ARTIGO

DOI: https://doi.org/10.21728/p2p.2025v11n2e-7277

Data de submissão: 23/09/2024 Data de aprovação: 09/01/2025 Data de publicação: 16/01/2025

IMPLEMENTAÇÃO DE *CLUSTERS* DE BAIXO CUSTO E ALTO DESEMPENHO COM *RASPBERRY PI* E *APACHE HADOOP*

Um mapeamento sistemático

João Fernandes Santos Filho[1]
Instituto Federal de Sergipe
joaofilho8274@gmail.com

José Aprígio Carneiro Neto[2]
Instituto Federal de Sergipe
jose.neto@ifs.edu.br

Resumo

Os avanços tecnológicos têm levado a indústria de computadores ao desenvolvimento de novas arquiteturas de computadores, mais velozes e robustas, denominadas de arquiteturas paralelas, compostas por computadores com múltiplos processadores, que colaboram entre si na solução de problemas complexos, explorando diferentes níveis de paralelismo. Uma alternativa economicamente viável, utilizada para atender a elevada demanda de processamento de dados, bem como os custos com gastos de energia e os problemas causados pelo custo de aquisição e manutenção de supercomputadores, é a utilização de *clusters*, formados por aglomerados de computadores, homogêneos e/ou heterogêneos, que trabalham de forma distribuída e colaborativa, efetuando o processamento de diversas tarefas paralelas simultâneas, se comportando como um único sistema. Entretanto, para tirar o máximo de eficiência de um *cluster* é necessário o uso de uma biblioteca de paralelismