFSIO*

A execução e coleta completa dos resultados do *benchmark TestDFSIO* teve um total de aproximadamente 18 horas de duração, sendo que a primeira bateria de testes demorou cerca de 7 horas (16 GB) e a última aproximadamente 11 horas (128 GB).

Durante a realização dos testes para este *benchmark*, foram coletadas as seguintes métricas: taxa de transferência (*Throughput*), taxa média de escrita/leitura (Média *I/O*), desvio padrão da taxa média de escrita/leitura (Desvio Padrão *I/O*) e tempo total gasto no processamento (Tempo Total).

As Tabelas 8 e 9 mostram a performance do *cluster* durante a bateria de testes para 1 e 10 arquivos, respectivamente. Nos testes envolvendo apenas 1 arquivo, as métricas Taxa Média e Desvio Padrão foram suprimidas, pois seus valores são iguais ao da métrica Taxa de transferência. Uma síntese de forma gráfica desses resultados pode ser visualizada na Figura 21.

O *TestDFSIO* com 1 arquivo (Tabela 8) permite observar que, o *cluster* com capacidade armazenamento de 16 GB, na etapa de escrita, comporta-se de uma maneira que, a medida que o tamanho do arquivo escrito aumenta, o mesmo tem melhor performance com uma maior capacidade de processamento (aumento do número de nós). Isso pode ser observado através da métrica *Throughput*. Entretanto, quando a etapa de leitura foi analisada, o mesmo apresentou um comportamento mais homogêneo. Já quando se analisou o seu comportamento utilizando os cartões de memória de 128 GB, o *cluster* com 2 e 8 nós mostraram capacidades de processamento parecidos em todos os tamanhos dos arquivos escritos, porém com 8 nós o mesmo apresentou uma leve melhora nas taxas de *Throughput*. Para a etapa de leitura, foi visto que o aumento do tamanho do arquivo lido é diretamente proporcional ao aumento da taxa de *Throughput*, fazendo com que ao incrementar poder de processamento (aumento do número de nós) ao *cluster*, são obtidas melhores taxas.

Já para os testes envolvendo a quantidade de 10 arquivos, os dados apresentados na Tabela 9 mostram inicialmente que o *cluster* com menor quantidade de nós e com capacidade de armazenamento de 16 GB não conseguiu realizar todo o processo de testagem, conseguindo apenas processar os arquivos de 250 MB. Isso se dá pelo fato do mesmo chegar ao seu limite total de armazenamento, visto que o mesmo tentará escrever e ler 10 arquivos de 500 MB (5 GB), 750 MB (7,5 GB) e 1 GB (10 GB). Em seguida, é possível observar que, para a etapa de escrita, ainda utilizando os cartões de 16 GB, o *cluster*, na grande maioria dos casos, apresenta uma melhor capacidade de processamento, a medida que, o número de nós é incrementado. Isso pode ser constatado analisando a taxa de *Throughput* juntamente com o tempo total gasto na execução, onde é possível observar que o aumento da taxa de transferência é diretamente proporcional a diminuição do tempo necessário para realizar a tarefa. Também pode ser visto que o *cluster* com 4 nós necessitou de aproximadamente 3 vezes mais tempo do que o *cluster* com 8 nós para finalizar a escrita dos 10 arquivos de 1 GB, corroborando a necessidade de incremento da capacidade de processamento do mesmo (aumento do número de nós). Para a etapa de leitura, foi visualizado que o aumento do tamanho do arquivo lido é diretamente relacionado ao aumento da taxa de *Throughput*, onde incrementando o número de nós faz com que o *cluster* tenha um melhor desempenho.

Quando se analisou os dados do *cluster* utilizando os cartões de memória de 128 GB, percebeu-se um comportamento semelhante aos testes com os cartões de 16 GB, onde a medida que a sua capacidade de processamento aumenta (número de nós), o *cluster* tem melhor desempenho, na grande maioria dos casos analisados. Isso pode ser visto como relação ao tempo necessário para realizar as tarefas de escrita e leitura executadas pelo *TestDFSIO*. O melhor

desempenho dos cartões com maior capacidade de armazenamento remete a tecnologia utilizada por eles, de forma análoga a que se foi analisada para o *Terasort* (Seção 7.1.1), onde, a tecnologia utilizada nos cartões de memória de 128 GB possui uma maior velocidade de gravação e leitura (*I/O*) e permite a gravação de arquivos muito grandes, diferentemente da tecnologia utilizada nos cartões de memória de 16 GB (Kingston Technology, 2020).

Após todas às análises, com relação ao *benchmark TestDFSIO*, não é possível dizer com certeza qual é o melhor cenário para este tipo de teste, mas de acordo com os dados apresentados nas Tabelas 8 e 9, o desempenho do *cluster* é um pouco melhor em um cenário com maior número de nós envolvidos no processamento. Isso provavelmente ocorreu devido a uma melhor distribuição das tarefas de processamento entre os nós do *cluster* neste cenário, fazendo com que a taxa média de entrada e saída de dados diminuísse, além do tempo total gasto para o processamento dessas tarefas.

Em paralelo à realização do *benchmarking* do *cluster* utilizando o *TestDFSIO*, foram coletadas informações inerentes ao comportamento do mesmo, através de um monitoramento ativo. A Seção 7.2.2 explana melhor todo esse processo de monitoramento.

Tabela 8 – Resultados usando o *benchmark TestDFSIO* - 1 Arquivo.

Capacidade	Nós	Operação	Tamanho Arquivo (MB)	*Throughput* (MB/s)	Tempo Total (s)
16 GB	2	Escrita	250	18,15	97,50
			500	2,68	221,40
			750	9,30	115,15
			1000	10,39	135,18
		Leitura	250	103,61	37,94
			500	117,90	44,11
			750	105,37	42,51
			1000	118,81	44,13
	4	Escrita	250	16,02	75,15
			500	13,30	73,80
			750	13,63	91,76
			1000	10,75	147,16
		Leitura	250	112,06	41,19
			500	106,95	41,58
			750	106,96	43,81
			1000	109,39	43,95
	8	Escrita	250	7,00	100,13
			500	5,35	133,98
			750	3,21	269,16
			1000	12,03	122,90
		Leitura	250	108,93	42,43
			500	90,66	48,64
			750	107,17	42,51
			1000	120,61	45,34
128 GB	2	Escrita	250	41,91	67,23
			500	29,98	56,75
			750	24,37	73,71
			1000	26,59	79,94
		Leitura	250	173,25	41,47
			500	213,86	42,15
			750	253,38	42,37
			1000	270,20	43,33
	4	Escrita	250	22,57	62,35
			500	21,73	69,69
			750	41,29	63,58
			1000	25,99	84,92
		Leitura	250	102,88	45,74
			500	109,00	42,50
			750	142,29	46,42
			1000	149,50	46,33
	8	Escrita	250	42,37	60,67
			500	32,55	59,81
			750	37,24	73,69
			1000	30,56	79,03
		Leitura	250	88,43	46,67
			500	101,07	52,70
			750	114,50	51,48
			1000	236,07	44,39

Tabela 9 – Resultados usando o *benchmark TestDFSIO* - 10 Arquivos.

Capacidade	Nós	Operação	Tamanho Arquivo (MB)	*Throughput* (MB/s)	Média *I/O* (MB/s)	Desvio Padrão *I/O*	Tempo Total (s)
16 GB	2	Escrita	250	2,86	5,04	3,73	610,78
			500	x	x	x	x
			750	x	x	x	x
			1000	x	x	x	x
		Leitura	250	33,82	39,75	16,43	57,18
			500	x	x	x	x
			750	x	x	x	x
			1000	x	x	x	x
	4	Escrita	250	2,31	3,11	2,21	188,34
			500	4,07	4,28	1,1	182,67
			750	2,81	2,9	0,48	404,25
			1000	6,58	8,2	3,91	1185,77
		Leitura	250	51,77	59,23	20,88	40,29
			500	58,41	65,78	22,16	50,25
			750	30,85	33,7	10,14	72,58
			1000	46,63	60,44	30,28	61,15
	8	Escrita	250	8,68	15,91	7,4	251,25
			500	9,25	10,13	2,95	432,56
			750	2,91	3,43	1,35	364,74
			1000	4,19	4,76	1,46	476,19
		Leitura	250	34,95	42,81	18,81	49,88
			500	37,53	48,2	27,25	62,76
			750	27,7	33,07	15,47	74,32
			1000	24,76	30,64	15,93	105,36
128 GB	2	Escrita	250	3,58	4,34	2,07	208,05
			500	2,61	2,62	0,14	276,77
			750	3,66	4,83	3,19	466,53
			1000	2,39	2,39	0,05	503,39
		Leitura	250	7,87	9,44	4,74	95,29
			500	8,22	8,51	1,63	123,53
			750	7,99	8,33	1,74	162,44
			1000	7,61	11,59	8,20	249,85
	4	Escrita	250	4,69	9,48	7,86	190,43
			500	2,94	2,98	0,37	247,28
			750	2,65	2,67	0,21	356,70
			1000	3,68	4,89	2,58	417,23
		Leitura	250	18,04	54,49	62,01	86,27
			500	12,88	15,23	6,86	122,42
			750	11,4	19,24	13,56	182,06
			1000	9,83	10,55	3,29	185,00
	8	Escrita	250	6,7	8,13	3,78	116,41
			500	4,05	4,17	0,75	217,39
			750	5,58	5,81	1,23	216,88
			1000	4,58	4,86	1,22	347,93
		Leitura	250	48,21	50,96	12,75	58,78
			500	41,68	49,99	26,15	64,82
			750	31,07	34,07	9,25	83,04
			1000	29,35	32,35	11,79	87,96

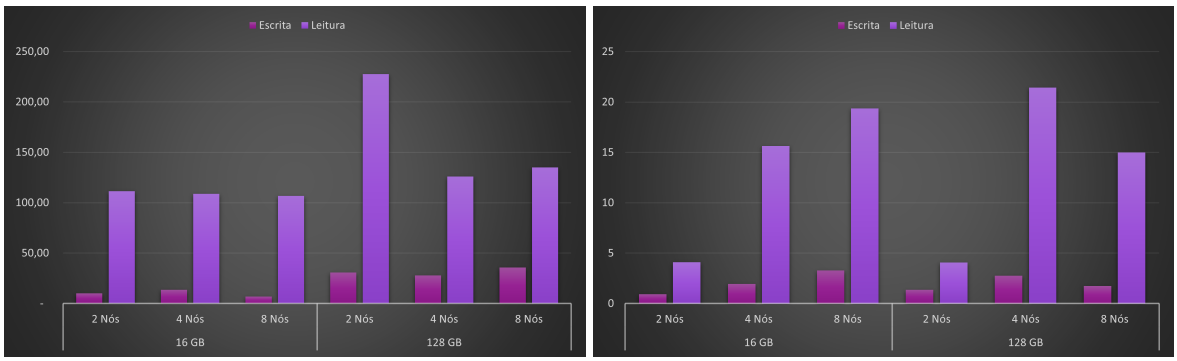

Figura 21 – Síntese dos resultados do *benchmark TestDFSIO*.

7.2.2 Monitoramento - *TestDFSIO*

O monitoramento dos recursos do *cluster* durante o processo de *benchmarking* demonstrado na subseção anterior são apresentados nesta subseção. O mesmo está dividido em dois cenários de forma análoga ao processo de testagem do *cluster*: o primeiro com os nós com capacidade de armazenamento de 16 GB e o segundo com o *cluster* utilizando os cartões de memória de 128 GB em cada um dos seus nós.

Todas as métricas coletadas durante o processo de *benchmarking* utilizando o *TestDFSIO* podem ser vistas nas Figuras 22, 23, 24, 25, 26, 27, sendo as 3 primeiras para o uso dos cartões de 16 GB e as 3 últimas com armazenamento de 128 GB.

De maneira semelhante a que se foi observada durante a execução do *Terasort* (Seção 7.1.2), nota-se a grande quantidade processos que são executados no *cluster*, tendo picos variando entre 145 e 160 processos para o uso dos cartões de memória de 16 GB e tendo uma variância entre 185 e 195 processos em execução para a utilização dos cartões de 128 GB. Essa grande quantidade de processos necessita de um maior esforço para processa-los, sendo que esse esforço pode ser visto através da análise dos dados de utilização do processador (CPU), mostrando aproximadamente 100% de uso. Além disso, essa maior necessidade de poder de processamento requer também um maior consumo energético, onde é possível observar esse aumento através dos dados de temperatura dos dispositivos, na qual os nós analisados, em modo ocioso, trabalham em uma temperatura que varia entre 42 $°C$ e 45 $°C$ e quando requisitados, há um aumento da temperatura, chegando a picos de até 60 $°C$.

Ao se analisar a utilização da memória RAM durante o uso dos cartões de memória de 16 GB, foi visto que há um consumo de memória RAM com picos de uso de mais de 50% quando o *cluster* opera com até 4 nós. Quando o mesmo trabalha com 8 nós, o uso de RAM passa a ser constante, operando em um valor de abaixo de 10%. Ao utilizar os cartões de 128 GB, o *cluster* apresenta um comportamento semelhante, porém há picos de uso de memória RAM de aproximadamente 75% quando opera com até 4 nós.

Quando se analisou o uso de disco e a utilização da rede, em ambos cenários, o *cluster* obteve um comportamento muito parecido ao que foi analisado durante o processo de *benchamarking* utilizando o *Terasort* (Seção 7.1.2).

7.2.2.1 Armazenamento de 16 GB

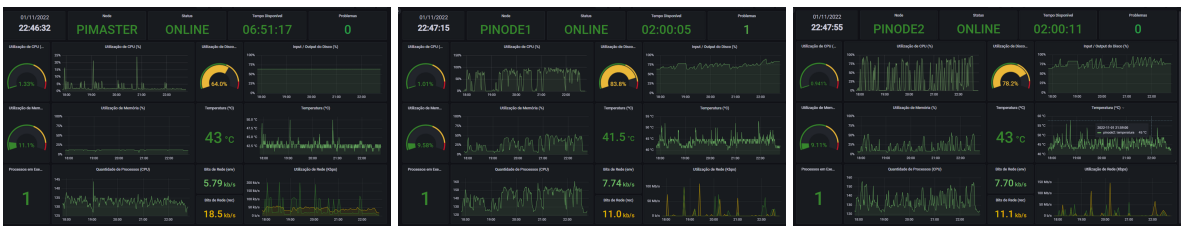

Figura 22 – Comportamento do *cluster* com armazenamento de 16 GB - *TestDFSIO* - Parte 1.

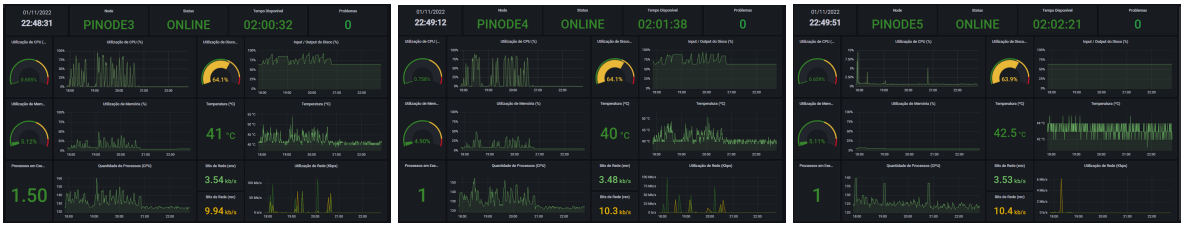

Figura 23 – Comportamento do *cluster* com armazenamento de 16 GB - *TestDFSIO* - Parte 2.

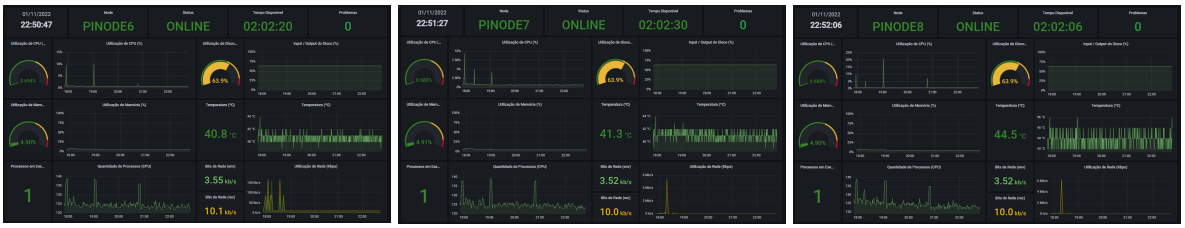

Figura 24 – Comportamento do *cluster* com armazenamento de 16 GB - *TestDFSIO* - Parte 3.

7.2.2.2 Armazenamento de 128 GB

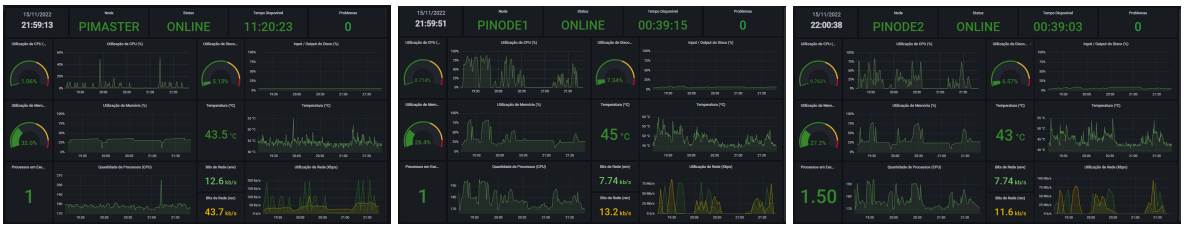

Figura 25 – Comportamento do *cluster* com armazenamento de 128 GB - *TestDFSIO* - Parte 1.

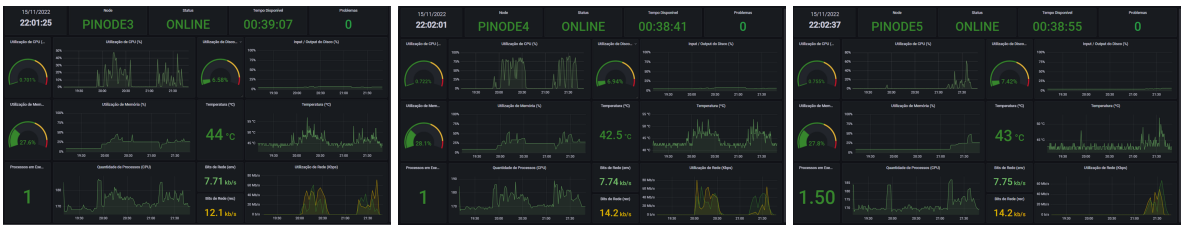

Figura 26 – Comportamento do *cluster* com armazenamento de 128 GB - *TestDFSIO* - Parte 2.

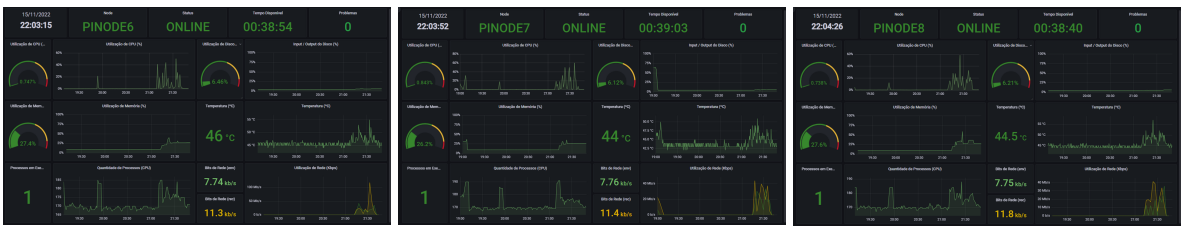

Figura 27 – Comportamento do *cluster* com armazenamento de 128 GB - *TestDFSIO* - Parte 3.

8 Considerações Finais

Este capítulo apresenta, em 4 seções, as considerações finais desta dissertação. O mesmo discorre, na Seção 8.1, a conclusão deste trabalho. A seguir, na Seção 8.2, são mostradas as limitações encontradas durante o desenvolvimento desta dissertação. a Seção 8.3 apresenta as contribuições acadêmicas, oriundas do desenvolvimento desta pesquisa. E, por fim, na Seção 8.4 é mostrado o que se almeja desenvolver em possíveis trabalhos futuros, usando esta dissertação como base.

8.1 Conclusão

Este trabalho apresentou o desenvolvimento de um *cluster big data* de baixo custo, seu processo de *benchmarking* e monitoramento de recursos, motivado pela necessidade de se obter computação de alta performance (HPC) de forma mais econômica do que a fornecida através do uso de supercomputadores para o processamento de *Big Data*. Também disponibilizou um modelo de passo a passo de como desenvolver e configurar o *cluster* do início ao fim, ajudando à comunidade e facilitando a montagem e utilização desses tipos de *clusters*.

O *cluster* desenvolvido é composto por 9 *Raspberry Pis 4B*, sendo 1 trabalhando como mestre, e os demais como escravos. Além disso, 1 *Raspberry Pi 3B+* foi utilizado como servidor de monitoramento dos recursos do *cluster*, utilizando as ferramentas *Zabbix* para a coleta das informações, e o *Grafana*, para disponibilizar as informações coletadas em forma de *dashboards*.

Para a testagem e validação do *cluster* desenvolvido, foram executados dois dos principais *benchmarks* da comunidade *Big Data*: *Terasort* e *TestDFSIO*. Durante o processo de *benchmarking* do *cluster*, o mesmo operou com 3 configurações diferentes, tendo sempre 1 nó mestre e mudando o número de nós escravos para 2, 4, e 8 nós. Além disso, todo esse processo de testagem e validação foi feito com 2 capacidades de armazenamento diferentes: a 1ª, em que cada nó do *cluster* utilizou um cartão de memória de 16 GB, e na 2ª, cada nó estava com um cartão de

memória de 128 GB. Ademais, houve também uma variação entre o tamanho dos arquivos testados (250 MB, 500 MB, 750 MB e 1 GB), tendo seu limite máximo de 1 GB devido à limitação dos cartões de memória de 16 GB de armazenamento.

Os resultados do *benchmark Terasort* foram obtidos em um processo de aproximadamente 8 horas e mostraram que quanto maior é o tamanho do arquivo processado, faz-se necessário uma maior capacidade de processamento no *cluster*, mostrando que quanto mais nós no *cluster*, melhor será o seu desempenho no processamento das tarefas. Com relação ao *benchmark TestDFSIO*, foram necessárias cerca de 18 horas de duração dos testes e os resultados apontaram que não foi possível dizer com certeza qual é o melhor cenário para este tipo de teste, mas, de acordo com os dados apresentados, o desempenho do *cluster* é um pouco melhor em um cenário com maior número de nós envolvidos no processamento.

Foi visto também que o uso de um cartão de memória com maior capacidade de armazenamento e com uma melhor tecnologia que permita uma maior velocidade de escrita/leitura (*I/O*) e gravação de arquivos grandes tem influência direta no desempenho do *cluster*. Deste modo, utilizando um SBC, como é a *Raspberry Pi*, combinado com o *Apache Hadoop*, pode se obter uma solução de *cluster big data* robusta e eficiente, levando-se em consideração o seu custo-benefício, como foi monitorado nessa testagem.

O servidor de monitoramento permitiu analisar o comportamento dos recursos do *cluster* durante todo o processo de *benchmarking*. Ao se analisar os *dashboards* desenvolvidos, foi possível observar os consumos dos recursos de hardware, tais como: uso de CPU, memória usada, estatísticas de tráfego de rede, temperatura do nó, uso de armazenamento de disco, quantidade de processos em execução, etc. Toda essa gama de informações permite se analisar os impactos ocasionados na variação das configurações de quantidade de nós utilizados no *cluster*, bem como no tamanho dos arquivos processados pelo mesmo, além de auxiliar na sua manutenção.

8.2 Limitações Encontradas

Durante o desenvolvimento deste trabalho, surgiram algumas dificuldades que impossibilitaram realizar algumas tarefas que estavam previstas na concepção desta dissertação, porém não puderam ser realizadas por algum motivo. Algumas das principais limitações foram:

- Limite máximo nos arquivos testados — Por conta da limitação na capacidade de armazenamento dos cartões de memória de 16 GB, o tamanho máximo dos arquivos gerados foram de 1 GB;
- Aumento do número de nós do *cluster* — Pretendia-se aumentar ainda mais a quantidade de nós do *cluster*, adquirindo mais *Raspberry Pis* para tal. Porém, devido a alta do dólar

para adquiri-los do exterior e com seu preço no mercado nacional ultrapassar o valor unitário de R$ 1.000,00 [1], ficou inviável a escalabilidade do *cluster*;

- Medição do consumo energético — De forma análoga à limitação anterior, almejava-se medir o consumo energético de cada nó do *cluster*, assim como a medição do consumo do mesmo por completo. Porém, pelo custo elevado da compra de 10 sensores individuais[2] para cada *Raspberry Pi*, além de um medidor de energia elétrica na alimentação geral do *cluster* (Wattímetro), não foi possível realizar tal tarefa;

8.3 Contribuições Acadêmicas

Esta seção apresenta as principais contribuições acadêmicas oriundas do desenvolvimento deste trabalho. Essas contribuições foram, basicamente, os artigos que serviram como base para o desenvolvimento de alguns capítulos desta dissertação, sendo possíveis encontrar as suas versões completas nos Apêndices deste trabalho. São eles:

- *Low-cost clusters on big data - A systematic study* — Aprovado como *Full Paper* e apresentado na *11ª Euro American Conference on Telematics and Information Systems - EATIS' 22* (Apêndice A);
- *A Systematic Review on Teaching Parallel Programming* — Aprovado como *Short Paper* e apresentado na *11ª Euro American Conference on Telematics and Information Systems - EATIS' 22* (Apêndice B);
- *Clusters Big Data* utilizando *Raspberry Pi* e *Apache Hadoop* - Uma Quasi-Revisão Sistemática da Literatura — Aprovado como *Work in Progress* e apresentado no *XII Brazilian Symposium on Computing Systems Engineering - SBESC 2022* (Apêndice C);
- *The development of a low-cost big data cluster using Apache Hadoop and Raspberry Pi. A complete guide* — Aprovado e publicado no periódico *Computers and Electrical Engineering* (Apêndice D);

8.4 Trabalhos Futuros

Nesta seção, são apresentados as possíveis extensões oriundas desta dissertação. Diante da relevância dos resultados obtidos com o desenvolvimento desta pesquisa, pretende-se dar continuidade na mesma em um futuro próximo. Para isso, foram definidos os seguintes pontos como possíveis trabalhos futuros:

[1] <https://encurtador.com.br/iARZ4>
[2] <http://encurtador.com.br/pKMNO>

- *Benchmarking* do *cluster* com arquivos maiores — Pode ser feita uma nova bateria de testes no *cluster* utilizando apenas os cartões de memória de 128 GB, gerando arquivos maiores dos que os utilizados durante o desenvolvimento desta dissertação, chegando ao seu limite máximo de armazenamento;

- Testagem do *cluster* com outros *benchmarks* — A utilização de outros *benchmarks*, como os que foram encontrados na QRSL (*Wordcount* e *Pi Quasi-Monte Carlo*), pode incrementar ainda mais a relevância desta pesquisa;

- Aplicar o uso do *cluster* em um ambiente "real" — Pretende-se utilizar o *cluster* desenvolvido para o armazenamento de um ou mais tipos de documentos fiscais eletrônicos na Secretaria de Estado da Fazenda do Estado de Sergipe - SEFAZ/SE, a fim de validar seu uso de forma "industrial";

- Aquisição do material para medição do consumo energético — Podem ser adquiridos os materiais necessários para que seja possível medir o consumo energético do *cluster* durante as suas utilizações;

- Instalação do *Apache Spark* — Almeja-se fazer essa instalação no *cluster* e testá-lo de forma análoga ao que foi feito nesta dissertação e realizar uma comparação dos resultados obtidos;

- Disponibilização do *cluster* na Internet — Com isso, pode ser possível a realização de *testbed*, para que outros pesquisadores possam comparar seus resultados com os deste trabalho;

Referências

Apache Software Foundation. *Apache Hadoop*. 2023. <https://hadoop.apache.org>. Online; Acessado em 03/01/2023. Citado 6 vezes nas páginas 9, 18, 23, 24, 25 e 26.

BAROLLI, L. *Advanced Information Networking and Applications: Proceedings of the 35th International Conference on Advanced Information Networking and Applications (AINA-2021), Volume 2*. Springer International Publishing, 2021. (Lecture notes in networks and systems, v. 2). ISBN 9783030750756. Disponível em: <https://books.google.com.br/books?id=gmwrEAAAQBAJ>. Citado na página 16.

BöTHER, M.; RABL, T. Scale-down experiments on tpcx-hs. In: *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*. New York, NY, USA: Association for Computing Machinery, 2021. (BiDEDE '21). ISBN 9781450384650. Disponível em: <https://doi.org/10.1145/3460866.3461774>. Citado 3 vezes nas páginas 18, 38 e 40.

BOURHNANE, S. et al. High-performance computing: A cost effective and energy efficient approach. *Advances in Science, Technology and Engineering Systems*, v. 5, n. 6, p. 1598–1608, 2020. ISSN 24156698. Citado 3 vezes nas páginas 17, 37 e 40.

GANTZ, J.; REINSEL, D. Extracting value from chaos. *IDC iview*, v. 1142, n. 2011, p. 1–12, 2011. Citado 2 vezes nas páginas 17 e 22.

GARTNER. *Definition of Big Data - IT Glossary | Gartner*. 2011. <https://www.gartner.com/en/information-technology/glossary/big-data>. Acessado: 12/05/2021. Citado 2 vezes nas páginas 17 e 22.

GIGER, P.; SRIKUGAN, S.; PERSAUD, B. L. A Raspberry Pi Cluster for Teaching Big-Data Analytics. 2020. Citado 2 vezes nas páginas 18 e 27.

Grafana Labs. *Grafana Documentation*. 2023. Acessado em: 21/01/2023. Disponível em: <https://grafana.com/docs/>. Citado na página 30.

HAJJI, W.; TSO, F. P. Understanding the performance of low power raspberry pi cloud for big data. *Electronics (Switzerland)*, v. 5, n. 2, 2016. ISSN 20799292. Disponível em: <https://doi.org/10.3390/electronics5020029>. Citado 3 vezes nas páginas 18, 36 e 40.

HENNESSY, J. L.; PATTERSON, D. A. *Computer architecture: a quantitative approach*. [S.l.]: Elsevier, 2011. Citado na página 18.

IBM. *The 5 V's of big data*. 2016. <https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/>. Acessado: 20/06/2021. Citado 2 vezes nas páginas 17 e 22.

KANITHAN, S. et al. Smart dustbin using lora and tensorflow network. In: *2021 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C)*. [S.l.: s.n.], 2021. p. 290–296. Citado 2 vezes nas páginas 17 e 27.

KHAN, M. A. ud-din; UDDIN, M.; GUPTA, N. Seven v's of big data understanding big data to extract value. *Proc. of the 2014 Zone 1 Conference of the American Society for Engineering Education*, p. 1–5, 2014. Citado 2 vezes nas páginas 22 e 23.

Kingston Technology. *Um guia para cartões SD e microSD*. 2020. Acessado em: 01/02/2023. Disponível em: <https://www.kingston.com/br/blog/personal-storage/microsd-sd-memory-card-guide>. Citado 2 vezes nas páginas 47 e 55.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele University Technical Report TR/SE-0401*, v. 33, 08 2004. ISSN 1353-7776. Citado 2 vezes nas páginas 9 e 33.

KOMNINOS, A. et al. Performance of raspberry pi microclusters for edge machine learning in tourism. In: *AmI*. [S.l.: s.n.], 2019. Citado 4 vezes nas páginas 9, 29, 38 e 40.

KUSS, F. et al. Aulacast: A single board computer platform to support teaching. In: . [S.l.: s.n.], 2018. p. 366–373. Citado 2 vezes nas páginas 18 e 26.

LEE, E.; OH, H.; PARK, D. Big Data Processing on Single Board Computer Clusters: Exploring Challenges and Possibilities. *IEEE Access*, IEEE, v. 9, p. 142551–142565, 2021. ISSN 21693536. Citado 3 vezes nas páginas 18, 35 e 40.

MUNDE, K.; JAHAN, N. Hadoop architecture and applications. *International Journal of Innovative Research in Science, Engineering and Technology (An ISO*, v. 3297, 2007. Citado na página 24.

NETO, A. A. et al. Clusters big data utilizando raspberry pi e apache hadoop - uma quasi-revisão sistemática da literatura. In: *Anais Estendidos do XII Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2022. p. 92–97. ISSN 2763-9002. Disponível em: <https://sol.sbc.org.br/index.php/sbesc_estendido/article/view/22848>. Citado na página 34.

NETO, A. J. A.; NETO, J. A. C.; MORENO, E. D. The development of a low-cost big data cluster using apache hadoop and raspberry pi. a complete guide. *Computers and Electrical Engineering*, v. 104, p. 108403, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790622006206>. Citado 2 vezes nas páginas 41 e 43.

NETO, A. J. A.; NETO, J. A. C.; ORDONEZ, E. D. M. Low-cost clusters on big data - a systematic study. In: *Proceedings of the 11th Euro American Conference on Telematics and Information Systems*. New York, NY, USA: Association for Computing Machinery, 2022. (EATIS '22). ISBN 9781450397384. Disponível em: <https://doi.org/10.1145/3544538.3544635>. Citado 2 vezes nas páginas 18 e 32.

NETO, J. A. C.; NETO, A. J. A.; MORENO, E. D. A systematic review on teaching parallel programming. In: *Proceedings of the 11th Euro American Conference on Telematics and Information Systems*. New York, NY, USA: Association for Computing Machinery, 2022. (EATIS '22). ISBN 9781450397384. Disponível em: <https://doi.org/10.1145/3544538.3544659>. Citado na página 18.

OJOKOH, B. A. et al. Big data, analytics and artificial intelligence for sustainability. *Scientific African*, v. 9, p. e00551, 2020. ISSN 2468-2276. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2468227620302891>. Citado na página 23.

ORACLE. *O que é computação de alto desempenho (HPC)?* 2023. <https://www.oracle.com/br/cloud/hpc/what-is-hpc/>. Acessado em: 17/01/2023. Citado 3 vezes nas páginas 17, 27 e 28.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Swindon, UK: BCS Learning & Development Ltd., 2008. (EASE'08), p. 68–77. Citado 2 vezes nas páginas 9 e 31.

QURESHI, B.; KOUBAA, A. On performance of commodity single board computer-based clusters: A big data perspective. *EAI/Springer Innovations in Communication and Computing*, p. 349–375, 2020. ISSN 25228609. Disponível em: <https://doi.org/10.1007/978-3-030-13705-2_15>. Citado 3 vezes nas páginas 9, 18 e 27.

RAMOS, M. J. O que é a computação de alto desempenho? *Revista de Ciência Elementar*, ICETA, v. 9, n. 4, dez. 2021. Disponível em: <https://doi.org/10.24927/rce2021.070>. Citado na página 28.

RAYALA, V.; KALLI, S. R. Big data clustering using Improvised Fuzzy C-Means clustering. *Revue d'Intelligence Artificielle*, v. 34, n. 6, p. 701–708, 2021. ISSN 19585748. Citado na página 16.

SARALADEVI, B. et al. Big Data and Hadoop-a Study in Security Perspective. *Procedia Computer Science*, v. 50, p. 596–601, 2015. ISSN 1877-0509. Disponível em: <https://www.sciencedirect.com/science/article/pii/S187705091500592X>. Citado na página 24.

SCOLATI, R. et al. A containerized big data streaming architecture for edge cloud computing on clustered single-board devices. *CLOSER 2019 - Proceedings of the 9th International Conference on Cloud Computing and Services Science*, n. May, p. 68–80, 2019. Citado 2 vezes nas páginas 37 e 40.

SCOLATI, R. et al. A Containerized Edge Cloud Architecture for Data Stream Processing. *Communications in Computer and Information Science*, v. 1218 CCIS, n. May, p. 150–176, 2020. ISSN 18650937. Citado 2 vezes nas páginas 37 e 40.

SRINIVASAN, K. et al. An efficient implementation of mobile Raspberry Pi Hadoop clusters for Robust and Augmented computing performance. *Journal of Information Processing Systems*, v. 14, n. 4, p. 989–1009, 2018. ISSN 2092805X. Citado 3 vezes nas páginas 18, 37 e 40.

TAY, Y. C. Data generation for application-specific benchmarking. *Proc. VLDB Endow.*, VLDB Endowment, v. 4, n. 12, p. 1470–1473, aug 2011. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3402755.3402798>. Citado na página 25.

TRICKDOID. *Compreendendo o Big Data com Benefícios, Características e Exemplos de Aplicativos de Big Data*. 2021. <https://trickdroid.org/compreendendo-o-big-data-com-beneficios-caracteristicas-e-exemplos-de-aplicativos-de-big-data/>. Acessado: 17/11/2021. Citado 2 vezes nas páginas 9 e 23.

TURANA, J. S.; SUKOCO, H.; KUSUMA, W. A. Hadoop performance analysis on raspberry pi for dna sequence alignment. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, v. 14, n. 3, p. 1059–1066, 2016. Citado 2 vezes nas páginas 36 e 40.

VIRTANA. *What are clusters - Virtana Glossary*. 2022. <https://www.virtana.com/glossary/what-is-a-cluster/>. Acessado em: 17/12/2022. Citado na página 28.

Zabbix Company. *What is Zabbix*. 2023. <https://www.zabbix.com/documentation/6.0/en/manual/introduction/about>. Online; Acessado em 21/01/2023. Citado 2 vezes nas páginas 28 e 29.

Apêndices

APÊNDICE A – *Low-cost clusters on big data - A systematic study*

Low-cost clusters on big data - A systematic study

Antônio José Alves Neto
antonio.neto@dcomp.ufs.br
Federal University of Sergipe
São Cristovão, Sergipe, BRA

José Aprigio Carneiro Neto
jose.neto@ifs.edu.br
Federal Institute of Sergipe
Itabaiana, Sergipe, BRA

Edward David Moreno Ordonez
edwdavid@gmail.com
Federal University of Sergipe
São Cristovão, Sergipe, BRA

ABSTRACT

The growing gap between users and the Big Data analytics requires innovative tools that address the challenges faced by big data such as volume, variety, and velocity. Therefore, it becomes computationally inefficient to analyze this massive volume of data. Moreover, advancements in the field of Big Data applications and data science poses additional challenges, where High-Performance Computing (HPC) solution has become a key issue and has attracted attention in recent years.

Because of the high costs to obtain a HPC, the researchers are looking for a solution that copes with the increasing demand on processing power due to the expanding amount of the data produced. Eventually, they have been trying to implement big data clusters based on low-cost and low-energy hardware.

The goal of this paper is to identify how is possible to develop a big data cluster using hardware structures of low cost, exposing the studies found in literature. In order to fulfill this, a Systematic Literature Mapping (SLM) was realized, resulting in several relevant papers which are able to response three research questions. The SLM identified Single Board Computers (SBC) as hardware structure most used, being the Raspberry PI with major citations, and Apache Hadoop and Apache Spark as big data platforms used in these clusters. The validation of these clusters it was done with some popular algorithms, where the algorithms K-Means and Map-Reduce were the most quoted in the selected studies, this last based on its original approach, unlike the approach from Hadoop Map-Reduce.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Distributed programming languages**.

KEYWORDS

Beowulf, Big Data, Cluster, Low Cost, Raspberry Pi, Single Board Card, Systematic Literature Mapping

ACM Reference Format:
Antônio José Alves Neto, José Aprigio Carneiro Neto, and Edward David Moreno Ordonez. 2022. Low-cost clusters on big data - A systematic study. In *11 Euro American Conference on Telematics and Information Systems (EATIS2022), June 1–3, 2022, Aveiro, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3544538.3544635

EATIS2022, June 1–3, 2022, Aveiro, Portugal

ACM ISBN 978-1-4503-9738-4/22/06…$15.00
https://doi.org/10.1145/3544538.3544635

1 INTRODUCTION

Nowadays, a large amount of data has been generated every day from various sources such as social media, satellites, sensors, mobile devices, computer simulations and business transaction. This data produces valuable information useful for business intelligence, forecasting, decision support, intensive data research. For example, Walmart has nearly 2.5 petabytes and Facebook stores nearly 30 petabytes of data, such huge data is known as Big Data [27].

The growing gap between users and the Big Data analytics requires innovative tools that address the challenges faced by big data volume, variety, and velocity. Therefore, it becomes traditional computing can be inefficient to analyze such massive volume of data. Moreover, advancements in the field of Big Data application and data science poses additional challenges, where High-Performance Computing (HPC) solution has become a key issue and has attracted attention in recent years [1].

HPC refers to the computing system, including several processors as part of a single machine or a cluster of several computers as an individual resource. HPC owes its feature of high speed computing to its great ability to process information. Therefore the main methodology that is currently applied to HPC is parallel computing. In short, HPC is legendary for its processing capacity [20].

Throughout the years, HPC has been provided via either supercomputers, or cluster of commodity computers. The first venue is no longer opted for mainly because of its high cost and not-so-easy maintenance which leaves the clustering as the perfect alternative [4].

Because of the high costs to obtain a HPC [9], the researchers are looking for solutions that can cope with the increasing demand on processing power due to the expanding amount of the data produced. Eventually, researchers have been trying to implement big data clusters based on low-cost and low-energy hardware [22], [12]. There is a significant amount of work that has been carried in this direction. Most of it used single board computers as a basis of the clusters and tried to prove that the hardware supports the installation of different big data's analytics platforms [4].

This paper aims to identify how is possible to develop a big data cluster using structures of low cost, exposing the studies found in literature papers. In order to fulfill, the Systematic Literature Mapping (SLM) was chosen as research method.

This study is organized as follows: In the Section 2, we present some necessary prior knowledge about the theme of this study. The Section 3 defines the SLM protocol used in this study. The Section 4 shows the results obtained through the research process, filter

and analysis of the selected papers. The threats to validity of this work are presented in the Section 5. Finally, Section 6 presents the obtained results are discussed and suggestions of future works and in the Section 6 are made the acknowledgements.

2 BACKGROUND

In this section, some necessary prior knowledge about the theme of this study are discussed.

2.1 Big Data

The definition of “Big Data” is complex and constantly changing. However, there is some consensus in the literature on the main characteristics of Big Data as described by a widely cited Gartner report [11]: "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."

Similarly, Gantz & Reinsel defined big data using four V’s (Volume, Variety, Velocity, and Value) in 2011 [10]. In 2012, the characteristic "Veracity" was introduced as a fifth characteristic of big data [13]. While many other V’s exist [10], in this paper we focus on the five most common characteristics of big data, as illustrated in Figure 1.

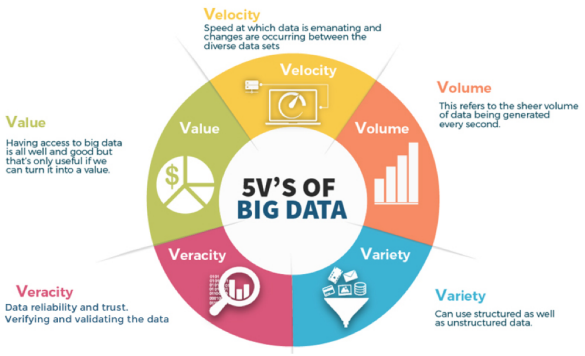

Figure 1: 5V’s of Big Data [33]

"Volume" it is the large amount of data generated daily around the world, coming from various kind of sources as audios, videos, messages, social media, etc [34]. "Velocity" refers how fast the data is generated, stored and analyzed to enable a better decision making. This increasing can be observed in real time in Internet Stats website. "Variety", as well as the volume, the data variety is very large. There are structured data, in sequential files, databases and non-data structured, which are the contents coming from different kind of sources. "Value" - one of the most important features of Big Data is related to the aggregate value obtained in decision making based on previously analysed data by the organizations [34]. Finally, "Veracity" dealing with a high volume, velocity and variety of data, it not possible to grantee a 100% of correct information. The quality of data can be vary. The data accuracy of analysis depends on the veracity of the source data [21].

2.2 Single Board Computer (SBC)

The popularization of smartphones has allowed a great increase on the industry of systems on chips (SoC) of low cost, high process capacity and interaction with many kinds of sensors [19]. A Single Board Computer (SBC) is a kind of computer implemented on a printed single board that uses the SoC’s capacities in its modeling. It also allows data storage in an external unit and offers several interfaces to interact with other electronic devices [18].

Constructed as a tool to support programming learning, it became one of the main references on developing tool prototypes that requires processing power similar to traditional computers.

Figure 2: Raspberry PI - A kind of SBC. Adapted from Qureshi and Koubba [26].

3 METHODOLOGY

A SLM is an evidence-based form of secondary study that provides a comprehensive overview of a research area, identifying common publication venue types (e.g. conference or journal), quantitative analyses (e.g. number of published studies per year), and research findings in the investigated research area [24]. Figure 3 details the SLM construction process.

A SLM offers several benefits, such as identifying gaps and clusters of papers, based on frequently occurring themes in a current research, by using a systematic and objective procedure [17]; helping plan a new research, avoiding duplicated effort [6]; and also identifying topics and areas for future systematic literature reviews [32].

For making the SLM, in this paper we used the methodologies suggested by Kitcheman et. al [17] and Petersen et al. [25]. Following the protocol developed by these authors, it was realized a search aiming primary studies about the development and validation of low cost big data cluster. The analysis and discussion about the found papers was done by the authors of this paper.

3.1 Research Questions (RQs)

To accomplish the paper’s goal we define the following main question (MQ) which guides this study: “**How to get a low cost big data cluster and how to validate it?**”. In order to answer this Main Question (MQ), the three following research questions were raised, described as follows:

RQ1: Which are the hardware structure used to achieve a low cost cluster?
RQ2: Which are the big data platforms used in these clusters?

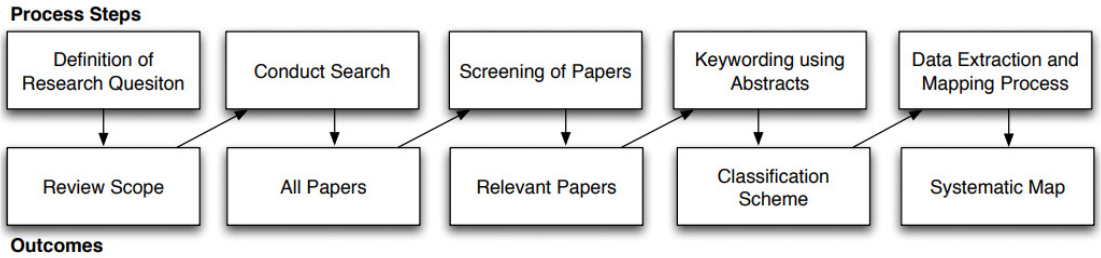

Figure 3: The SLM Process [24].

RQ3: Which algorithms are used to validate the big data clusters cited in these studies?

3.2 Search Strategy

A search strategy must be developed and implemented to find studies that potentially can answer the MQ and consequently the RQs.

The query string used to perform the searches was developed based at PICO (Population, Intervention, Comparison and Outcomes) Model [25], [16]. The PICO Model aims to highlight the effects of an intervention on a population and a control environment is used to validate the results. Following this model, the keywords used to create the search string used in this SLM can be observed in Table 1. In this study there is no comparison environment. Therefore, the comparison term was not used.

Table 1: PICO Model Structure.

PICO	Keyword
Population	cluster, clusters
Intervention	big data
Comparison	-
Outcome	low cost, low costs

According to Pai et al [23]. the elements of PICO Model can be related by the AND logical operator. Since comparison (C) is not applicable to this study, this was done for the set of keywords chosen to represent population (P), intervention (I) and outcome (O), resulting in the following structure: (P) AND (I) AND (O). Within each of these three elements of the structure, their keywords were combined with the OR operator (Figure 4).

Using the search string, the authors performed searches using the query string presented in Figure 4, in the following the main two academic paper search engines with the purpose of finding relevant papers about low cost clusters big data context: Scopus and Web of Science. These two databases were chosen due to their coverage, containing works of main databases such as IEEE, ACM and Springer Link, etc. It is important to highlight that all the databases were accessed through CAPES Journals Portal [5].

Each database has its own particularities, in order to achieve a similar result, the query string had to be adapted to each search engine.

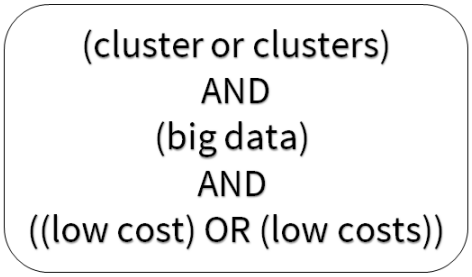

Figure 4: Query String.

3.3 Inclusion and Exclusion Criteria

Inclusion and exclusion criteria are used to exclude studies that are not relevant to answer the research questions [24]. The inclusion criteria used to define the papers that would be used in this mapping were the following:

IC1: Publications focused on low cost big data cluster;
IC2: Publications that contribute to ask at least one of these RQs;
IC3: Papers published between 2015 and 2021;
IC4: Papers written in English or Portuguese language;

And the papers that would be excluded were the following:

EC1: Gray literature;
EC2: Inconsistent studies with the research topic;
EC3: Duplicate papers;
EC4: Papers not available;

3.4 Paper Selection

The SLM conduction followed the protocol introduced in previous subsections and can be observed in Figure 5. The selection and the information extraction phases were performed with Mendeley tool support [8].

Executing the query described in Section 3.2, 175 papers were found, of this number 49 duplicated papers were removed and after applying the exclude criteria described in Section 3.3, 90 were discarded, resulting in 36 selected for full reading. After a full read of these 36 studies, it was classified 14 papers as relevant to realize this SLM.

4 ANALYSIS AND RESULTS

In this section are presented the results of SLM. The data extraction results are exposed, analyzed and discussed.

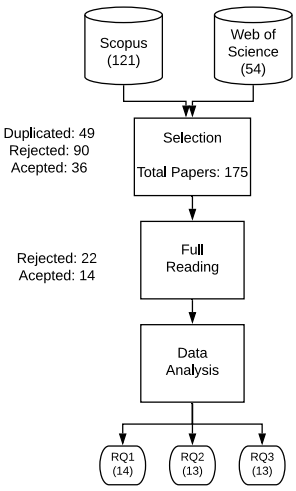

Figure 5: Data Selection Process.

4.1 Preliminary Results

Some additional information that was judged as a secondary information, but relevant to better understanding of this study.

The Figure 6 presents a comparative between the pre-selected papers and the selected over the years. The years with major publications incidence were 2017 and 2020 and coincidentally the years with more papers selected. An important point to highlight is the number of citations, between the selected papers, for Haiji and Tso [12] with 27 citations.

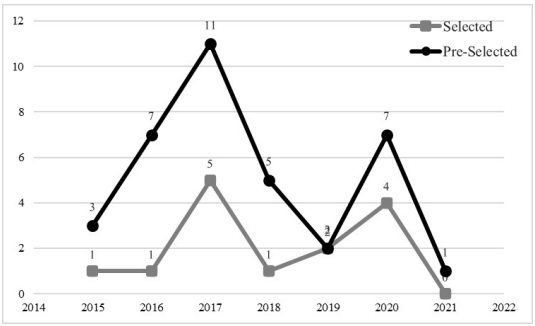

Figure 6: Published Papers Amount per Year.

In the Figure 7, it is possible to observe observe the major participation in publications about low cost big data cluster between Asia and Europe continents.

4.2 Analysis of Selected Studies

Haiji and Tso [12] present a set of experiments to test the performance of a single node and a cluster of 12 Raspberry Pi 2 boards with and without environment virtualization using Docker to study its feasibility for real-time big data analytics. They evaluated their performance and concluding that in a virtualized environment, CPU

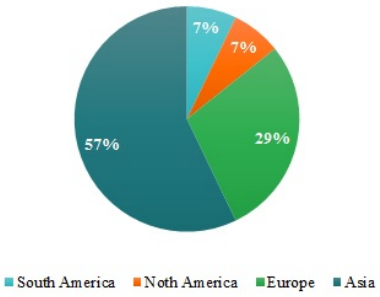

Figure 7: Distribution of low cost big data cluster publication through the continents (in percentage).

and memory consumption becomes higher, network throughput decreases, and burstiness occurs less often and less intensively.

Srinivasan et al. [31] present a performance comparison between a SBC clusters and a single computer. It was used an Apache Hadoop combined with the Raspberry Pi cluster to compute the big data generated via feature point extraction on an image. The MapReduce operation in Hadoop and the salient features of Raspberry Pi 3 helps us perform the experiment more efficiently compared to a high processing single computer. However, the same cannot be said when they deal with a smaller dataset. They concluded that a low cost of the Raspberry Pi clusters and the scalable Hadoop environment can be used for a variety of complex operations in this era of big data.

Saffran et al. [28] performed a study of an unconventional low-cost energy-efficient High Performance Computing (HPC) cluster composed of Raspberry Pi nodes. This cluster was compared with a well-known co-processor used in HPC (Intel Xeon Phi) for data mining algorithms. The results showed that the Raspberry Pi cluster can consume up to 90% less power than Intel Xeon Phi.

In their studies Scolati et al. [30], [29] proposed and implemented a low-power and low-cost cluster of SBC to apply common models and technologies from the big data domain. The authors implemented the system using a cluster of Raspberry Pi and Docker to containerize and deploy an Apache Hadoop and Apache Spark data streaming processing cluster. They evaluated the performance, showing that by using containerization increased fault tolerance and ease of maintenance can be achieved.

Kim and Son [15] show a case study on the feasibility of Raspberry-Pi as a big data cluster in smart factory where the cluster will be composed of more than 3 Raspberry Pi (nodes). They will install Hadoop and use it to share data among nodes in a cluster and in the future they will compare its performance with a Arduino cluster.

Wang et al. [35] developed a big data cluster with Raspberry Pi in order to mitigate the need of leverage low-cost hardware to build the distributed computing test environment for the purpose of the algorithm validation and experiment simulation. They developed a visualized user interface to manage cluster, deploy service containers, and manage workflow and presented the containerized service workflow that links the components of data source, distributed data processing and visualization.

Wu et al. [36] developed a scalable and distributed storage for time series data and made its validation using a low-cost cluster based in a SBC with a comprehensive set of performance measures. It used three (3) kind of SBC: Raspberry Pi, Cubieboard and OrangePi. The results showed that low-cost hardware and open source technologies could be utilized for large-scale time series data storage.

Bharathi et al. [2],[3] present how a distributed and scalable computing facility using a beowulf cluster can be used to provide data storage and analytical capabilities, for exploratory needs of a small or medium scale scientific application.

The papers of Ordonez et al. [22] and Fotache et al. [9] present a low cost architecture based in beowulf cluster to solve machine learning problems in big data analytics. They used some popular machine learning models as well as some graph algorithms and their architecture works well for. Results were positive and encouraging for extending the cluster number of nodes and the database scale.

Qureshi and Koubaa [26] present a comparison between 2 low cost big data cluster based in SBC (Raspberry Pi and Ordrod Xu) and a low cost big data cluster based in beowulf architecture. It was done the evaluation of performance, the viability and cost-effectiveness of these clusters. The results from these studies show that while SBC-based clusters are energy efficient overall, the operation cost to performance ratio can vary based on the workload.

In similar way Kaewkasi and Srisuruk [14] developed a low cost big data cluster using 22 ARM board in order to study the performance in usage of this kind of SBC and made a comparison with a cluster mentioned in their related works, showing a better performance of the developed ARM board cluster.

4.3 RQ1 - Which are the hardware structure used to achieve a low cost cluster?

The SLM presented 6 types of hardware used to achieve a low cost big data cluster. They are ARM board, Beowulf, Cubieboard, Odroid, OrangePi and Raspberry Pi. Excluding Beowulf, all the others hardware structures mentioned are classified as SBC (Section 2).

It is important to note that Raspberry Pi (64,28%) and Beowulf (35,71%) were the hardware structures most cited in the found studies. All these kind of hardware can be observed in Table 2.

Table 2: Type of hardware used to get a low cost big data cluster.

Hardware	Reference
ARM Board	[14]
Beowulf	[9], [22], [26], [2], [3]
Cubieboard	[36]
Odroid	[26]
OrangePi	[36]
Raspberry Pi	[26], [30], [29], [15] , [28], [35], [36] , [12] , [31]

4.4 RQ2 - Which are the big data platforms used in these clusters?

It was found 3 big data platforms in these studies, they are: Apache Hadoop, Apache Spark and Parallel DBMS (Vertica). The Apache Hadoop was mentioned in 11 papers, the Apache Spark in 6 and the Parallel DBMS (Vertica) in just 1. Safran et al. [28] did not specify the big data platform used in their study. Some studies mentioned the usage of more than one big data platform. All these informations can be observed in Table 3.

Table 3: Big Data platforms used in low cost cluster

Big Data Platform	Reference
Apache Hadoop	[26], [30], [29], [15], [35], [36], [2], [3], [12], [14] , [31]
Apache Spark	[22] , [30], [29], [12], [14], [9]
Parallel DBMS	[22]

4.5 RQ3 - Which algorithms are used to validate the big data clusters cited in this studies?

It was identified 10 algorithms used in big data clusters of these studies, being the Map Reduce and K-Means the algorithms most cited with 4 mentions.

It is important to explain that the Map-Reduce(*) algorithm cited in these four papers is based on its original approach, unlike the approach from Hadoop Map-Reduce. Kim and Son [15] don't present no one algorithm because their paper is dedicated to a theoretical aspects.

These informations can be observed in Table 4. Some studies mentions the usage of more than one algorithm. All these information can be observed in Table 4.

Table 4: Algorithms that are used to validate the big data clusters

Algorithm	Reference
Apriori	[28]
CMFW	[14]
Fuzzy C-Means	[2], [28]
K-Means	[22], [2], [28], [3]
Map-Reduce*	[36], [35], [30], [29]
Naive Bayes	[22]
Pi/Quasi-Monte Carlo	[26]
SSR	[22]
SURF	[31]
TPC-H	[9]
WordCount	[26], [12]

4.6 Trends/Towards and Challenges in this research field?

After of analyzing all the 14 selected papers, it was highlighted some important topics regarding to trends, challenges and towards designing and developing of low cost big data clusters. In this

section, the topics that were considered the most important in this research area are presented.

One of the challenges encountered is "resource management". Sometimes, more resources are needed for realize a certain operation. Scolati et al [30], [29] point a possible opportunity is using machine learning to implement autonomous configuration and resource management.

In similar way Kaewkasi and Srisuruk [14] and Footache et al.[9] pretend to use Solid State Drive (SSD) disks to increase the storage capacity and processing speed of their SBCs. These authors suggest to make a data pre-processing structure to optimizing the second level operations speed.

Kaewkasi and Srisuruk et al. [14] and Saffran et al.[28] propose the usage of new Graphics Processing Unit (GPU)-based SBCs. With their highly-parallel and low-watt characteristics, it would be an opportunity to explore a Big Data process with the second-level parallelism provided by these massive coprocessors.

Quereshi and Kouba [26], Kim and Son [15], Wang et al. [35] and Srinivasan et al. [31] suggest the usage of this low-cost big data cluster to academic research, teaching and learning. Due to the high cost of obtaining an HPC and to use these low-cost clusters as an alternative for HPC.

Scolati et at [30], [29] say that this kind of architecture can be applied in Autonomous driving area since they need to coordinate their behaviour, often using modern telecommunications technologies such as 5G mobile networks and require an on-board capability in order to guarantee the low latency requirements. SBC clusters are an example of computational infrastructure.

5 THREATS TO VALIDITY

For a work be acceptable as a contribution to scientific knowledge, the researcher needs to convince the readers that the conclusions drawn from an empirical study are valid [7]. In this section are exposed all threats to validity found in this study.

- To ensure an unbiased papers selection, the complete SLM protocol was defined at the beginning of the process. However, a threat related to the papers quality assessment found that have been included can not be ruled out, since the studies were selected without assigning scores.
- The non-inclusion of relevant papers can be classified as a threat to validity. The predefined include and exclude criteria might have excluded potential relevant papers for this study.
- Limited database number usage may have contributed to the non-inclusion of potential relevant documents.
- The exposure of the whole SLM protocol used in this paper, allows other researchers to replicate this study and obtain similar results.

6 CONCLUSION AND FUTURE WORKS

In this paper, a SLM was performed to identify how to get a low cost big data cluster and how to validate it. In order to achieve this goal, 3 "research questions" were raised. To answer these questions, searches were performed in Scopus and Web of Science databases, returning 175 papers. After application of the inclusion and exclusion criteria, a detailed reading was performed and 14 relevant papers were selected for this study.

Regarding the RQ1, about which are the hardware structure used to achieve a low cost cluster, it was mentioned in 14 studies. The hardware structures used to get a low cost big data cluster found in these studies are the ARM Board, Beowulf, Cubieboard, Odroid, OrangePi and Raspberry Pi, where the Raspberry Pi and Beowulf were the most quoted in literature, with 9 and 6 mentions respectively.

The RQ2 aimed to find which are the big data platforms used in these clusters. From 14 papers selected, the results pointed 3 big data platforms, they are: Apache Hadoop, Apache Spark and Parallel DBMS (Vertica). The Apache Hadoop was mentioned in 11 papers, the Apache Spark in 6 and the Parallel DBMS (Vertica) in only 1. Just one study did not specify the big data platform used and some studies mention the usage of several big data platforms.

In the RQ3, it was intended to identify the algorithms used to validate the big data clusters cited in these studies. It was found 10 (Ten) algorithms which were identified in these studies, being the Map Reduce and K-Means the algorithms most quoted with 4 citations. One of the authors does not present no one algorithm because its paper is a theoretical paper.

This SLM showed that exists the possibility to get a big data cluster using low cost structures. Thus, this research shows relevant results and provides support for a better understanding about the development low cost big data clusters. Hence, this paper can contribute to the development of another studies and tools in big data and low-cost clusters.

As future work, also it is intended to evolve this SLM to a Systematic Literature Review (SLR), increasing the effectiveness of the results obtained. It is intended to make a development of a low cost big data cluster using Raspberry Pi as hardware structure and Apache Hadoop as big data platform and use it in different benchmarks and show and discuss the evaluation performance.

ACKNOWLEDGMENTS

We appreciate all reviewers of this work.

REFERENCES

[1] Awais Ahmad, Anand Paul, Sadia Din, M Mazhar Rathore, Gyu Sang Choi, and Gwanggil Jeon. 2018. Multilevel Data Processing Using Parallel Algorithms for Analyzing Big Data in High-Performance Computing. *International Journal of Parallel Programming* 46, 3 (2018), 508–527. https://doi.org/10.1007/s10766-017-0498-x

[2] Reena Bharathi, Shailaja Shirwaikar, Vilas Kharat, and Gajanan Aher. 2017. A cloud-based data analytical framework for medium scale scientific applications. In *Proceedings of the International Conferences on Computer Graphics, Visualization, Computer Vision and Image Processing 2017 and Big Data Analytics, Data Mining and Computational Intelligence 2017 - Part of the Multi Conference on Computer Science and Information Systems 2017*. 213–222. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85040182744&partnerID=40&md5=ff27499f32db22f0b52ea0d0e8418163

[3] Reena Bharathi, S. C. Shirwaikar, and Vilas Kharat. 2017. A distributed, scalable computing facility for big data analytics in atmospheric physics. *Communications in Computer and Information Science* 721 (2017), 529–540. https://doi.org/10.1007/978-981-10-5427-3_54

[4] Safae Bourhnane, Mohamed Riduan Abid, Khalid Zine-Dine, Najib Elkamoun, and Driss Benhaddou. 2020. High-performance computing: A cost effective and energy efficient approach. *Advances in Science, Technology and Engineering Systems* 5, 6 (2020), 1598–1608. https://doi.org/10.25046/aj0506191

[5] CAPES/MEC. 2022. Portal de Periódicos da Capes. http://www.periodicos.capes.gov.br/.

[6] Paulo Anselmo da Mota Silveira Neto, Ivan do Carmo Machado, John D. McGregor, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2011. A systematic mapping study of software product lines testing. *Information and*

Software Technology 53, 5 (2011), 407 – 423. Special Section on Best Papers from XP2010.

[7] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. *Selecting Empirical Methods for Software Engineering Research.* Springer London, London, 285–311. https://doi.org/10.1007/978-1-84800-044-5_11

[8] Elsevier. 2022. Mendeley Reference Management Software Researcher Network. https://www.mendeley.com/downloads.

[9] Marin Fotache, Marius Iulian Cluci, and Valericã Greavu-Erban. 2020. Low cost big data solutions: The case of apache spark on beowulf clusters. In *IoTBDS 2020 - Proceedings of the 5th International Conference on Internet of Things, Big Data and Security*. 327–334. https://doi.org/10.5220/0009407903270334

[10] John Gantz and David Reinsel. 2011. Extracting value from chaos. *IDC iview* 1142, 2011 (2011), 1–12.

[11] Gartner. 2021. Definition of Big Data - IT Glossary | Gartner. https://www.gartner.com/en/information-technology/glossary/big-data. Accessed: 2021-05-12.

[12] Wajdi Hajji and Fung Po Tso. 2016. Understanding the performance of low power raspberry pi cloud for big data. *Electronics (Switzerland)* 5, 2 (2016). https://doi.org/10.3390/electronics5020029

[13] IBM. 2016. The 5 V's of big data. https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/. Accessed: 2021-06-20.

[14] Chanwit Kaewkasi and Wichai Srisuruk. 2015. Optimizing performance and power consumption for an ARM-based big data cluster. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, Vol. 2015-January. https://doi.org/10.1109/TENCON.2014.7022399

[15] Chae Soo Kim and Seung Beom Son. 2019. A Study on Big Data Cluster in Smart Factory using Raspberry-Pi. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* (2019), 5360–5362. https://doi.org/10.1109/BigData.2018.8622539

[16] Barbara Kitchenham. 2004. Procedures for Performing Systematic Reviews. *Keele University Technical Report TR/SE-0401* 33 (08 2004).

[17] Barbara A Kitchenham, David Budgen, and O Pearl Brereton. 2011. Using mapping studies as the basis for further research–a participant-observer case study. *Information and Software Technology* 53, 6 (2011), 638–651.

[18] Fabiano Kuss, Marcos Castilho, Leticia Peres, and Fabiano Silva. 2018. Aulacast: A Single Board Computer Platform to Support Teaching. 366–373. https://doi.org/10.5220/0006776803660373

[19] A J Lewis, M Campbell, and P Stavroulakis. 2016. Performance evaluation of a cheap, open source, digital environmental monitor based on the Raspberry Pi. *Measurement* 87 (2016), 228–235. https://doi.org/10.1016/j.measurement.2016.03.023

[20] Anthony M Middleton and Lexisnexis Risk. 2015. White Paper Introduction to HPCC (High-Performance Computing Cluster). (2015).

[21] Bolanle A Ojokoh, Oluwarotimi W Samuel, Olatunji M Omisore, Oluwafemi A Sarumi, Peter A Idowu, Emile R Chimusa, Ashraf Darwish, Adebayo F Adekoya, and Ferdinand A Katsriku. 2020. Big data, analytics and artificial intelligence for sustainability. *Scientific African* 9 (2020), e00551. https://doi.org/10.1016/j.sciaf.2020.e00551

[22] Carlos Ordonez, Sikder Tahsin Al-Amin, and Xiantian Zhou. 2020. A Simple Low Cost Parallel Architecture for Big Data Analytics. In *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*. 2827–2832. https://doi.org/10.1109/BigData50022.2020.9378386

[23] M. Pai, M. Mcculloch, Jennifer Gorman, N. Pai, W. Enanoria, Gail Kennedy, P. Tharyan, and J. Colford. 2004. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. *The National medical journal of India* 17 2 (2004), 86–95.

[24] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* (Italy) *(EASE'08)*. BCS Learning & Development Ltd., Swindon, UK, 68–77.

[25] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[26] Basit Qureshi and Anis Koubaa. 2020. On performance of commodity single board computer-based clusters: A big data perspective. *EAI/Springer Innovations in Communication and Computing* (2020), 349–375. https://doi.org/10.1007/978-3-030-13705-2_15

[27] Venkat Rayala and Satyanarayan Reddy Kalli. 2021. Big data clustering using Improvised Fuzzy C-Means clustering. *Revue d'Intelligence Artificielle* 34, 6 (2021), 701–708. https://doi.org/10.18280/RIA.340604

[28] João Saffran, Gabriel Garcia, Matheus A. Souza, Pedro H. Penna, Márcio Castro, Luís F.W. Góes, and Henrique C. Freitas. 2017. A low-cost energy-efficient raspberry Pi cluster for data mining algorithms. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10104 LNCS (2017), 788–799. https://doi.org/10.1007/978-3-319-58943-5_63

[29] Remo Scolati, Ilenia Fronza, Nabil El Ioini, Areeg Samir, Hamid Reza Barzegar, and Claus Pahl. 2020. A Containerized Edge Cloud Architecture for Data Stream Processing. *Communications in Computer and Information Science* 1218 CCIS, May (2020), 150–176. https://doi.org/10.1007/978-3-030-49432-2_8

[30] Remo Scolati, Ilenia Fronza, Nabil El Ioini, Areeg Samir, and Claus Pahl. 2019. A containerized big data streaming architecture for edge cloud computing on clustered single-board devices. In *CLOSER 2019 - Proceedings of the 9th International Conference on Cloud Computing and Services Science*. 68–80. https://doi.org/10.5220/0007695000680080

[31] Kathiravan Srinivasan, Chuan Yu Chang, Chao Hsi Huang, Min Hao Chang, Anant Sharma, and Avinash Ankur. 2018. An efficient implementation of mobile Raspberry Pi Hadoop clusters for Robust and Augmented computing performance. *Journal of Information Processing Systems* 14, 4 (2018), 989–1009. https://doi.org/10.3745/JIPS.01.0031

[32] Dan Tofan, Matthias Galster, Paris Avgeriou, and Wes Schuitema. 2014. Past and future of software architectural decisions – A systematic mapping study. *Information and Software Technology* 56, 8 (2014), 850 – 872.

[33] Trickdroid. 2021. Compreendendo o Big Data com Benefícios, Características e Exemplos de Aplicativos de Big Data. https://trickdroid.org/compreendendo-o-big-data-com-beneficios-caracteristicas-e-exemplos-de-aplicativos-de-big-data/. Accessed: 2021-07-06.

[34] M. Ali ud-din Khan, M. Uddin, and Navarun Gupta. 2014. Seven V's of Big Data understanding Big Data to extract value. *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education* (2014), 1–5.

[35] Xiaodong Wang, Shouhao Jiang, Xiaowei Xu, Zhiyong Wu, and Ye Tao. 2017. A Raspberry Pi and LXC Based Distributed Computing Testbed. *Proceedings - 2016 International Conference on Digital Home, ICDH 2016* (2017), 170–174. https://doi.org/10.1109/ICDH.2016.044

[36] Kehe Wu, Yayun Zhu, and Zuge Chen. 2017. Time Series Data Storage System Based on Big Data Technologies: Architecture and Implementation. *International Journal of Emerging Electric Power Systems* 18, 3 (2017). https://doi.org/10.1515/ijeeps-2016-0279

APÊNDICE B – *A Systematic Review on Teaching Parallel Programming*

A Systematic Review on Teaching Parallel Programming

José Aprígio C Neto
Federal Institute of Sergipe
jose.neto@ifs.edu.br

Antônio José A Neto
Federal University of Sergipe
antonio.neto@dcomp.ufs.br

Edward David Moreno
Federal University of Sergipe
edward@dcomp.ufs.br

ABSTRACT

This work aimed to perform a systematic review of the literature on teaching parallel programming using low-cost clusters, identifying the main programming languages, hardware platforms and software tools used in teaching-learning this type of programming. The research results showed that the most used clusters in the teaching of parallel programming were assembled from multicore machines (Cluster Beowulf) and by single board computers (SBC), in addition to multicore machines with graphics acceleration cards (GPUs). Regarding the use of programming languages, software tools and parallelism libraries used in teaching parallel programming, it is observed that most of the researched works mentioned the use of C, C++, and JAVA programming languages, and as parallelism libraries the use of MPI, OpenMP, CUDA and Apache Hadoop. Furthermore, the tests on the clusters were carried out through the implementation of parallelized generic algorithms and, in some cases, using algorithms that involve matrix operations.

CCS CONCEPTS

• **Applied computing**; • **Education**; • **Interactive learning environment**;

KEYWORDS

Parallel Programming, Clusters, Teaching, Education

ACM Reference Format:
José Aprígio C Neto, Antônio José A Neto, and Edward David Moreno. 2022. A Systematic Review on Teaching Parallel Programming. In *11 Euro American Conference on Telematics and Information Systems (EATIS2022), June 01–03, 2022, Aveiro, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3544538.3544659

1 INTRODUCTION

Parallel architectures are computational structures constituted by multiple processors that cooperate with each other to solve complex problems that involve a high hardware processing power, exploring different levels of parallelism [1].

In this context, parallel programming emerges as an important tool in solving computational problems with a high degree of complexity, aiming to explore parallelism techniques, as well as the use of libraries that work with the explicit execution of processes or threads in a way simultaneous.

Parallel programming is a technique based on data parallelism and consists of the simultaneous execution of complex tasks that involve large volumes of data [3], dividing them into smaller tasks that will be distributed and executed by several processors simultaneously [4]. Parallel programming aims to exploit the potential concurrent existent in a program's code, using parallelism libraries that work with the explicit execution of processes and/or threads simultaneously [2].

For this, the equipment used in this type of programming must have a parallel architecture [2], where these processors and/or cores can communicate with each other so that there is a synchronization in the execution of tasks [4]. Considering that parallel applications involve great computational power and demand a high response time, parallel processing seeks to reduce this time by dividing the computational load between processors and/or cores, allowing an acceleration in the execution of tasks and consequently, a reduction in the response time of the tasks [4].

However, teaching parallel programming is a complex and challenging task, considering that it involves a series of programming steps and models, as well as solving computational problems and understanding various types of hardware platforms, in addition to skills and skills to work with this new type of technology.

Therefore, this article aims to make a systematic review of the literature on studies carried out in the context of teaching parallel programming using low-cost clusters, identifying the main methodologies, architectures and software tools used in the teaching-learning process of this type of programming.

2 RELATED WORKS

Mwasaga and Joy [8] carried out a systematic literature review to investigate articles published on the teaching of high-performance computing in the period from 1988 to 2018, showing the main artifacts used as interventions in the teaching-learning process of HPC (High-performance computing), considering that these artifacts facilitate the understanding of students in the study of parallel and distributed computing.

The results of the authors' research [8] identified 211 articles related to the topic addressed. In addition, the study revealed that most publications reported having used Beowulf clusters in their practices, as well as pedagogical tools that helped in the development of these activities. In the work, the authors also report that the study was of fundamental importance for the compression of which artifacts were most used in the teaching of high-performance computing (HPC).

3 METHODOLOGY

The systematic review consists of a secondary study method that uses primary studies related to a particular research topic as a data source, with the aim of answering a specific question in an objective and impartial manner. The methods used in the preparation of systematic reviews include research question elaboration, literature search, article selection, data extraction, methodological quality

EATIS2022, June 01–03, 2022, Aveiro, Portugal

ACM ISBN 978-1-4503-9738-4/22/06…$15.00
https://doi.org/10.1145/3544538.3544659

assessment, data synthesis, evaluation of the quality of evidence, and writing and publishing the results [5].

The systematic review carried out in this article focused on articles published in the scientific databases of Scopus and Web of Science from January 2012 to November 2021.

3.1 Research Questions (RQs)

To fulfill the objective of this article, the following main question that will guide the studies (MQ) was defined: "What are the teaching strategies and hardware and software technologies used in the implementation of a cluster for teaching parallel programming?

To answer the main question (MQ) of this research, the following questions were raised in parallel:

- RQ1: What programming languages were used in the development of parallel algorithms for executing and testing clusters?
- RQ2: What hardware platforms were used in implementing the clusters, as well as in practical activities?
- RQ3: What software tools were used in the construction of clusters, as well as in the development of practical activities?
- RQ4: What types of benchmarks were used in the cluster performance analysis tests?

The query expression used in this research was "((parallel AND programming) AND (educat* OR teach* OR learn*) AND (framework OR solution* OR tool* OR code*) AND (cluster*))".

To limit the number of references, in searches carried out in bibliographic databases, the query expression was used in the following search fields: title, abstract and keywords.

3.2 Inclusion and Exclusion Criteria

The criteria used for the inclusion of articles used in this systematic review were:

- IC1: Publications focused on teaching parallel programming using a cluster.
- IC2: Publications that could respond to RQs.
- IC3: Articles published between 2012 and 2021.
- IC4: Articles written in English or Portuguese.

Regarding the exclusion criteria of the articles identified in the searches of this systematic review, the following criteria were used:

- EC1: Gray Literature.
- EC2: Studies inconsistent with the research topic.
- EC3: Duplicate articles.
- EC4: Articles not available for analysis.

3.3 Selected Publications

According to the search criteria used in this research, 479 publications were initially identified, 288 belonging to the Web of Science database and 191 to the Scopus database. Of which 69 were excluded for being duplicates and 385 for being outside the scope of the investigations proposed in this work, leaving a total of 25 publications for further analysis, 16 in the Web of Science database and 09 in the Scopus database.

It is noteworthy that all publications identified and analyzed in this research met the inclusion and exclusion criteria established in the scope of work. In addition, most publications ranked for further analysis have enough information that they can identify strategies, hardware platforms and software tools. However, some publications selected as the object of study contain only partial information related to the research questions.

4 ANALYSIS AND RESULTS

In this section, the results obtained in the searches carried out in the respective databases specified in this work will be presented, analyzed, and discussed.

4.1 Programming Languages

The survey results identified the use of various programming languages in teaching parallel programming. Among the identified languages, three of them stand out regarding their use, they are: the C, C++, and JAVA language, as shown in Table 1.

Table 1: Languages used in developing parallel algorithms

Languages	References
C	[6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [17], [19], [23], [24], [26], [29], [31]
C++	[6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [17], [19], [29], [30], [31]
JAVA	[6], [18], [19], [26], [29], [31]
Fortran	[9], [17], [31]
GCC++	[10], [13], [19]
Python	[19], [31]
GCC	[10], [13]

The use of these programming languages is more frequent because they are traditional languages, which have simple command syntaxes that are easy for students to understand. Furthermore, several parallelism libraries support its use.

In the case of the JAVA language, it is observed that its use is because it is a language that presents a compilation and execution time suitable for use in parallel programming. In addition, it allows fast and accurate error detection and debugging, as well as support the use of parallelism through the threading API.

In addition to the programming languages mentioned in Table 1, other programming languages used in teaching parallel programming were identified in the researched works, however, all with low relevance in relation to the number of times they were mentioned by the authors surveyed. Therefore, these languages were not inserted in the table, they are SCALA [6], GO [6], RUST [6], GFortran [10], C# [22], PHP [26] and HTML [26]. These languages were mentioned only once.

It is noteworthy that, when it comes to teaching programming, it is essential that undergraduate students have at least knowledge of some types of programming languages, including: a systems language, an object-oriented language, and a scripting language.

4.2 Hardware Platforms

In the analysis of the researched articles, several types of hardware were identified in the cluster solutions used in teaching parallel

Table 2: Hardware platforms used in clusters

Hardware	References
Multicore Computers	[6], [7], [8], [9], [10], [11], [15], [18], [19], [20], [22], [24], [26], [27], [28], [29], [30]
GPUs	[7], [11], [14], [18], [19], [20], [22], [29]
Raspberry Pi	[6], [10], [12], [14], [16], [31]
ODROID	[10], [11], [12],[13]
Jetson	[10], [12], [16]
Laptop's Multicore	[6], [18], [25]
Servers (multicore)	[6], [10], [11]
Servers NFS	[11], [26]
Parallella	[12], [15]

programming, from simple low-cost hardware, such as single board computers (SBCs), to supercomputers, as shown in Table 2.

Among the different types of hardware identified in the survey, the highlight goes to multicore computers, used in Beowulf cluster solutions. Their greater use is because they are machines that are more easily found in the institutions' laboratories, with no need for new investments to acquire new equipment.

It is worth noting that in some studies the use of graphics accelerator cards (GPUs) and single card computers (SBCs), such as the Raspberry Pi, was also identified in cluster solutions.

Other hardware platforms appeared in the searches during the analysis of the search results, however, with little relevance in relation to the number of times they were cited by the authors. These hardware platforms were removed from Table 2, they are: LattePanda [10], VIM3Pro [10], Hikey970 [10], Servers NAS [10], Cubieboard [13] and Supercomputer [23]. These hardware platforms were mentioned only once.

4.3 Software Tools

The researchers identified several types of software tools used in the development of activities involving the teaching of parallel programming, among them the following stand out: OpenMP, MPI, CUDA, MPICH, POSIX and Apache Hadoop, as shown in Table 3.

Each of these software has its own characteristics, thus, its uses and implementations depend on the type of parallel architecture used, as well as the type of problem it is intended to solve.

The MPI (Message Passing Interface) is widely used in multiprocessing based on message-passing, where data is moved through message exchange operations.

OpenMP (Open Multi-Processing) is an API that provides parallel processing of algorithms between the cores of a processor with shared memory, which can be used in different types of architectures that have shared memory.

CUDA (Compute Unified Device Architecture) is an API that works with graphics cards (GPUs), designed to work with the programming languages C, C++, and Fortran.

Hadoop is an open-source tool developed in JAVA to work with clusters of common machines and with a high volume of data processing. Furthermore, it is a fault tolerant tool.

Table 3: Software platforms used in clusters

Software	References
OpenMP	[6], [7], [10], [11], [12], [13], [14], [15], [16], [18], [19], [21], [24], [29], [30], [31]
MPI	[6], [7], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [21], [23], [24], [26], [27], [28], [29], [30], [31]
CUDA	[6], [7], [10], [14], [18], [19], [20], [16], [29]
Apache Hadoop	[6], [18], [19], [17], [29], [31]
MPICH	[6], [9], [11], [13], [31]
POSIX	[6], [15], [17], [19], [31]
OpenCL	[6], [10], [13], [18]
Ubuntu	[10], [11], [26]

MPICH is a standard for passing messages involving distributed memory applications, used in parallel computing, and is available for the Linux operating system.

POSIX (Portable Operating System Interface) is an interface for programming APIs, which guarantees the portability of a program's source code from an operating system, also offering real-time services, interface with threads and communication between processes via network.

In addition to the software tools mentioned, other tools were also identified during the analysis of research results, however, as they were mentioned only once, they were not included in Table 3, they are: Scala [6], OpenACC, [6], Julia [6], Erlang [6], TBB [7], OpenSSH [9], Glibc [10], OmpSs [10], Cubian [13], Epiphany [15], Pilot [17], Intel Cilk Plus [18], JVM [18], MPL Express [18], JCUDA [18], SAUCE [19], NVCC [19], Task Parallel [22], Microsoft Visual Studio [22], Scholar [24], Rstudio Server [24], Thin line [24], Jupyterhub [24], OpenNebula [26], Windows 7 [27], VirtualBox [27] and NPACI Rocks Cluster [27].

4.4 Benchmarks Used in Cluster Testing

During the analysis of the results, several types of benchmark tests involving the use of parallel algorithms were identified, as shown in Table 4.

Table 4: Benchmarks and tests used in clusters

Benchmarks	References
Generic Parallel Algorithms	[6], [7], [9], [10], [11], [12], [13], [14], [15], [17], [18], [19], [16], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]
Linpack	[9], [31]

All tests aimed to analyze the behavior and performance of the cluster solutions in question.

In addition to the generic parallel algorithms used for the tests on the clusters, the presence of a specific benchmark test was evidenced in some analyzed works, where the Linpack tool was used. This

tool is used in ranking of the most powerful supercomputers in the world, the TOP500 list. Performance measured by Linpack equates to the number of floating-point operations a machine can perform per second (FLOPS).

Furthermore, to the mentioned tests, several other tests were identified during the analysis of the research results, however, they were removed from the table because they were mentioned only once, they are Load Tests [6], PARSEC [10], SPLASH-2 [10], Rodinia [10], SPEC OMP2012 [10], Thread Number Tests [11], Process Number Tests [11].

5 CONCLUSIONS

According to the research results, it is observed, in general, that the teachers of educational institutions have been using small clusters, formed by multicore machines or single board computers (SBCs), for practical activities involving the teaching of parallel programming, due to the fact that these equipment's have a low acquisition cost or because they are machines that are already available in the institution itself, in the laboratories, with no need to invest in the acquisition of new equipment for the development of these activities.

Regarding the libraries used in teaching parallel programming, there is a predominance of the use of libraries that involve passing through messages, as is the case with MPI. Message passing forms the basis of high-performance computing codes, making it a natural choice to begin studies of parallel programming. In addition to the MPI library, several other libraries supported practical activities in teaching parallel programming, including: OpenMP, CUDA, MPICH, POSIX and Apache Hadoop. It is worth noting that most of these libraries can be implemented with the help of programming languages C, C++, JAVA, and Python.

In the tests and performance analysis of the clusters implemented by the researched authors, it is verified the use of benchmarks executed from generic parallelized algorithms and operations involving the use of matrices, where students were able to evidence and understand in practice the scheduling of machines parallel, as well as the functioning of architectures that use shared memory.

As a future work, we intend to develop practical studies based on the construction of low-cost clusters, with the objective of encouraging the use of parallel and distributed programming in courses in the computing area of higher education institutions.

REFERENCES

[1] Schepke, Claúdio. 2009. Ambientes de Programação Paralela. Trabalho Individual I, Universidade Federal do Rio Grande do Sul/PPGC.

[2] Soares, Felipe. Augusto. Lara., Nobre, Cristiane. Neri., de Freitas, H.C. 2019. Parallel programming in computing undergraduate courses: a systematic mapping of the literature. IEEE Latin America Transactions, 17(08), 1371-1381.

[3] Sato, Liria. Matsumoto., Midorikawa, Edson. Toshimi., & Senger, Hermes. 1996. Introdução a programação paralela e distribuída. Anais do XV Jornada de Atualização em Informática, Recife, PE, 1-56.

[4] Rebonatto, M. T. 2004. Introdução à programação paralela com MPI em agregados de computadores (clusters). In Congresso Brasileiro de Ciência da Computação, Itajaí (pp. 938-955).

[5] Galvão, Taís. Freire., & Pereira, Mauricio. Gomes. 2014. Revisões sistemáticas da literatura: passos para sua elaboração. Epidemiologia e Serviços de Saúde, 23, 183-184.

[6] Galvão Adams, Joel. C. 2021. Evolving PDC curriculum and tools: A study in responding to technological change. Journal of Parallel and Distributed Computing, 157, 201-219.

[7] Bednárek, David., Kruliš, Martin., & Yaghob, Jakub. 2021. Letting future programmers experience performance-related tasks. Journal of Parallel and Distributed Computing, 155, 74-86.

[8] Mwasaga, Nkundwe. Moses., & Joy, Mike. 2020. Using high-performance computing artifacts as a learning intervention: a systematic literature review. In Proceedings of the 2nd International Conference on Intelligent and Innovative Computing Applications (pp. 1-10).

[9] Datti, Ahmad. A., Umar, Hadiza. A., & Galadanci, Jamil. 2015. A beowulf cluster for teaching and learning. Procedia Computer Science, 70, 62-68.

[10] Velásquez, R. A., Isaza, S., Montoya, E., García, L. G., & Gómez, J. 2020. Embedded cluster platform for a remote parallel programming lab. In 2020 IEEE Global Engineering Education Conference (EDUCON) (pp. 763-772). IEEE.

[11] Alvarez, Lluc., Ayguade, E. & Mantovani, F. 2018. Teaching hpc systems and parallel programming with small-scale clusters. In 2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC) (pp. 1-10).

[12] Adams, J. C., Matthews, S. J., Shoop, E., Toth, D., & Wolfer, J. 2017. Using inexpensive microclusters and accessible materials for cost-effective parallel and distributed computing education. Journal of Computational Science Education, 8(3), 2.

[13] Toth, David. 2014. A portable cluster for each student. In 2014 IEEE Intl. Parallel & Distributed Processing Symposium Workshops (pp. 1130-1134).

[14] Adams, Joel. C., Caswell, J., Matthews, S. J., Peck, C., Shoop, E., Toth, D., & Wolfer, J. 2016. The micro-cluster showcase: 7 inexpensive beowulf clusters for teaching pdc. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 82-83).

[15] Matthews, Suzanne. J. 2016. Teaching with parallella: A first look in an undergraduate parallel computing course. Journal of Computing Sciences in Colleges, 31(3), 18-27.

[16] Wolfer, James. 2015. A heterogeneous supercomputer model for high-performance parallel computing pedagogy. In 2015 IEEE Global Engineering Education Conference (EDUCON) (pp. 799-805). IEEE.

[17] Gardner, William. B., & Carter, John. D. 2014. Using the pilot library to teach message-passing programming. In 2014 Workshop on Education for High Performance Computing (pp. 1-8). IEEE.

[18] Shafi, A., Akhtar, A., Javed, A., & Carpenter, B. 2014. Teaching parallel programming using java. In 2014 Workshop on Education for High Performance Computing (pp. 56-63).

[19] Shafi, Schlarb, Moritz., Hundt, Christian., & Schmidt, Bertil. 2015. Sauce: A web-based automated assessment tool for teaching parallel programming. In European Conference on Parallel Processing (pp. 54-65). Springer, Cham.

[20] Ludin, M., Weeden, A., Houchins, J., Thompson, S., Peck, C., Babic, I., & Sergienko, E. 2013. LittleFe: The high-performance computing education appliance. In IEEE Intl. Conf. on Cluster Computing (CLUSTER) (pp. 1-1).

[21] Cuenca, Javier., & Giménez, Domingo. 2016. A parallel programming course based on an execution time-energy consumption optimization problem. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 996-1003).

[22] Liu, Jie. 2016. 20 years of teaching parallel processing to computer science seniors. In 2016 Workshop on Education for High-Performance Computing (EduHPC) (pp. 7-13).

[23] Moore, Shirley. V., & Dunlop, Steven. R. 2016. A flipped classroom approach to teaching concurrency and parallelism. In 2016 IEEE Intl. Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 987-995).

[24] Baldwin, M. E., Zhu, X., Smith, P. M., Harrell, S. L., Skeel, R., & Maji, A. 2016. Scholar: A campus hpc resource to enable computational literacy. In 2016 Workshop on Education for High-Performance Computing (EduHPC) (pp. 25-31). IEEE.

[25] Capel, Manuel. I., Tomeu, Antonio. J., & Salguero, Alberto. G. 2017. Teaching concurrent and parallel programming by patterns: An interactive ICT approach. Journal of Parallel and Distributed Computing, 105, 42-52.

[26] Maia, Régis. Matheus. Silveira. 2016. Ferramenta para criação de clusters virtuais para o ensino de programação paralela e distribuída.

[27] Beserra, D., França, M., Melo, C., Sousa, Y., Romeiro, S., Andrade, M., & Sousa, E. 2013. Ambiente virtualizado para ensino de programaçao paralela e computaçao em cluster. In XXXIII Congresso da Sociedade Brasileira de Computação/XXI Workshop sobre Educação em Computação, Maceió.

[28] Shoop, E., Brown, R., Biggers, E., Kane, M., Lin, D., & Warner, M. 2012. Virtual clusters for parallel and distributed education. In Proc. of the 43rd ACM technical symposium on Computer Science Education (pp. 517-522).

[29] Shoop Saule, Erik. 2018. Experiences on teaching parallel and distributed computing for undergraduates. In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 361-368).

[30] Mangual, O., Teixeira, M., Lopez, R., & Nevarez, F. 2014. Hybrid MPI-OpenMP versus MPI Implementations: A Case Study. POLYTECHNIC UNIV OF PUERTO RICO HATO REY.

[31] Mwasaga, N. M., & Joy, M. 2020. Implementing micro High-Performance Computing (μHPC) artifact: Affordable HPC Facilities for Academia. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-9).

APÊNDICE C – *Clusters Big Data* utilizando *Raspberry Pi* e *Apache Hadoop* - Uma Quasi-Revisão Sistemática da Literatura

Clusters Big Data utilizando *Raspberry Pi* e *Apache Hadoop* - Uma Quasi-Revisão Sistemática da Literatura

1st Antônio José A. Neto*, 2nd José M. dos Santos*, 3rd José A. C. Neto§, 4rd Edward D. Moreno*
*Departamento de Computação - Universidade Federal de Sergipe (UFS) - São Cristóvão - Sergipe
§Coordenação de Computação - Instituto Federal de Sergipe (IFS) - Itabaiana - Sergipe
{antonio.neto, edward}@dcomp.ufs.br, marcelotos@academico.ufs.br, jose.neto@ifs.edu.br

***Resumo*—Este trabalho tem como objetivo identificar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados. Para tal fim, foi elaborada uma Quasi-Revisão Sistemática da Literatura (QRSL), resultando em 9 artigos relevantes aptos a responder 3 questões de pesquisa. A QRSL identificou que os modelos de *Raspberry Pis* mais utilizados no desenvolvimento dos *clusters* são a *Raspberry Pi 4B* e a *Raspberry Pi 2B*, e que para sua validação os *benchmarks Terasort* e *Wordcount* são os mais citados na literatura, seguidos da abordagem original do *Map Reduce* e o *TestDFSIO*. As 3 únicas ferramentas encontradas para monitoramento dos recursos do *cluster* foram a *Ganglia*, *Grafana* e a *Prometheus*.**

***Index Terms*—*Cluster*, *Big Data*, *Raspberry Pi*, *Apache Hadoop*, *Benchmarks*, Revisão Sistemática**

I. Introdução

O termo *big data* é definido como os ativos de informações de alto volume, alta velocidade e/ou alta variedade, que exigem formas inovadoras e eficazes de processamento de informações, permitindo *insights* aprimorados, tomada de decisões e automação de processos [1].

Com os avanços na utilização de *big data* e *data science*, são exigidos, cada vez mais, recursos computacionais para processar esses grandes volumes de dados, tornando-se indispensável o uso de computação de alto desempenho (*High Performance Computing* - HPC).

A HPC refere-se a um sistema de computação que inclui vários processadores como parte de uma única máquina ou um *cluster* trabalhando como um recurso individual. A HPC deve sua característica de computação de alta velocidade à sua grande capacidade de processar informações [2]. Em contra partida, por conta do seu alto custo, tem se buscado alternativas de soluções mais baratas e eficazes. Atualmente, com os avanços de *hardware* que o mundo está presenciando, a *Raspberry Pi* tem trazido oportunidades de implantações de *clusters* econômicos e energeticamente eficientes.

A *Raspberry Pi* é um Computador de Placa Única (*Single Board Computer* - SBC) desenvolvido para promover a ciência da computação na educação [3]. Os *clusters* formados por esses dispositivos, atrelado ao uso do *Apache Hadoop*, têm se mostrado uma solução viável e econômica para a realização de tarefas que envolvem o uso de *big data*.

A montagem de um *cluster* formado por esse tipo de equipamento (Figura 1) não requer grandes investimentos para sua implantação. Além disso, a *Raspberry Pi* permite o uso de diversas bibliotecas de paralelismo, bem como o desenvolvimento de diversos algoritmos paralelos utilizando linguagens de programação [4] e os pesquisadores vêm convergindo para a ideia de que a melhor forma de implementar um *cluster* de *big data* de baixo custo é combinando *Raspberry Pi* com *Apache Hadoop* [5]–[8].

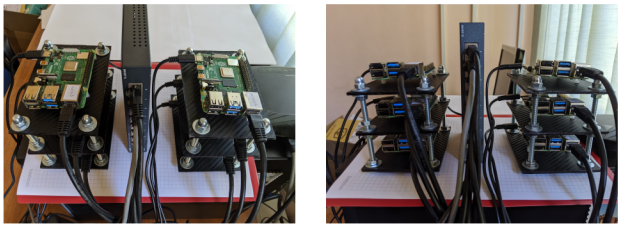

Figura 1. *Cluster Raspberry Pi*. Adaptado de Komninos *et al.* [9].

O *Apache Hadoop* é uma estrutura que permite o processamento distribuído de grandes conjuntos de dados em *clusters*, utilizando modelos de programação simples. Ele foi projetado para escalar desde servidores únicos até milhares de máquinas, cada uma, oferecendo computação e armazenamento locais. Ao invés de depender de recursos de *hardware* para fornecer alta disponibilidade, a própria biblioteca foi projetada para detectar e lidar com falhas na camada de aplicação, oferecendo um serviço altamente disponível em um *cluster* [10].

Seguindo essa vertente, este trabalho tem como objetivo identificar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados. Para tal fim, foi elaborada uma Quasi-Revisão Sistemática da Literatura (QRSL), expondo os estudos relevantes nessa área de pesquisa.

O restante deste o trabalho está organizado da seguinte forma: A Seção II apresenta a metodologia utilizada. O estado da arte é descrito na Seção III. Os resultados encontrados através do método de pesquisa, filtragem e análise dos artigos selecionados são mostrados na Seção IV. A Seção V descreve as ameaças à validade deste trabalho. As considerações finais

sobre os resultados encontrados, bem como os trabalhos futuros sugeridos são apresentados na Seção VI, e por fim, os agradecimentos são feitos na Seção VII.

II. Metodologia

Uma QRSL é um estudo secundário baseado em evidências, que tem como proposta, o levantamento, a identificação, a avaliação e a classificação da literatura para responder questões de pesquisas específicas [11]. Diante disso, o protocolo utilizado neste trabalho segue as diretrizes de revisão sintetizadas por Kitchenham [11]. Embora a QRSL cumpra todas as fases de avaliação da revisão sistemática, ela não é considerada como tal pela falta de uma linha base para meta-análise, lhe faltando em seu estudo, modelos de comparação para os resultados [12].

Esta QRSL foi desenvolvida em três fases. São elas: Planejamento, Condução e Relatório Final da Revisão (Figura 2).

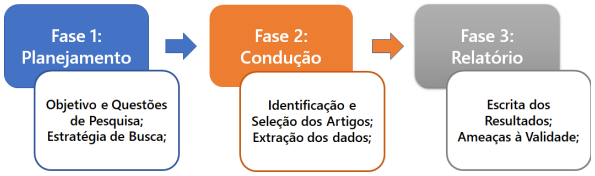

Figura 2. Fases do Planejamento da QRSL.

A. Planejamento

O planejamento foi iniciado com a definição do objetivo e das questões de pesquisa. Em seguida, a estratégia de busca foi construída, determinando as bases, os termos de busca e os critérios de inclusão e exclusão para a seleção dos artigos.

1) Objetivo e Questões de Pesquisa (QP): Este estudo tem o objetivo de levantar e avaliar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados. Para alcançar esse objetivo foram definidas as seguintes questões de pesquisa:

- QP1: Quais são os modelos de *Raspberry Pis* que são utilizados para o desenvolvimento do *cluster big data*?
- QP2: Quais são os *benchmarks*/algoritmos/técnicas que estão sendo utilizados no *cluster big data*?
- QP3: Quais ferramentas são utilizadas para monitorar os recursos do *cluster*?

2) Bases de Dados: As bases de dados utilizadas neste estudo foram a *Scopus* e a *Web of Science*. As mesmas foram utilizadas pelo fato de indexarem as principais bases de dados científicas, além de seus motores de busca automatizados possuírem uma maior eficiência em encontrar estudos relevantes [11]. É importante destacar que todas as bases de dados foram acessadas através do Portal de Periódicos da CAPES [13].

3) Termos de Busca: Após a definição das bases de dados para encontrar os estudos primários, foi estabelecida a *string* de busca, com finalidade de identificação e classificação desses estudos, conforme pode ser visto na Figura 3. Nas duas bases de dados utilizadas foram empregados recursos de busca avançada, com o propósito de encontrar as palavras-chave pesquisadas, contidas nos títulos e/ou nos resumos do material prospectado.

```
(big data) AND (cluster) AND
(raspberry) AND (hadoop)
```

Figura 3. *String* de Busca.

4) Critérios de Inclusão e Exclusão: Os critérios de inclusão e exclusão são utilizados para delimitar estudos mais relevantes que respondam, ou não, as questões de pesquisa. Foram definidos os seguintes critérios de inclusão (CI):

- CI1: Publicações focadas no desenvolvimento de um *cluster big data* usando *Raspberry Pi* e *Apache Hadoop*;
- CI2: Publicações que contribuam para responder pelo menos uma das QPs;
- CI3: Trabalhos publicados entre os anos de 2015 e 2022;
- CI4: Trabalhos escritos nos idiomas Português ou Inglês;

E os critérios de exclusão (CE) foram:

- CE1: Publicações inconsistentes com o tópico de pesquisa;
- CE2: Publicações duplicadas;
- CE3: Publicações indisponíveis para *download*;

B. Condução da Revisão

A busca dos trabalhos foi executada no final do primeiro semestre de 2022. A quantidade de artigos identificados, o método de filtragem e a extração dos dados são detalhados a seguir.

A busca retornou 17 artigos, que em uma primeira análise, foram verificados o título e o resumo desses estudos, atentando-se ao objetivo do trabalho e à sua conclusão, identificando o alinhamento do artigo ao escopo da QRSL, resultando em 10 artigos. Posteriormente, os 10 artigos foram analisados por completo, avaliando a capacidade dos mesmos em responder ao objetivo dessa QRSL. Dentre todos os trabalhos, apenas 1 não atendeu aos requisitos [14], sendo removido pelo fato de que o estudo diverge do escopo desta QRSL. Todo esse processo metodológico desta pesquisa pode ser visualizado através da Figura 4.

C. Relatório da Revisão

Ao final de todo o processo metodológico da QRSL, foram considerados 9 artigos como relevantes. Uma síntese de cada um dos trabalhos selecionados é apresentada na Seção III.

Os artigos analisados por esta QRSL, demonstraram o quão importante tem sido essa discussão e como a temática desta pesquisa tem sido explorada e pesquisada no mundo.

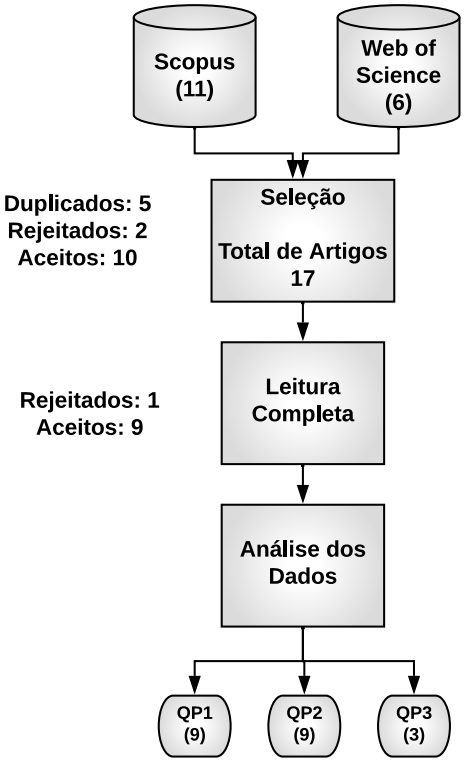

Figura 4. Processo de Seleção dos Artigos.

A Figura 5 demonstra a distribuição das participações nas publicações por continente, sendo a Europa o continente com maior percentual de publicações dentre os demais. Alguns trabalhos apresentam uma multinacionalidade entre os autores, resultando em mais de um país representado nos trabalhos encontrados.

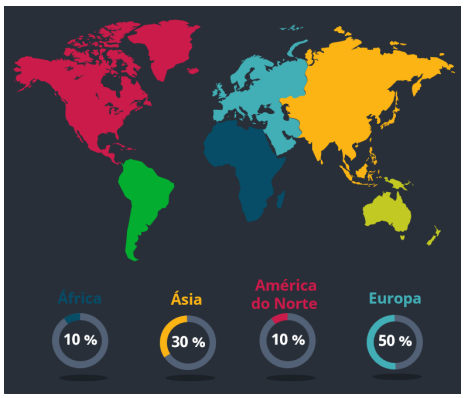

Figura 5. Distribuição das publicações por continente.

III. Estado da Arte

A. Big Data Processing on Single Board Computer Clusters: Exploring Challenges and Possibilities - Lee et al. [6]

Nesse trabalho, os autores exploraram os desafios das últimas gerações de *Raspberry Pis* utilizados em *clusters big data*. Eles construíram 1 *cluster big data* composto por 6 nós (5 escravos e 1 mestre), onde em cada nó escravo utiliza 1 *Raspberry Pi 4B* e o nó mestre utiliza 1 *Desktop* (Processador i5 de 3.7GHz (6 *cores*), *Solid State Drive* (SSD) de 500 *Gigabytes* (GB) e Memória RAM de 8 GB) e *Apache Hadoop*.

Os autores realizaram uma bateria de testes, a fim de avaliar a performance de uma única *Raspberry Pi*, assim como a performance do *cluster* desenvolvido, utilizando alguns dos principais *benchmarks*: *Wordcount*, *Terasort*, *TestDFSIO* e o *Pi Quasi-Monte Carlo*.

Eles comparam o *cluster* desenvolvido com o *Desktop* e com 1 *cluster big data* composto por 6 *Raspberry Pis 3B* (1 nó mestre e 5 nós escravos), além de estudarem o impacto da performance no armazenamento de dados utilizando 3 diferentes tipos de dispositivos: *Micro Storage Device Card* (MicroSD *Card*) de 32 GB; MicroSD *Card* 64 GB; *Universal Flash Storage* (UFS) de 256 GB.

Os mesmos concluíram que ao utilizar uma versão mais recente do *hardware Raspberry Pi*, o *cluster* tem uma melhor capacidade de processamento, bem como a utilização do dispositivo UFS, como unidade de armazenamento, tem um melhor *ratio* (*Input/Output*) em relação ao uso de MicroSD *Card*.

B. Hadoop Performance Analysis on Raspberry Pi for DNA Sequence Alignment - Turana et al. [15]

Nessa pesquisa, os autores buscaram analisar a sequência de alinhamento de DNA. Para isso, foi construído 1 *cluster Hadoop* na *Raspberry Pi B* usando o hardware da *Raspberry* como *commodity*. Foram utilizados 6 nós no *cluster* (1 nó mestre e 5 nós escravos). Eles usaram a biblioteca nativa *Biodoop* e o *benchmark Wordcount* para os testes, além de utilizar a ferramenta *Ganglia* para o monitoramento dos recursos do *cluster*.

Os autores realizaram uma comparação entre o *cluster Raspberry Pi B* e 6 *Desktops* (processador *dual core* de 1,86 GHz, *Hard Disk Drive (HDD)* de 160 GB e Memória RAM de 1 GB).

Foi concluído que é possível utilizar o *hardware* alternativo de baixo custo para implementar o *cluster Hadoop*, mesmo com a desvantagem do aumento de tempo na conclusão dos experimentos comparados.

C. Understanding the Performance of Low Power Raspberry Pi Cloud for Big Data - Haiji & Tso [5]

Esse trabalho apresenta um conjunto de experimentos para testar a performance de 1 nó único e de 1 *cluster* composto por 12 *Raspberry Pis 2B* (1 mestre e 11 escravos) com e sem ambiente de virtualização, utilizando a ferramenta *Docker* para estudar a viabilidade do mesmo para *big data analyics* em tempo real.

Utilizando os *benchmarks Wordcount* e *Terasort*, os autores executaram uma bateria de testes com diversas configurações de tamanhos de arquivos, variando entre 1 GB a 6 GB. Os mesmos avaliaram que em um ambiente virtualizado, o consumo de CPU e memória RAM torna-se maior, a taxa de

transferência da rede diminui e os picos de acessos ocorrem com menos frequência e menos intensidade.

D. Towards Green Data Centers - Bourhnane et al. [16]

No artigo mencionado, foi implementado um comparativo de *hardware* no uso do *Apache Hadoop*. Enquanto um experimento usou 1 *Desktop* (2 processadores *Core 2 Duo de 2.20 GHz, HDD de 160GB*), no outro foi utilizado 1 *cluster* com 5 *Raspberry Pis 3B+*, sendo 1 nó mestre e 4 nós escravos. Esse *cluster* tinha como diferencial, o uso de 5 HDDs de 1 TB (externos), além de 5 MicroSD *Card* (internos) de 8 GB.

A finalidade da pesquisa era agregar uma motivação no uso da prática de computação eficiente e ecológica, ou seja, na computação verde. Para tanto, foram executados testes com os *benchmarks TestDFSIO* e *Terasort* para a análise de desempenho e eficiência energética do *cluster*, utilizando a *Raspberry Pi*.

Os experimentos mostraram desempenho significativo com o *TestDFSIO* em relação ao *cluster* tradicional. No entanto, o *Terasort* forneceu um menor desempenho, que pode ser facilmente superado adicionando mais nós ao *cluster*. Nos testes de consumo energético no *cluster Raspberry Pi*, foi comprovado o baixo consumo de energia, provando ser esse, um experimento de computação verde.

E. An Efficient Implementation of Mobile Raspberry Pi Hadoop Clusters for Robust and Augmented Computing Performance - Srinivasan et al. [17]

Nesse trabalho, os autores utilizaram 6 *clusters* diferentes, sendo eles configurados com 5 a 10 *Raspberry Pis 3B*, sempre utilizando o padrão de 1 nó mestre e os nós remanescentes como escravos. Também foi utilizado 1 *Desktop* (processador i5, 3.1 GHz, Memória RAM de 8 GB) para os mesmos experimentos.

Foi explorado pelos autores, a implementação do *Hadoop*, tanto para o *cluster* construído com *Raspberry Pi* quanto para o *Desktop*. Tais experimentos visavam testes de extração de pontos de recurso em uma imagem utilizando o algoritmo *Surf* e comparando os resultados obtidos.

Foi verificado pelos pesquisadores que os *clusters* formados por *Raspberry Pi* têm melhor performance que o *Desktop* para grandes *datasets*.

F. A containerized big data streaming architecture for edge cloud computing on clustered single-board devices — A Containerized Edge Cloud Architecture for Data Stream Processing - Scolati et al. [18], [19]

Em ambos artigos foram realizados experimentos em 1 *cluster* de 8 *Raspberry Pis 2B* com o gerenciador *Docker Swarm*. Ao *cluster* foi conferida a seguinte configuração: 1 nó mestre, 4 nós escravos e 3 nós para coleta de dados do *cluster*.

Os autores implementaram o *cluster* utilizando o *Docker* para hospedar o *Apache Hadoop* e o *Apache Spark* como *containers* e, dessa forma, aprimorar o processamento de *streaming de dados*. Para monitorar os recursos do *cluster*, os mesmos usaram as ferramentas *Prometheus* e *Grafana*.

Eles avaliaram o desempenho dos mesmos, mostrando que o uso da "conteinerização" aumenta a tolerância à falhas e a facilidade de manutenção.

G. Scale-down Experiments on TPCx-HS - Böther & Rabl [7]

Nesse estudo, os autores fizeram a execução do *benchmark* TPCx-HS em 2 *clusters* utilizando *Raspberry Pi*: O primeiro *cluster* composto por 4 *Raspberry Pis 3B+* trabalhando como escravos e 1 *Raspberry Pi 4B* como mestre, utilizando *Micro SD Cards* de 32 GB em cada dispositivo e o segundo *cluster* composto por 5 *Raspberry Pis 4B* (1 mestre e 4 escravos), onde cada *Raspberry* utilizou MicroSD *Card* de 128 GB para armazenamento.

Nos experimentos, foram utilizadas 2 configurações diferentes nos tamanhos dos arquivos usados pelo *benchmark* TPCx-HS: 1 GB e 10 GB. Em seguida, foi feita a comparação dos resultados obtidos com os disponíveis no site[1] do *benchmark*.

Os autores concluíram que a geração de *Raspberry Pi 4B* resolve o gargalo de memória RAM que existia na geração de *Raspberry Pi 3B+*. Além disso, os mesmos defendem o uso de *clusters Raspberry Pi* devido ao seu custo-benefício.

H. Performance of Raspberry Pi microclusters for Edge Machine Learning in Tourism - Komninos et al. [9]

No desenvolvimento dessa pesquisa foram utilizadas 6 *Raspberry Pis 4B*, com 4 GB de RAM e cartão MicroSD *Card* de 64 GB em cada nó. No *cluster* foram instalados o *Hadoop* para armazenamento distribuído de arquivos e o *Spark* para aprendizado de máquina.

Tal estudo tinha como objetivo utilizar recursos computacionais em uma arquitetura distribuída de baixo custo, para atender aplicações turísticas através da análise de *big data*.

Após experimentos e testes, foi demonstrado que o desempenho do *cluster* foi suficiente para fins de treinamento e execução de modelos de aprendizado de máquina em um contexto de computação de borda.

IV. Resultados

Nesta seção, os resultados da extração de dados são expostos, analisados e discutidos e, para um melhor entendimento e visualização dos mesmos, foi feita uma subdivisão de acordo com as QPs.

A. QP1: Quais são os modelos de Raspberry Pis que são utilizados para o desenvolvimento do cluster?

A QRSL apontou 5 diferentes modelos de *Raspberry Pis* utilizados para o desenvolvimento dos *clusters* citados nos trabalhos encontrados, sendo que o *Raspberry Pi 2B* e *Raspberry Pi 4B* foram os modelos mais utilizados nos experimentos. Cada um foi citado em 3 artigos. Alguns trabalhos utilizaram mais de 1 modelo de *Raspberry Pi*, geralmente para fazer algum comparativo entre os resultados encontrados.

É possível observar que quanto mais recente é o trabalho, mais recente é a versão da *Raspberry Pi* (com exceção dos trabalhos de Scolati *et al*. [18], [19]), justificando-se pelo fato

[1]https://www.tpc.org/tpcx-hs/results/tpcxhs_perf_results5.asp?version=2

de que quanto mais nova é a versão do dispositivo, mais poderoso computacionalmente ele é.

Todos os modelos encontrados podem ser visualizados na Tabela I. Um comparativo entre os modelos existentes de *Raspberry Pis* pode ser encontrado no trabalho de SocialCompare [20].

Tabela I
MODELOS DE RASPBERRY PIS UTILIZADOS NO DESENVOLVIMENTO DO CLUSTER

Modelo	Referência
Raspberry Pi 4B	[6], [7], [9]
Raspberry Pi 3B+	[7], [16]
Raspberry Pi 3B	[6], [17]
Raspberry Pi 2B	[5], [18], [19]
Raspberry Pi B	[15]

B. QP2: Quais são os benchmarks/algoritmos/técnicas que estão sendo utilizados no cluster big data?

Foram identificados 9 *benchmarks*/algoritmos/técnicas nos artigos selecionados, sendo o *Terasort* e o *Wordcount* os 2 mais utilizados, com 3 citações cada. Alguns estudos mencionaram a utilização de mais de 1 *benchmark*/algoritmo/técnica.

A incidência da utilização dos algoritmos/*benchmarks Terasort* e o *Wordcount*, juntamente com o *TestDFSIO* e o *Pi Quasi-Monte Carlo*, deve-se ao fato de que os mesmos estão disponíveis nas distribuições do *Apache Hadoop*, sendo tecnicamente mais cômodo aos pesquisadores utilizarem *benchmarks*/algoritmos/técnicas padrões (*default*), do que a instalação de um novo.

É importante salientar que o algoritmo *Map-Reduce* citado nos trabalhos de Scolati *et al.* [18], [19] é baseado em sua abordagem original, diferentemente da abordagem do *Map-Reduce* do *Apache Hadoop*.

Todos os *benchmarks*/algoritmos/técnicas mencionados nos trabalhos encontrados podem ser observados na Tabela II.

Tabela II
BENCHMARKS/ALGORITMOS/TÉCNICAS UTILIZADAS NOS CLUSTERS BIG DATA

***Benchmark*/Algoritmo/Técnica**	**Referência**
Árvore de Decisão	[9]
Map-Reduce	[18], [19]
Pi Quasi-Monte Carlo	[6]
Regressão Linear	[9]
SURF	[17]
Terasort	[5], [6], [16]
TestDFSIO	[6], [16]
TPCx-HS	[7]
Wordcount	[5], [6], [15]

C. QP3: Quais ferramentas são utilizadas para monitorar os recursos do cluster?

Para resposta da QP3, apenas 3 trabalhos mencionaram o uso de alguma ferramenta para monitorar os recursos do *cluster*. Foram elas: *Ganglia*[2], *Grafana*[3] e *Prometheus*[4], conforme pode ser observado na Tabela III.

Todos os *softwares* são de código aberto (*open source*). O *Ganglia* é um sistema de monitoramento distribuído e escalável para uso na computação de alto desempenho, como *clusters* e computação em *grids*. O *Grafana* é uma plataforma para visualizar e analisar métricas por meio de *dashboards* personalizados. Ele tem suporte para diversos tipos de bancos de dados e pode ser instalado em qualquer sistema operacional. O *Prometheus* é uma solução de monitoramento para gravar e processar qualquer série temporal puramente numérica. Ele reúne, organiza e armazena métricas juntamente com identificadores exclusivos de data/hora.

Em alguns trabalhos, os autores mencionaram o fato de monitorar os recursos do *cluster* sem utilizar uma ferramenta específica para tal [5]–[7], [16].

Tabela III
FERRAMENTAS UTILIZADAS PARA O MONITORAMENTO DOS RECURSOS DO CLUSTER

Ferramenta	Referência
Ganglia	[15]
Grafana	[18], [19]
Prometheus	[18], [19]

V. AMEAÇAS À VALIDADE

Para que um trabalho seja aceito como contribuição ao conhecimento científico, o pesquisador precisa convencer os leitores de que as conclusões tiradas de um estudo empírico são válidas [21]. Nesta seção estão expostas todas as ameaças à validade encontradas neste estudo.

A. Validade de Construção

A *string* de busca e as questões de pesquisa podem não cobrir todos os estudos relevantes da área. Como forma de mitigação, as palavras-chave e as questões de pesquisa foram elaboradas a partir de uma leitura prévia do assunto, através do trabalho de Neto *et al.* [8].

B. Validade Interna

Caso seja executada novamente, esta QRSL poderia apresentar um resultado diferente do obtido. Para mitigar essa ameaça, a metodologia definida por Kitchenham [11] foi usada nesse trabalho. Conflitos na seleção de artigos foram discutidos entre os autores, para atenuar o viés pessoal na seleção dos estudos.

C. Validade Externa

O resultado da busca pode não conter todos os artigos relevantes para o estudo. Como forma de diminuir essa ameaça, a pesquisa foi realizada em duas das principais bases de dados científicas, *Web of Science* e *Scopus*.

[2]http://ganglia.sourceforge.net/
[3]https://grafana.com/
[4]https://prometheus.io/

VI. Considerações Finais

Neste trabalho, uma QRSL foi elaborada com o objetivo de levantar e avaliar como estão sendo desenvolvidos os *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*, e como os mesmos estão sendo validados e monitorados, através de 3 questões de pesquisa. Para responder essas questões, foram feitas buscas nas bases de dados *Scopus* e *Web of Science*, que retornaram 17 trabalhos. Após a aplicação do protocolo da QRSL, apenas 9 estudos foram classificados como relevantes.

Foram encontrados 5 modelos diferentes de *Raspberry Pis*, sendo o *Raspberry Pi 2B* e *Raspberry Pi 4B* os modelos mais utilizados nos experimentos e cada um citado em 3 artigos (QP1).

A respeito da QP2, foram identificados 9 *benchmarks*/algoritmos/técnicas nos artigos selecionados, sendo o *Terasort* e o *Wordcount* os 2 mais utilizados, com 3 citações cada. Alguns trabalhos mencionam o uso de mais de 1 *benchmark*/algoritmo/técnica.

Por fim, como resposta para a QP3, foram encontradas 3 ferramentas para monitoramento dos recursos do *cluster* (*Ganglia*, *Grafana* e *Prometheus*) em 3 dos 9 artigos. Alguns autores mencionaram o fato de monitorar os recursos do *cluster* sem a utilização de uma ferramenta específica.

Essa QRSL mostrou um leque significativo de experimentos no uso dos *clusters big data* de baixo custo, utilizando *Raspberry Pi* e *Apache Hadoop*. Além desta pesquisa mostrar resultados relevantes, fornecendo assim subsídios para um melhor entendimento sobre o desenvolvimento desses *clusters*, este artigo pode auxiliar outros pesquisadores para o desenvolvimento de novos estudos nessa área de pesquisa.

Como trabalho futuro, está sendo desenvolvido 1 *cluster big data* de baixo custo composto por 9 *Raspberry Pis 4B* (1 mestre e 8 escravos) e *Apache Hadoop*. Além disso, será utilizado 1 *Raspberry Pi 3B+* como servidor de monitoramento dos recursos do *cluster*, utilizando as ferramentas *Zabbix*[5] e *Grafana*, devido ao fato de que as mesmas são ferramentas *open-source* e os autores possuírem um conhecimento prévio na utilização das mesmas.

A priori, será utilizado nos experimentos o armazenamento (MicroSD *Card*) de 16 GB e posteriormente o armazenamento de 128 GB. Serão executados alguns dos *benchmarks* encontrados nesta QRSL, comparando os resultados obtidos com os resultados apresentados pelos *clusters* citados nesta QRSL, além de avaliar o comportamento dos recursos (CPU, RAM, Temperatura, etc.) do *cluster* desenvolvido.

VII. Agradecimentos

Agradecemos à todos os revisores que se dispuseram a avaliar este trabalho, melhorando a qualidade desta pesquisa.

Referências

[1] Gartner, "Big data," Disponível em: https://www.gartner.com/en/information-technology/glossary/big-data. Acesso em: 18 de maio 2022, 2022.

[2] A. Middleton and P. Solutions, "Hpcc systems: Introduction to hpcc (high-performance computing cluster)," *White paper, LexisNexis Risk Solutions*, 2011.

[3] P. Giger, S. Srikugan, and B. L. Persaud, "A Raspberry Pi Cluster for Teaching Big-Data Analytics," Master's thesis, Universität Zürich, 2020.

[4] N. M. Mwasaga and M. Joy, "Implementing micro high performance computing (μhpc) artifact: Affordable hpc facilities for academia," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–9.

[5] W. Hajji and F. P. Tso, "Understanding the performance of low power raspberry pi cloud for big data," *Electronics (Switzerland)*, vol. 5, no. 2, 2016. [Online]. Available: https://doi.org/10.3390/electronics5020029

[6] E. Lee, H. Oh, and D. Park, "Big Data Processing on Single Board Computer Clusters: Exploring Challenges and Possibilities," *IEEE Access*, vol. 9, pp. 142 551–142 565, 2021.

[7] M. Böther and T. Rabl, "Scale-down experiments on tpcx-hs," in *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*, ser. BiDEDE '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3460866.3461774

[8] A. J. A. Neto, A. C. Neto, and E. D. Ordonez, "Low-cost clusters on big data - a systematic study," in *Proceedings of the Euro American Conference on Telematics and Information Systems*, ser. EATIS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3544538.3544635

[9] A. Komninos, I. Simou, N. Gkorgkolis, and J. D. Garofalakis, "Performance of raspberry pi microclusters for edge machine learning in tourism," in *AmI*, 2019.

[10] A. S. Foundation, "Apache hadoop," https://hadoop.apache.org, 2022.

[11] B. Kitchenham, "Procedures for performing systematic reviews," *Keele University Technical Report TR/SE-0401*, vol. 33, 08 2004.

[12] G. H. Travassos, P. S. M. dos Santos, P. G. Mian, P. G. M. Neto, and J. Biolchini, "An environment to support large scale experimentation in software engineering," in *13th IEEE International Conference on Engineering of Complex Computer Systems (iceccs 2008)*, 2008, pp. 193–202.

[13] CAPES/MEC, "Portal de periódicos da capes," http://www.periodicos.capes.gov.br/, 2022.

[14] J. Lin, "Scaling down distributed infrastructure on wimpy machines for personal web archiving," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, p. 1351–1355. [Online]. Available: https://doi.org/10.1145/2740908.2741695

[15] J. S. Turana, H. Sukoco, and W. A. Kusuma, "Hadoop performance analysis on raspberry pi for dna sequence alignment," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 3, pp. 1059–1066, 2016.

[16] S. Bourhnane, M. R. Abid, R. Lghoul, K. Zine-Dine, N. Elkamoun, and D. Benhaddou, "Towards green data centers," in *Sustainable Energy for Smart Cities*, J. L. Afonso, V. Monteiro, and J. G. Pinto, Eds. Cham: Springer International Publishing, 2020, pp. 291–307.

[17] K. Srinivasan, C. Y. Chang, C. H. Huang, M. H. Chang, A. Sharma, and A. Ankur, "An efficient implementation of mobile Raspberry Pi Hadoop clusters for Robust and Augmented computing performance," *Journal of Information Processing Systems*, vol. 14, no. 4, pp. 989–1009, 2018.

[18] R. Scolati, I. Fronza, N. El Ioini, A. Samir, and C. Pahl, "A containerized big data streaming architecture for edge cloud computing on clustered single-board devices," *CLOSER 2019 - Proceedings of the 9th International Conference on Cloud Computing and Services Science*, no. May, pp. 68–80, 2019.

[19] R. Scolati, I. Fronza, N. El Ioini, A. Samir, H. R. Barzegar, and C. Pahl, "A Containerized Edge Cloud Architecture for Data Stream Processing," *Communications in Computer and Information Science*, vol. 1218 CCIS, no. May, pp. 150–176, 2020.

[20] SocialCompare, "Raspberrypi models comparison — comparison tables," Disponível em: https://socialcompare.com/en/comparison/raspberrypi-models-comparison. Acesso em: 07 de julho 2022, 2022.

[21] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, *Selecting Empirical Methods for Software Engineering Research*. London: Springer London, 2008, pp. 285–311.

[5] https://www.zabbix.com/

APÊNDICE D – *The development of a low-cost big data cluster using Apache Hadoop and Raspberry Pi. A complete guide*

Computers and Electrical Engineering 104 (2022) 108403

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

The development of a low-cost big data cluster using Apache Hadoop and Raspberry Pi. A complete guide☆

Antônio José Alves Neto [a,*], José Aprígio Carneiro Neto [b], Edward David Moreno [a]

[a] Federal University of Sergipe (UFS), São Cristovão, 49100-000, Sergipe, Brazil
[b] Federal Institute of Sergipe (IFS), Itabaiana, 49500-000, Sergipe, Brazil

ARTICLE INFO

Keywords:
Apache Hadoop
Big data
Cluster
Grafana
Raspberry Pi
Step-by-step
Terasort
TestDFSIO
Zabbix

ABSTRACT

This paper provides a complete guide to the development, testing, and monitoring of a low-cost big data cluster through a detailed step-by-step configuration and installation of Apache Hadoop using 9 Raspberry Pis 4B. For the tests and performance evaluation, were used the Terasort and TestDFSIO benchmarks. The benchmarks were performed in different sizes of data files (250 MB up to 1 GB) and different slaves nodes quantity (2, 4, and 8). The results showed that the combination of Raspberry Pi and Apache Hadoop can be a very efficient and robust solution to get a low-cost big data cluster, considering its costs/benefits. Using a Raspberry Pi 3B+ as a monitoring server, we installed the Zabbix and Grafana tools, making it possible to collect important information in real-time, helping to better monitoring of the cluster's devices and better visualization of the behavior and performance of the cluster.

1. Introduction

Over the years, the significant increase in the quantity and complexity of computational applications has led the processor industry to develop new computer architectures, based on the use of machines with multiple processing units, which cooperate among them, and explore different levels of parallelism, and provide a high processing power which is used in solving complex tasks that demand the use of a large volume of data (big data). Among these types of architectures, the clusters-based multicomputers stand out [1].

Clusters are loosely coupled systems, formed by a set of computers that work collaboratively with each other, using message exchange libraries. Furthermore, clusters formed by SBC (Single Board Computers) are a viable alternative for research development, being scalable, flexible, highly available, and low-cost solutions that allow load balancing and it is fault-tolerant [2].

Recently, researchers have been trying to implement big data clusters based on low-cost and low-energy hardware [1,3,4]. There are a significant amount of papers carried out in this direction. Most of them used SBCs as a basis of the clusters and tried to prove that their hardware supports the installation of different big data analytics platforms [5], standing out among these SBCs the Raspberry Pi [4].

The Raspberry Pi is a SBC developed to promote computer science education. This platform offers different models to fit different needs [6]. Clusters formed by Raspberry Pi have proved to be a viable and economical solution for carrying out tasks involving the use of Big Data. The assembly of a cluster formed by this equipment does not require large investments for its implementation. In addition, the Raspberry Pi allows the use of several parallelism libraries, as well as the development of several parallel algorithms using the most used programming languages on the market [7].

☆ This paper is for regular issues of CAEE. Reviews were processed by Associate Editor Dr. S. Smys and recommended for publication.

* Corresponding author.
E-mail address: antonio.neto@dcomp.ufs.br (A.J.A. Neto).

https://doi.org/10.1016/j.compeleceng.2022.108403
Received 17 April 2022; Received in revised form 19 September 2022; Accepted 20 September 2022
Available online 7 October 2022

However, the lack of detailed material explaining all the steps of the installation, the configuration process, and finally the certification that the Hadoop cluster is working correctly is a problem little explored by academia. As a way to improve this scenario, this study was proposed.

Thus, this paper has the goal of providing a complete guide to the development, testing, and monitoring of a low-cost big data cluster. To do this, a detailed step-by-step configuration and installation of Apache Hadoop using 9 Raspberry Pis 4B was developed. For the tests and performance evaluation of this cluster, the Terasort and TestDFSIO benchmarks were used.

In addition, a Raspberry Pi 3B+ was also used as a monitoring server. In this server, we installed monitoring tools (Zabbix and Grafana—open and free tools). These tools make it possible to collect some important information about each cluster's device, in real-time, such as temperature, network stats, memory and CPU usage, and processes, among others. This collected information helped to get a better monitoring of the cluster's devices.

This study is organized as follows: In Section 2 are presented some necessary prior knowledge about the theme of this study. The State-of-the-Art is presented in Section 3. Section 4 presents all the resources (hardware and software) used in cluster development. Section 5 shows the entire process of configuration and development of the cluster in a detailed form, and the presentation of the benchmarks used is done in Section 6. The tests execution on cluster is done in Section 7 and the results are presented in Section 8. Section 9 shows the monitoring process used in this paper. Section 10 presents the discussion of the obtained results and suggestions for future works, and finally, the last Section presents the acknowledgments.

2. Background

In this section, some necessary prior knowledge about the theme of this study is discussed.

2.1. Cluster

A cluster is a group of inter-connected computers or hosts that work together to support applications and middleware (e.g. databases). In a cluster, each computer is referred to as a "node". Unlike grid computers, where each node performs a different task, computer clusters assign the same task to each node. Nodes in a cluster are usually connected through high-speed local area networks. Each node runs its instance of an operating system. A computer cluster may range from a simple two-node system connecting two personal computers to a supercomputer with a cluster architecture. Computer clusters are often used for cost-effective high-performance computing (HPC) and high availability (HA). If a single component fails in a computer cluster, the other nodes continue to provide uninterrupted processing. A computer cluster can provide faster processing speed, larger storage capacity, better data integrity, greater reliability, and wider availability of resources. Computer clusters are usually dedicated to specific functions, such as load balancing, high availability, high performance, or large-scale processing [8].

2.2. Raspberry Pi

The Raspberry Pi is a small SBC and it is strongly used to promote computer science education since it offers different models to fit different needs. For example, the Raspberry Pi Zero is ideal for small robotics projects because of the tiny form factor and low price. But, it does not have an Ethernet or a powerful processor. On the other hand, the Raspberry Pi 4B is suited for more sophisticated projects and comes with a quad-core Cortex-A72 (ARM v8) 64-bit processor and up to 4 GB of SDRAM (see Fig. 1) [6].

Fig. 1. Raspberry PI.
Source: Adapted from Qureshi and Koubba [9].

The official operating system is called Raspberry Pi OS, a Debian-based Linux distribution optimized for Raspberry Pi. Besides the officially supported Raspberry Pi OS, several third-party operating systems exist. In general, software and hardware are backward-compatible which simplifies upgrading [6].

2.3. Big data

The definition of "Big Data" is complex and constantly changing. However, there is some consensus in the literature on the main characteristics of Big Data as described by a widely cited Gartner report [10]: "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation".

Similarly, Gantz and Reinsel [11] defined big data using the four V's (Volume, Variety, Velocity, and Value) in 2011. In 2012, the characteristic "Veracity" was introduced as the fifth characteristic of big data [12]. The five most common characteristics of big data are illustrated in Fig. 2.

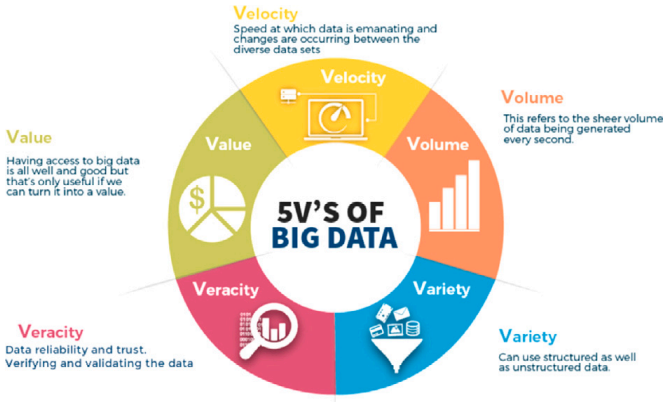

Fig. 2. 5V's of big data [13].

"Volume" is the large amount of data generated daily around the world, coming from various kinds of sources such as audio, videos, messages, social media, etc [14]. "Velocity" refers to how fast the data is generated, stored, and analyzed to enable better decision making. This increase can be observed in real-time on the Internet Stats website. "Variety", as well as the volume, the data variety is very large. There are structured data, in sequential files, databases, and non-data structured, which are the contents coming from different kinds of sources. "Value"—one of the most important features of Big Data is related to the aggregate value obtained in decision making based on previously analyzed data by the organizations [14]. Finally, "Veracity" deals with a high volume, velocity, and variety of data, it is not possible to guarantee 100% of correct information. Thus, the quality of data can vary. The data accuracy of the analysis depends on the veracity of the source data [15].

2.4. Apache Hadoop

The Apache Hadoop software library is a framework that allows the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each one of them offering local computation and storage. Rather than rely on hardware to deliver high-availability (HA), the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures [16].

Created by Goug Cutting and Mike Cafarella in 2005 for supporting a distributed search Engine Project, the Apache Hadoop is an open-source Java Framework that helps to store, access, and gain large resources from big data in a distributed fashion at less cost, high degree of fault tolerance and high scalability [17]. Hadoop is a technology that meets the needs of big data. It is based on the simple programming model MapReduce from Google. The Hadoop software is part of the Apache project, composed of some other important projects (Fig. 3). Large corporations such as IBM, Intel, and Oracle also rely on an extension of Hadoop support for their big data solutions [17].

The base Apache Hadoop framework is composed of the following modules shown in Fig. 4.

Munde & Jahan [18] describe each part of the Hadoop framework as follows:

- **Hadoop Common Files**—Contains Java libraries and utilities. These libraries provide file system and OS-level abstractions. Also contains the necessary Java Archive files and scripts to start the Hadoop;
- **Hadoop Distributed File System (HDFS)**—It is a distributed file system that stores data on commodity machines, providing very high bandwidth across the cluster. It provides high fault tolerance and native support of large data sets and stores data on commodity hardware;
- **Map-Reduce**—Implementation of the MapReduce programming model for large-scale data processing, it is used for parallel processing of large datasets;
- **Yet Another Resource Negotiator (YARN)**—The resource-management platform responsible for managing computing resources in clusters and using them for scheduling user's applications;

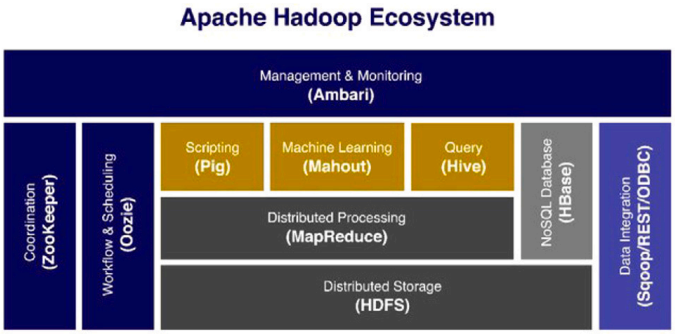

Fig. 3. Apache Hadoop ecosystem [16].

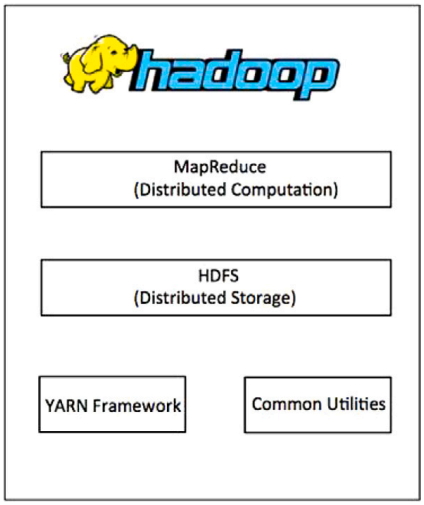

Fig. 4. Apache Hadoop framework [16].

3. State of the art

Lee et al. [19] have explored the challenges and possibilities of the latest generation Raspberry Pi for cluster-based big data processing. They built a cluster of 5 Raspberry Pi 4B nodes with a separate master node (6 nodes total) and installed Apache Hadoop. The authors performed several tests to evaluate the performance of a single Raspberry Pi, as well as the performance of the developed cluster. They used some of the main benchmarks, for example, WordCount, Terasort, TestDFSIO, and the Pi Quasi-Monte Carlo, and compared the Raspberry Pi 4B cluster to both a Raspberry Pi 3B cluster and a desktop PC for a more objective performance evaluation. Besides the authors studied the performance impact on data storage using three different types of devices: Micro Storage Device Card (MicroSD Card) of 32 GB; MicroSD Card of 64 GB; and Universal Flash Storage (UFS) of 256 GB. They concluded that using a newer version of the hardware, the cluster has a better processing capacity, as well as the use of the UFS device to storage has a better ratio (Input/Output—I/O) regarding the use of MicroSD Card.

Similarly, Scolati et al. [20,21] proposed and implemented a low-power and low-cost cluster to apply common models and technologies from the big data domain. The authors implemented the system using a cluster of 8 Raspberry Pi 2 and Docker to containerize and deploy an Apache Hadoop data streaming processing cluster. They evaluated the performance, showing that by using containerization increased fault tolerance and ease of maintenance can be achieved.

Haiji and Tso [3] presented a set of experiments to test the performance of a single node and a cluster of 12 Raspberry Pi 2 boards with and without environment virtualization using Docker to study its feasibility for real-time big data analytics. The authors evaluated the performance using the Wordcount e Terasort benchmarks and concluded that in a virtualized environment, CPU and memory consumption becomes higher, network throughput decreases, and burstiness occurs less often and less intensively.

Srinivasan et al. [22] presented the Apache Hadoop combined with the Raspberry Pi cluster to compute the big data generated via feature point extraction on an image. The Map-Reduce operation in Hadoop and the salient features of Raspberry Pi 3B help to perform the experiment more efficiently compared to a high-processing single computer. However, the same cannot be said when they deal with a smaller dataset. The authors concluded that the low cost of the Raspberry Pi clusters and the scalable Hadoop environment can be used for a variety of complex operations in this era of big data. To make the monitoring of cluster resources, they used the Prometheus tool.

Turana et al. [23] analyzed the DNA sequence with a cluster Hadoop on Raspberry Pi B (5 slaves and 1 master). They used the native Biodoop library and the Wordcount benchmark for the tests and used the Ganglia tool for monitoring the cluster resources.

The authors performed a comparison between the cluster Raspberry Pi B and 6 Desktops and concluded that it is possible to use an alternative hardware cost-effective way to implement a Hadoop cluster, even with the disadvantage of increased time to complete the compared experiments.

Bourhnane et al. [5] implemented a comparison in the use of Apache Hadoop with experiments using a single desktop and a cluster of Raspberry Pi 3B+ with 5 nodes (1 master and 4 slaves). The goal of their research was to add motivation to the use of efficient and ecological computing practice, that is, green computing. To do that, tests were performed with the benchmarks TestDFSIO and Terasort to analyze the performance and energy efficiency of the cluster. The experiments showed significant performance with TestDFSIO compared to the traditional cluster. However, the Terasort benchmark provided lower performance, which can be easily overcome by adding more nodes to the cluster. In energy consumption tests on the Raspberry Pi cluster, the low energy consumption was proven, proving to be a green computing experiment.

Table 1 presents a summary of the main characteristics observed in related works using Raspberry-based low-cost clusters and Apache Hadoop. There it is possible to compare the use of hardware platforms, the number of nodes used in the experiments, the benchmarks used for evaluating the performance and behavior of the cluster, and the tools (monitoring and helper) in each one of the main references found in this research.

We highlight our work and compare it to other relevant papers. Thus, it is possible to observe that our paper use hardware platforms more updated, and varies the number of boards used in the cluster to 2, 4, and 8 nodes. It presents a step-by-step model of how to develop and configure the cluster, which can help the community as it facilitates the assembly and use of these clusters. We use the benchmarks more used in the Big data community, such as Terasort and TestDFSIO benchmarks. We also use tools (Zabbix and Grafana) that allow better visualization of the performance and behavior, in real-time, of the different nodes and the cluster as a whole. Finally, in this paper, we change the size of the files and observe the impact of using larger sizes in the applications, and show different metrics (such as CPU usage, memory used, network stats, and node's temperature) during its execution in real-time.

Table 1
Summarizing the main related works.

Reference	Hardware	Nodes (Slaves)	Step-by-step	Benchmarks	Monitoring tool	Helper tool
Lee et al. [19]	Raspberry Pi 4B, Raspberry Pi 3B	5	No	Terasort, TestDFSIO, Pi Quasi-Monte Carlo, Wordcount	X	X
Scolati et al. [20,21]	Raspberry Pi 2B	4	No	X	Prometheus, Grafana	Docker
Haiji and Tso [3]	Raspberry Pi 2B	11	No	Wordcount, Terasort	X	Docker
Srinivasan et al. [22]	Raspberry Pi 3B	4, 5, 6, 7, 8, 9	No	Surf	X	X
Turana et al. [23]	Raspberry Pi B	5	No	Wordcount	Ganglia	X
Bourhnane et al. [5]	Raspberry Pi 3B+	4	No	TestDFSIO, Terasort	X	X
This work	Raspberry Pi 4B, Raspberry Pi 3B+'	2, 4, 8	Yes	TestDFSIO, Terasort	Zabbix, Grafana	X

4. Cluster setup

In this section, all resources (hardware and software) used for low-cost cluster development are presented.

4.1. Hardware structures

The developed cluster is composed of 9 Raspberry Pis 4B, 1 working as master server and the other 8 working as slaves. Each Raspberry Pi used a 16 GB memory card and so we improve the cluster storage capacity. All these Raspberry are connected through a Switch of 16 ports.

To monitor the resources of this cluster, it was used 1 Raspberry Pi 3B+ as a monitoring server. The software and tools used to monitor the cluster are discussed in Section 4.2.

In Table 2 all used hardware components are presented in a detailed form and can be observed in Fig. 5. The unit prices were obtained in a market, in April 2022.

The configurations of the Raspberry Pi 4B used as nodes cluster and Raspberry Pi 3B+ used as monitoring server are shown in Table 3.

Table 2
Hardware components used in low-cost cluster.

Hardware	Amount	Unit price	Total cost
Case	10	$3	$30
Memory card	10	$5	$50
Raspberry Pi 3B+	1	$35	$35
Raspberry Pi 4B	9	$55	$495
Switch	1	$35	$35
UTP cable	10	$1	$10
Total	–	–	**$655**

Fig. 5. Hardware structures of the cluster.

Table 3
Configurations of Raspberry Pis used in the cluster.

	Raspberry Pi 4B	Raspberry Pi 3B+
Ethernet/Network	1 GB	1 GB
On board RAM memory	4 GB	4 GB
Processor (CPU)	Cortex A-72 Quad Core x64 (1.5 GHz)	Cortex A-53 Quad Core x64 (1.2 GHz)
Power source	5 V/3 A USB-C	5 V/3 A USB-C
Storage	16 GB	16 GB

4.2. Software platforms

The operating system used in all Raspberry Pis was the Raspberry Pi OS.[1] The big data platform chosen was the Apache Hadoop.[2] To make the cluster monitoring, it was used the Zabbix[3] to collect and store the metrics and to show and visualize the performance results we used the visual interface of Grafana.[4]

In Table 4 all used software and tools are presented in a detailed form. It is important to highlight that all software and tools used are open and free versions.

Table 4
Softwares and tools used on cluster development.

Software	Version
Apache Hadoop	3.2.0
Grafana	8.1.5
Java Development Kit (JDK)	1.8.0_212
Raspberry Pi OS	5.10
Zabbix Agent	5.4.6
Zabbix Zerver	5.4.4

[1] https://www.raspberrypi.com/software/.
[2] https://hadoop.apache.org/.
[3] https://www.zabbix.com/.
[4] https://www.grafana.com/.

5. Cluster development—Step-by-step

To start the cluster configuration is mandatory the formatting of memory cards in fetch format (FAT32), installation of operating system in all used devices, and the correct version of Java Development Kit (JDK) required for the Apache Hadoop version (Table 4).

The user used in the configuration of devices was the user **pi**, the OS user default, with root permission.

This section is divided into three parts. The first part (From Sections 5.1 to 5.5) defines the basic configurations. The second part contains the configuration of the Apache Hadoop cluster with a single node (Section 5.6), and the last part explains the configuration of the Apache Hadoop cluster with multiples nodes (Section 5.7)

5.1. Network configuration

The first step to cluster development is the network configuration. Initially, it is necessary to set a fixed IP in each device. So, to do this, we open the terminal and write the following command: **sudo nano -w /etc/dhcpcd.conf**. In this file, we can put the network settings of the device.

```
interface eth0
static ip_address=192.168.0.20/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8
```

Code 1: Network Configuration.

For the changes to take effect, a device reboot is required. The reboot can be executed by the following command: **sudo reboot**. It is important to highlight that the IP addresses might be different. In this study, it was used the network configuration described in Table 5.

Table 5
Raspberry IPs Settings.

Raspberry name	IP
pimaster	192.168.0.20
pinode1	192.168.0.22
pinode2	192.168.0.24
pinode3	192.168.0.26
pinode4	192.168.0.28
pinode5	192.168.0.30
pinode6	192.168.0.32
pinode7	192.168.0.34
pinode8	192.168.0.36

To ensure correct operation, it is used the command in terminal mode: **ssh pi@Device IP Adress**. The **yes option** will be chosen. Next, the user's password is inserted. The next time that needs to connect it, it will be necessary to insert just the user credentials.

To insert the created SSH credentials, it is used the command: **sudo nano -w ~/.ssh/config**.

5.2. Configuring SSH connection

The configuration of SSH communication is mandatory to establish the connection between the devices in the same network without the need to insert the user credentials. For this happens, it is necessary to edit the **~/.ssh/config** file and insert the hostname, the host address, and the user who manages the cluster. This configuration must be done in all devices composing the cluster.

For the changes to take effect, a device reboot is required.

```
Host pimaster
HostName 192.168.0.20
User pi
Host pinode1
HostName 192.168.0.22
User pi
Host pinode2
HostName 192.168.0.24
User pi
Host pinode3
HostName 192.168.0.26
User pi
Host pinode4
HostName 192.168.0.28
User pi
Host pinode5
HostName 192.168.0.30
User pi
Host pinode6
HostName 192.168.0.32
User pi
Host pinode7
HostName 192.168.0.34
User pi
Host pinode8
HostName 192.168.0.36
User pi
```

Code 2: SSH Connection.

5.3. Hosts configuration

By default, all devices have a different hostname than those they were configured in Section 5.2. Using the command **sudo nano -w /etc/hostname**, we can change the hostname default to the hostname used previously. It is important to note that each raspberry used in the cluster will have a unique hostname that will let identify it. The nomenclature used in this study can be observed in Sections 5.1 and 5.2.

5.4. Creating public and private access keys

The creation of public and private keys allows one to access the cluster devices without inserting the user credentials, facilitating access and making a secure SSH connection. For that, it is necessary to execute in each device the following command: **ssh-keygen -t ed25519**.

5.4.1. Concatenating the public keys in authorized keys directory

In each slave device, we can login it and execute the command:

cat ~/.ssh/id_ed25519.pub | ssh pi@192.168.0.20 'cat >> .ssh/authorized_keys'.

This command will concatenate the public keys created in the previous section with the keys in the **authorized_keys** directory, based on Raspberry master (IP Address 192.168.0.20).

In Raspberry master, the command **cat .ssh/id_ed25519.pub >> .ssh/authorized_keys** will be executed to finish the keys concatenating. In this way, it is possible to login into the raspberry master without passing the access credentials.

5.4.2. Sync SSH configuration

To replicate the SSH access without credentials access in raspberry slaves, it is necessary execute in raspberry master the command used in previously section, changing the master IP Address to corresponding slave IP address **cat ~/.ssh/id_ed25519.pub | ssh pi@192.168.0.XX 'cat >> .ssh/authorized_keys'**.

Now is possible to login into the raspberry master without passing the access credentials. To finish the synchronization allowing to connect any raspberry without access credentials, in raspberry master we can execute the command: **scp ~/.ssh/authorized_keys raspname:~/.ssh/authorized_keys**, changing **raspname** to corresponding name of raspberry slave (**pinode1**, **pinode2**...).

5.5. Creating helpers scripts

In this section, we will prepare some scripts that will be useful when building the low-cost cluster. To do this, in a Raspberry master we will use the command **nano -w ~/.bashrc** and insert the following code at end of the file:

```
# Get the slaves hostname (pinodes)
function nodes {
  grep "pi" /etc/hosts | awk '{print $2}' | grep -v $(hostname)
}
#Execute a command in all nodes of cluster.
function cluster_cmd {
    for pi in $(nodes); do ssh $pi "$@"; done
        $@
}
#Reboot the cluster
function cluster_reboot {
    cluster_cmd sudo shutdown -r now
}
#Sent a file to cluster
function cluster_scp {
  for pi in $(nodes); do
     cat $1 | ssh $pi "sudo tee $1" > /dev/null 2>&1
     done
}
```

Code 3: Cluster Helper Functions.

For the changes to take effect, a device reboot is required. The reboot can be executed by the following command: **sudo reboot**. Finally, let us synchronize the cluster so that all nodes have access to the created helpers using the command **source ~/.bashrc && cluster_scp ~/.bashrc** in the **pimaster**.

5.6. Creating a Hadoop cluster with a single node

In this section, it is created a single-node configuration in Raspberry master. In this configuration, the **pimaster** will work as a master and slave at the same time in this cluster.

5.6.1. Download and extract of Apache Hadoop

First of all, we will download and extract the downloaded file of the Hadoop version using the following command:

```
cd && wget https://archive.apache.org/dist/hadoop/common/hadoop-3.2.0/hadoop-3.2.0.tar.gz
sudo tar -xvvf hadoop-3.2.0.tar.gz -C /opt/
rm hadoop-3.2.0.tar.gz
cd /opt
sudo mv hadoop-3.2.0 hadoop
```

Code 4: Download and Extract of Apache Hadoop.

After this process, we give root permission of this directory to the user pi through the command **sudo chown pi:pi -R /opt/hadoop**.

5.6.2. Editing the environment variables

Environment variables help programs know what directory to install files in, where to store temporary files, and where to find user profile settings. They help to shape the environment that the programs on your computer use to run. Thus, using the command **sudo nano -w /.bashrc** and at the end of the file, it is inserted the following code:

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

Code 5: Environment Variables : .bashrc.

And run **sudo nano -w /opt/hadoop/etc/hadoop/hadoop-env.sh** and at end of file, insert the following code:

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
```

Code 6: Environment Variables : hadoop-env.sh.

For the changes to take effect, a new device reboot is required. The reboot can be executed by the following command: **sudo reboot**. To check if the installation was completed successfully, run the command **hadoop version**.

5.6.3. Hadoop CORE configuration

This section defines the mandatory Hadoop file settings for the cluster to work correctly.

The first step is to identify who will be the file system master. Using the command **sudo nano -w /opt/hadoop/etc/hadoop/core-site.xml** will be inserted the code below, identifying the **pimaster** as master.

```
<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://pimaster:9000</value>
   </property>
</configuration>
```

Code 7: Hadoop Core : core-site.xml.

Continuing the Hadoop Core configuration, in this step the datanode and namenode directory paths will be defined. Using the command **sudo nano -w /opt/hadoop/etc/hadoop/hdsf-site.xml** will be inserted the code below, identifying the datanode and namenode paths. Furthermore, in **dfs.replication** property defines how many nodes will compose the data file system. In the first moment, the cluster will be used only 1 node.

```
<configuration>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///opt/hadoop_tmp/hdfs/datanode</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///opt/hadoop_tmp/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

Code 8: Hadoop Core : hdfs-site.xml.

The **mapred-site.xml** file is one of the important configuration files which is required for runtime environment settings of a Hadoop. It contains the configuration settings for MapReduce. In this file, it is specified a framework name for MapReduce, by property the MapReduce.framework.name. Using the command **sudo nano -w /opt/hadoop/etc/hadoop/mapred-site.xml**, this file will be changed, inserting the following code:

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
     </property>
</configuration>
```

Code 9: Hadoop Core : mapred-site.xml.

To finish the Hadoop CORE configuration, the **yarn-site.xml** file should be edited. This file contains the configuration settings related to YARN. For example, it contains settings for Node Manager, Resource Manager, Containers, and Application Master. Using **sudo nano -w /opt/hadoop/etc/hadoop/yarn-site.xml**, the file will be modified using the following code:

```
<configuration>
     <property>
         <name>yarn.nodemanager.aux-services</name>
         <value>mapreduce_shuffle</value>
     </property>
     <property>
        <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
     </property>
</configuration>
```

Code 10: Hadoop Core : yarn-site.xml.

Again, for the changes to take effect, a new device reboot is required. The reboot can be executed by the following command: **sudo reboot**.

Fig. 6. Cluster services running.

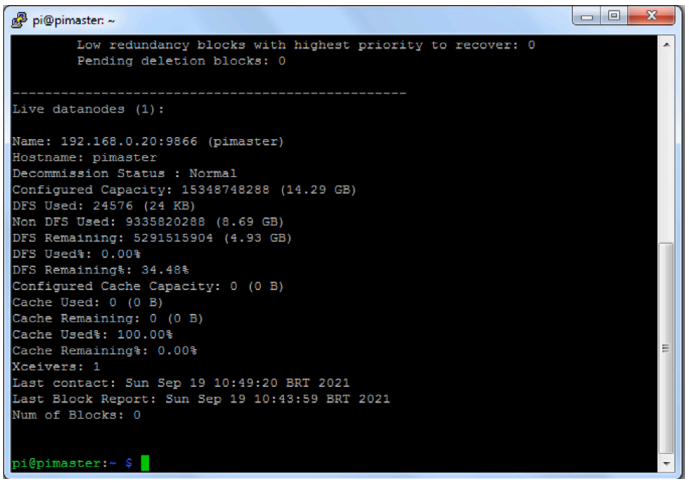

Fig. 7. Cluster single node status.

5.6.4. Formating HDFS

In **hdfs-site.xml** file, it was defined the path of datanode and namenode directory. However, these folders are not created. So, it is necessary to create them. For that, the following commands should be used:

```
sudo mkdir -p /opt/hadoop_tmp/hdfs/datanode
sudo mkdir -p /opt/hadoop_tmp/hdfs/namenode
sudo chown pi:pi -R /opt/hadoop_tmp
```

Code 11: Creating Namenode and Datanode.

After directory creation, the HDFS formatting is done by running the command **hdfs namenode -format -force**.

5.6.5. Hadoop initialization

Finally, start the cluster services using the following commands:

```
$HADOOP_HOME/sbin/start-dfs.sh
$HADOOP_HOME/sbin/start-yarn.sh
```

Code 12: Cluster Initialization.

To verify if the services are running, the command **jps** is executed. A screen similar to Fig. 6 will be displayed. Using the command **/opt/hadoop/bin/hdfs dfsadmin -report**, it is possible to check if the cluster with a unique node is working correctly. After the command execution, it will display a screen similar to Fig. 7. It is also possible to monitor the cluster using its interface web through URL: **http://pimaster:9870** or **http://192.168.0.20:9870**.

To stop the cluster services, the following commands are used:

```
HADOOP_HOME/sbin/stop-dfs.sh
HADOOP_HOME/sbin/stop-yarn.sh
```

Code 13: Stoping cluster services.

5.7. Creating a Hadoop cluster with a multiple nodes

After the creation of the single node cluster, in this section, the complete cluster will be configured.

5.7.1. Creating Hadoop directories and set user permission

Using the helper functions created in Section 5.5 will be created the Hadoop directories in the other cluster nodes, beyond to give root permission to **pi** user to access it. To do this, it is necessary to execute the following commands below:

```
cluster_cmd sudo mkdir -p /opt/hadoop_tmp/hdfs
cluster_cmd sudo chown pi:pi -R /opt/hadoop_tmp
cluster_cmd sudo mkdir -p /opt/hadoop
cluster_cmd sudo chown pi:pi /opt/hadoop
```

Code 14: Creating Hadoop directories and set permissions in all cluster nodes.

5.7.2. Installing Hadoop to all nodes cluster

Using the command **for pi in $(nodes); do rsync -avxP $HADOOP_HOME $pi:/opt/; done**, the Hadoop files are copied to all slaves Raspberry of cluster.

5.7.3. Hadoop CORE configuration

In this section, some files edited in Section 5.6.3 will be changed once again. This is because now the cluster is composed of more than one node and these changes are necessary to make it work properly. The edited files will have the codes below followed by a brief explanation of these changes.

In the **hdfs-site.xml**, the property **dfs.replication** will have its value **1** replaced with **3**. This value corresponds to the number of replication to the blocks in the DataNodes. This study used three (3) nodes.

```
<configuration>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///opt/hadoop_tmp/hdfs/datanode</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///opt/hadoop_tmp/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>3/value>
    </property>
</configuration>
```

Code 15: Hadoop Core : hdfs-site.xml.

In **mapred-site.xml** file will be added the property **mapreduce.application.classpath**, that identify the correct path of Hadoop applications and its algorithms default. The new configuration can be viewed in the code below.

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>
    <value>
    $HADOOP_HOME/share/hadoop/mapreduce/*, $HADOOP_HOME/share/hadoop/mapreduce/lib/*,
    $HADOOP_HOME/share/hadoop/common/*, $HADOOP_HOME/share/hadoop/common/lib/*,
    $HADOOP_HOME/share/hadoop/yarn/*, $HADOOP_HOME/share/hadoop/yarn/lib/*,
    $HADOOP_HOME/share/hadoop/hdfs/*, $HADOOP_HOME/share/hadoop/hdfs/lib/*
    </value>
  </property>
</configuration>
```

Code 16: Hadoop Core : mapred-site.xml.

5.7.4. Identifying the master and slave nodes

After Hadoop CORE changes, it is necessary to create 2 files that will identify the master node and in another file the slave nodes of the cluster. The files will be created in **$HADOOP_HOME/etc/hadoop/** path.

To create the file that will identify the master node, it is used the command

nano -w $HADOOP_HOME/etc/hadoop/master and to slave nodes run the command

nano -w $HADOOP_HOME/etc/hadoop/workers. The content to be inserted in these files is described below.

```
pimaster
```

Code 17: Master node identified.

```
pinode1
pinode2
pinode3
pinode4
pinode5
pinode6
pinode7
pinode8
```

Code 18: Slave nodes identified.

Then the file will be copied to the other nodes of the cluster using the command **cluster_scp /etc/hosts**. For the changes to take effect, a new cluster reboot is required again. The reboot can be executed using the helper function: **cluster_reboot**.

5.7.5. Formatting HDFS

The HDFS formatting is done running the command **hdfs namenode -format -force**.

5.7.6. Hadoop initialization

Finally, for starting the cluster services, are used the following commands in **pimaster**:

```
$HADOOP_HOME/sbin/start-dfs.sh
$HADOOP_HOME/sbin/start-yarn.sh
```

Code 19: Cluster Initialization.

To verify if the services are running, execute the command **jps**. A screen similar to Fig. 6 will be displayed. Using the command **/opt/hadoop/bin/hdfs dfsadmin -report**, it is possible to check if the cluster with a unique node is working correctly. After the command execution, a screen similar to Fig. 7 appears, but now shows all 8 nodes used in this cluster. It is also possible to monitor the cluster using its interface web through the URL: **http://pimaster:9870** or **http://192.168.0.20:9870**.

6. Benchmarks

A big data benchmark aims to generate application-specific workloads and tests capable of processing big data sets to produce meaningful evaluation results [24]. Some of the main benchmarks are available by default on Apache Hadoop installation and can be found in the path folder **opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar**. In this study, we used the Terasort and TestDFSIO benchmarks.

6.1. The Terasort benchmark

The Terasort is one of the widely used benchmarks for Hadoop. It measures the time to sort a different number of 100-byte records, stimulating and stressing almost every part of the Hadoop MapReduce framework as well as the HDFS file system making it an ideal choice to fine-tune the configuration of a Hadoop cluster [16]. For example: assuming the need to generate 1 GB of data, the number of lines of 100 bytes will be 10^7, since 1 GB = $10^7 * 100$ bytes = 10^9 bytes.

The used implementation of Terasort for Hadoop was included in the Apache Hadoop examples package. This code was used to win the annual general-purpose terabyte sort benchmark in 2008 [25]. There are three steps involved in Terasort benchmarking suite:

- Generating the input data via **Teragen**;
- Running the actual **Terasort** on the input data;
- Validating the sorted output data via **Teravalidate**;

6.1.1. Terasort usage

To generate random data, the following command is used:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar teragen <number of 100-byte rows> <input dir>
```

Code 20: Terasort Benchmark : Teragen.

The output of **Teragen** will be used and sorted by **Terasort**. To sort the generated data, the following command is used:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar terasort <teragen-output dir> <output dir>
```

Code 21: Terasort Benchmark : Terasort.

The **Teravalidade** use the **Terasort** output, to validate the correctly data sort. To ensure the data was sorted correctly, the following command is used.

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar teravalidate <terasort-output dir> <output dir>
```

Code 22: Terasort Benchmark : Teravalidate.

6.2. The TestDFSIO benchmark

The TestDFSIO is a benchmark test included in the Hadoop distribution, used to measure the aggregate ratio of data from a cluster and the throughput (data transfer rate), comparing the cluster performance during the writing and reading processes [16].

The TestDFSIO testing process works as follows: the recording program initiates multiple Map jobs, with each job written to separate HDFS files. The reader program initializes several Map jobs, reading the files written into HDFS by the writing program, with each job read sequentially. Subsequently, measurements of the tasks are performed.

The TestDFSIO test uses a single Reduce task to measure and calculate two performance metrics for each Map task: Average I/O Rate and Throughput. In this way, bottlenecks and performance problems in the configuration of the network, hardware, operating system, and Hadoop itself can be identified and corrected.

6.2.1. TestDFSIO usage

To execute the TestDFSIO benchmark, it is used the following command, choosing the mode of execution (writing or reading), the number of data files, and the size of each data file:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar TestDFSIO <-write/-read> -nrFiles <number of files> -fileSize
↪  <file zize (MB)>
```

Code 23: TestDFSIO Benchmark.

7. Cluster testing

In this section, the developed cluster is tested using the benchmarks described in Section 6 to ensure its correct functioning.

The benchmarks were executed in the cluster with 3 different slave nodes quantity. In the first execution, the cluster was working with 2 slave nodes. In the second, with 4 slave nodes and the last execution, the cluster was complete, with 8 slave nodes. In all executions, four (4) different data file sizes were used: 250 MB, 500 MB, 750 MB, and 1 GB.

Executions with different configurations are due to the need to obtain a better analysis of the cluster's performance and visualize its scalability influences and its performance.

7.1. Cluster testing using Terasort benchmark

Initially, folders should be created to store the data generated by the Terasort benchmark. In this experiment, output folders were created as follows:

- Teragen—**/opt/hadoop/test_cluster/teragen**;
- Terasort—**/opt/hadoop/test_cluster/terasort**;
- Teravalidate—**/opt/hadoop/test_cluster/teravalidate**;

Thus, it was used the command described in the code below:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar teragen <number of 100-byte rows>
↪  /opt/hadoop/test_cluster/teragen
```

Code 24: Generating data using Teragen.

After the data generation, the Terasort execution is performed using the command:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar terasort /opt/hadoop/test_cluster/teragen
↪  /opt/hadoop/test_cluster/terasort
```

Code 25: Sorting the data generated using Terasort.

Finally, the validation of data sorted is done using the command:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar teravalidate /opt/hadoop/test_cluster/terasort
↪  /opt/hadoop/test_cluster/teravalidate
```

Code 26: Data validation of sorting using Teravalidate.

7.2. Cluster testing using TestDFSIO benchmark

The cluster testing using the TestDFSIO benchmark was realized using 10 files for both operations. The write operation was done using the code below:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar TestDFSIO -write -nrFiles 10 -fileSize <file zize (MB)>
```

Code 27: TestDFSIO Benchmark.

And to realize the test for the read operation, we used the code:

```
hadoop jar opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar TestDFSIO -read -nrFiles 10 -fileSize <file zize (MB)>
```

Code 28: TestDFSIO Benchmark.

8. Benchmark results

The results of the benchmarks executed in the previous section (Section 7) are presented and discussed in this Section.

8.1. Terasort results

8.1.1. Teragen results

Table 6 shows the performance of the cluster during the execution of tests involving the Teragen benchmark.

As the file's size increases, there is a greater effort on the part of the cluster with a smaller capacity of nodes, to carry out the processing of those tasks. This effort is observed through the time spent by the CPU to execute those tasks, the traffic generated on the network, as well as the consumption of RAM memory, which increases considerably as the size of the files increases since it needs more processing for those tasks. This is because the cluster with a smaller capacity of nodes has few aggregated hardware resources, that is, fewer machines to process a greater volume of tasks (blocks), thus requiring a greater consumption of CPU and RAM for the processing of tasks, increasing the real-time of processing in the execution of these tasks.

As the cluster's capacity increases, by scaling the number of machines (nodes), it is observed that the map tasks are broken down and distributed more efficiently among the nodes, reducing the expense of RAM and CPU, and gradually reducing the time spent on the execution and processing of tasks. Therefore, it is necessary to increase the number of nodes in the cluster as the size of the tasks to be processed increases, until an ideal scenario is achieved for a good performance of the cluster and this solves the application problem.

In the test's scenarios involving the Teragen benchmark, it can be concluded that the greater cluster's capacity, about the number of installed nodes and the greater size of the file to be processed, the better performance is achieved using parallelism at the hardware level.

Table 6
Results using the Teragen benchmark.

Nodes	File size (MB)	Real time (s)	Map task time (ms)	Reduce task time (ms)	CPU time (ms)
2	250	0 m 59,632 s	41.025	X	19.100
	500	1 m 11,502 s	65.115	X	34.230
	750	1 m 24,851 s	78.937	X	49.660
	1000	1 m 33,361 s	111.635	X	63.200
4	250	1 m 00,902 s	43.719	X	18.630
	500	1 m 12,761 s	71.966	X	31.700
	750	1 m 22,098 s	86.642	X	44.490
	1000	1 m 26,232 s	94.004	X	64.110
8	250	1 m 02,211 s	46.274	X	19.990
	500	1 m 06,953 s	56.463	X	38.210
	750	1 m 19,640 s	81.030	X	50.310
	1000	1 m 26,703 s	97.463	X	63.170

8.1.2. Terasort results

Using the data generated by Teragen, the Terasort benchmark ranked this data and its results can be visualized as shown in Table 7.

Regarding the real-time processing (real-time), it was observed that in all scenarios of benchmark tests performed, involving different file sizes and cluster capacities, the best performance was from the cluster formed using 8 nodes. This is due to better use of available hardware resources by the machines in the cluster, as well as a better distribution of tasks and messages exchanged by Hadoop.

The tests showed that for a greater number of nodes in the cluster, this improves the distribution of task processing among different nodes, resulting in a reduction in tasks processing time, whether map or reduce, as well as the time spent by the CPU for the processing of tasks and the use of RAM by the nodes involved in this process. In this way, it is concluded that for a cluster to have satisfactory performance, it is necessary to adequate scale the number of nodes, as we submit it to larger and more complex tasks, which require a greater effort on the part of the CPU, RAM and network interconnection.

Table 7
Results using the Terasort benchmark.

Nodes	File size (MB)	Real time (s)	Map task time (ms)	Reduce task time (ms)	CPU time (ms)
2	250	1 m 37,175 s	61.456	35.301	60.840
	500	2 m 27,918 s	204.514	64.926	115.240
	750	2 m 29,141 s	280.074	87.601	164.740
	1000	3 m 11,633 s	443.156	100.449	220.130
4	250	1 m 38,421 s	61.788	36.136	60.650
	500	1 m 57,812 s	139.157	68.671	115.180
	750	2 m 15,779 s	191.900	79.092	154.020
	1000	2 m 38,097 s	297.362	83.318	208.910
8	250	1 m 22,566 s	36.892	32.190	57.160
	500	1 m 49,398 s	112.674	56.195	112.690
	750	2 m 06,802 s	144.245	69.600	153.750
	1000	2 m 32,163 s	226.526	103.355	203.100

8.1.3. Teravalidate results

To finish, it was used the validation of the data sorted by Terasort using the Teravalidate and the results are shown in Table 8.

As the size of nodes in the cluster increases and the size of the files to be broken and processed becomes larger, we found that the exchange of messages among nodes in the network, for data validation, was more intense. Consequently, it was a little longer when compared to the previous ones, due to the competition for access to the network by the nodes and the amount of information to be exchanged, and this causes a small delay in the execution of these tasks. In another appointment, as the number of tasks to be processed by the nodes increases, it requires a greater usage of RAM and the CPU, for the processing of these tasks by the nodes involved. And this contributes to a delay in the execution response time of those tasks demanded by the application.

It is worth mentioning that a cluster formed by a greater number of nodes requires a greater amount of information exchange through the interconnection network. This requires the use of high-speed equipment so that concurrency and delays on the network can be reduced, or to help reduce the potential congestion when exchanging messages between cluster nodes.

Table 8
Results using the Teravalidate benchmark.

Nodes	File size (MB)	Real time (s)	Map task time (ms)	Reduce task time (ms)	CPU time (ms)
2	250	0 m 48,590 s	11.391	7.164	8.990
	500	0 m 52,919 s	14.672	7.361	12.680
	750	0 m 55,940 s	17.585	7.567	15.930
	1000	0 m 58,863 s	21.838	7.027	19.990
4	250	0 m 52,853 s	13.039	10.075	10.480
	500	0 m 51,757 s	14.317	7.204	13.000
	750	0 m 56,935 s	18.679	7.612	16.890
	1000	0 m 59,750 s	21.900	7.516	20.770
8	250	0 m 49,865 s	11.332	7.332	9.170
	500	0 m 57,937 s	15.599	7.608	13.240
	750	0 m 57,100 s	18.537	7.539	17.220
	1000	1 m 01,050 s	22.649	7.545	21.240

8.2. TestDFSIO results

Table 9 shows the performance metrics of the clusters during the execution of tests involving the writing and reading algorithms of the TestDFSIO benchmark.

Table 9
Results using the TestDFSIO benchmark.

Nodes	Operation	File size (MB)	Throughput (MB/s)	Average I/O rate (MB/s)	Average I/O (std deviation)	Test Exec time (s)
2	Write	250	3,58	4,34	2,07	208,05
		500	2,61	2,62	0,14	276,77
		750	3,66	4,83	3,19	466,53
		1000	2,39	2,39	0,05	503,39
	Read	250	7,87	9,44	4,74	95,29
		500	8,22	8,51	1,63	123,53
		750	7,99	8,33	1,74	162,44
		1000	7,61	11,59	8,20	249,85
4	Write	250	4,69	9,48	7,86	190,43
		500	2,94	2,98	0,37	247,28
		750	2,65	2,67	0,21	356,7
		1000	3,68	4,89	2,58	617,23
	Read	250	18,04	54,49	62,01	86,27
		500	12,88	15,23	6,86	122,42
		750	11,40	19,24	13,56	182,06
		1000	9,83	10,55	3,29	185,00
8	Write	250	6,70	8,13	3,78	116,41
		500	4,05	4,17	0,75	217,39
		750	5,58	5,81	1,23	216,88
		1000	4,58	4,86	1,22	447,93
	Read	250	48,21	50,96	12,75	58,78
		500	41,68	49,99	26,15	64,82
		750	31,07	34,07	9,25	83,04
		1000	29,35	32,35	11,79	87,96

In the writing (write) and reading (read) tests involving the TestDFSIO benchmark, it was observed that the transfer rate of processed data (throughput) increases as the number of nodes in the network is scaled up. This occurs because as the size of the cluster grows and the size of the file to be processed becomes proportionally larger, so does the need to exchange messages among nodes in the network, due to the number of tasks distributed among them.

Regarding the average rate of input and output of the data (average I/O rate), very diversified scenarios were observed in the comparisons between the clusters, with no standard for analyzing the performance of this parameter. In some situations, the best scenario was for the cluster formed by a larger number of nodes, and in others, for the cluster formed by a smaller number of machines. This situation can be observed both in scenarios involving small and large files.

Therefore, regarding the TestDFSIO benchmark, it is not possible to say with certainty what is the best scenario for this type of test, but according to the data presented in Table 9, the cluster performance is a little better in the scenario with a greater number of nodes involved in the processing. This was probably due to the better distribution of tasks for processing among the nodes of the cluster in this scenario, causing the average rate of input and output of data decreases.

Regarding the input and output standard deviation rate (I/O rate std deviation), the same scenario observed about the average input and output rate of data (average I/O rate) was identified, that is, it is not possible to establish a standard, concerning the performance of different types of clusters tested, due to varied situations of cluster behavior in each test's scenario.

Analyzing the execution time of the tasks (test exec time), it was observed that for a greater number of nodes in the cluster and a greater file size processed, a better cluster performance will be achieved. This occurred because the tasks were broken and distributed among a greater number of nodes in the cluster, allowing the use of more hardware resources in the processing of these tasks, running them in a shorter time interval.

Through this last metric, it was observed that the performance of a cluster is better when the number of nodes involved in its configuration is proportionally greater, as well as when the task to be processed involves a relatively large and complex size.

9. Cluster monitoring

In addition to the information that is shown in the algorithm execution logs and on the Hadoop web platform, it was necessary to have a better view of information about devices such as CPU usage, Network Status, Temperature, etc. For this, the Zabbix Agent was used to collect, the Zabbix Server to store this information, and the Grafana to display it as dashboards.

In each device of the cluster was installed the Zabbix Agent,[5] used to collect the information. A Raspberry Pi 3B+ was used as the monitoring server and the software used for storing, monitoring, and displaying the collected information was installed on this device.[6,7]

Thereby, it was possible to track this important information in real-time, besides having a historical series of them. The information collected during the tests execution is showed in Section 7 and can be observed in Figs. 8 and 9.

Fig. 8. Monitoring—pimaster.

Fig. 9. Monitoring—pinode1.

10. Conclusions and future works

This paper serves as a reference for the development of a low-cost big data cluster using Apache Hadoop, containing a complete detailed guide on installation, development, testing, and monitoring. We built and showed the performance and behavior of a cluster using 9 Raspberry Pis 4B, platforms more updated in comparison to related works. We varied the number of nodes used in the cluster and the sizes of files that fit big data applications using Terasort and TestDSFIO benchmarks. We also used a Raspberry Pi 3B+ as a monitoring server, installing the Zabbix and Grafana tools, which allowed better visualization of the performance and behavior, in real-time, of the different nodes and the cluster as a whole.

The low-cost big data cluster described in this paper had good behavior when the number of nodes was increased up to 8 nodes. It was observed that as the size of the files to be processed increases, the demand for a better distribution of tasks among the nodes, as well as the use of more nodes so that tasks can be performed in a shorter period is needed. In the tests that require a greater usage of the Map and Reduce tasks, a greater effort was observed concerning CPU and RAM. Thus, our experiments showed that

[5] Zabbix Agent — https://www.zabbix.com/download_agents?version=5.4&release=5.4.6&os=Linux&os_version=412&hardware=ppc64le&encryption=No+encryption&packaging=Archive&show_legacy=0.

[6] Zabbix Server — https://www.zabbix.com/download?zabbix=5.0&os_distribution=raspberry_pi_os&os_version=10_buster&db=mysql&ws=apache.

[7] Grafana — https://grafana.com/grafana/download?edition=enterprise&pg=get&plcmt=selfmanaged-box1-cta1.

using Raspberry Pi, combined with Apache Hadoop, can be an efficient and robust solution to achieve a low-cost big data cluster, considering its costs/benefits.

Finally, for future work, we would like to make an upgrade on the cluster, increasing the number of nodes and increasing the storage capacity besides measuring the power consumption costs. Also, to evaluate the cluster using other benchmarks from other applications, as well as to get metrics about the performance in real big data scenarios and other large applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

We would like to thank the Federal Institute of Sergipe (IFS) — Campus Itabaiana for making available all Raspberry Pis used in this study and the Universidade Federal de Sergipe (Federal University of Sergipe) - Brazil for its support. We also appreciate the relevant comments and suggestions from all reviewers who helped and improved the quality of this study.

References

[1] Ordonez Carlos, Al-Amin Sikder Tahsin, Zhou Xiantian. A simple low cost parallel architecture for big data analytics. In: Proc. IEEE international conference on big data, big data 2020. 2020, p. 2827–32.
[2] Hennessy John L, Patterson David A. Computer architecture: a quantitative approach. Elsevier; 2011.
[3] Hajji Wajdi, Tso Fung Po. Understanding the performance of low power raspberry pi cloud for big data. Electron (Switzerland) 2016;5(2).
[4] Alves Neto Antonio Jose, Carneiro Neto Jose Aprigio, Moreno Ordonez Edward David. Low-cost clusters on big data - a systematic study. In: Proceedings of the 11th Euro American conference on telematics and information systems. New York, NY, USA: Association for Computing Machinery; 2022.
[5] Bourhnane Safae, Abid Mohamed Riduan, Zine-Dine Khalid, Elkamoun Najib, Benhaddou Driss. High-performance computing: A cost effective and energy efficient approach. Adv Sci Technol Eng Syst 2020;5(6):1598–608.
[6] Giger Peter, Srikugan Sajan, Persaud Badrie L. A raspberry pi cluster for teaching big-data analytics (Master's thesis), University of Zurich; 2020.
[7] Mwasaga Nkundwe Moses, Joy Mike. Implementing micro high performance computing (μhpc) artifact: Affordable HPC facilities for academia. In: 2020 IEEE frontiers in education conference (FIE). IEEE; 2020, p. 1–9.
[8] Virtana. What are clusters - virtana glossary. 2021, Accessed: 2021-11-17. https://www.virtana.com/glossary/what-is-a-cluster/.
[9] Qureshi Basit, Koubaa Anis. On performance of commodity single board computer-based clusters: A big data perspective. In: EAI/springer innovations in communication and computing. 2020, p. 349–75.
[10] Gartner. Big data. 2022, Accessed: 18-01-2022. https://www.gartner.com/en/information-technology/glossary/big-data.
[11] Gantz John, Reinsel David. Extracting value from chaos. IDC Iview 2011;1142(2011):1–12.
[12] IBM. The 5 V's of big data. 2022, Accessed: 2021-06-20. https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/.
[13] Trickdoid. Compreendendo o big data com benefí cios, caracterí sticas e exemplos de aplicativos de big data. 2021, Accessed: 2021-11-17. https://trickdroid.org/compreendendo-o-big-data-com-beneficios-caracteristicas-e-exemplos-de-aplicativos-de-big-data/.
[14] ud-din Khan M Ali, Uddin M, Gupta Navarun. Seven v's of big data understanding big data to extract value. In: Proc. of the 2014 Zone 1 conference of the American society for engineering education. 2014, p. 1–5.
[15] Ojokoh Bolanle A, Samuel Oluwarotimi W, Omisore Olatunji M, Sarumi Oluwafemi A, Idowu Peter A, Chimusa Emile R, Darwish Ashraf, Adekoya Adebayo F, Katsriku Ferdinand A. Big data, analytics and artificial intelligence for sustainability. Sci Afr 2020;9:e00551.
[16] Apache Software Foundation. Apache hadoop. 2022, https://hadoop.apache.org.
[17] Saraladevi B, Pazhaniraja N, Paul P Victer, Basha M S Saleem, Dhavachelvan P. Big data and Hadoop-a study in security perspective. Procedia Comput Sci 2015;50:596–601.
[18] Munde Kusum, Jahan Nusrat. Hadoop architecture and applications. Int J Innov Res Sci Eng Technol (An ISO 2007;3297.
[19] Lee Eunseo, Oh Hyunju, Park Dongchul. Big data processing on single board computer clusters: Exploring challenges and possibilities. IEEE Access 2021;9:142551–65.
[20] Scolati. Remo, Fronza. Ilenia, El Ioini. Nabil, Samir. Areeg, Pahl. Claus. A containerized big data streaming architecture for edge cloud computing on clustered single-board devices. In: Proc. of the 9th intl. conference on cloud computing and services science,. SciTePress, INSTICC; 2019, p. 68–80.
[21] Scolati Remo, Fronza Ilenia, El Ioini Nabil, Samir Areeg, Barzegar Hamid Reza, Pahl Claus. A containerized edge cloud architecture for data stream processing. Commun Comput Inf Sci 2020;1218(May):150–76.
[22] Srinivasan Kathiravan, Chang Chuan Yu, Huang Chao Hsi, Chang Min Hao, Sharma Anant, Ankur Avinash. An efficient implementation of mobile raspberry pi Hadoop clusters for robust and augmented computing performance. J Inform Process Syst 2018;14(4):989–1009.
[23] Turana Jaya Sena, Sukoco Heru, Kusuma Wisnu Ananta. Hadoop performance analysis on raspberry pi for DNA sequence alignment. TELKOMNIKA (Telecommun Comput Electron Control) 2016;14(3):1059–66.
[24] Tay YC. Data generation for application-specific benchmarking. Proc VLDB Endow 2011;4(12):1470–3.
[25] Nowicki Marek. Comparison of sort algorithms in Hadoop and PCJ. J Big Data 2020;7(1).

Antônio José Alves Neto is a master degree student at the UFS. He received a bachelor's in Computer Science at the UFS in 2017. He is an Analyst Big Data and Data Science Developer at the Treasury Office of Sergipe. His research interests include big data, data analysis, data engineering, and data science.

José Aprígio Carneiro Neto received a Ph.D. in Intellectual Property Science at UFS in 2018. He is a M.Sc. in Software Engineering from the Center for Advanced Studies and Systems of Recife - C.E.S.A.R. EDU - Brazil, in 2013. He is a professor at IFS in Computer Science.

Edward David Moreno received the M.Sc. and Ph.D. in Electrical Engineering from the University of São Paulo (USP) - Brazil. He is an associate professor at the UFS. He published more than 200 papers and participated in several events as Program Committee member. He is IEEE and ACM senior member and participated on the editorial board of several journals.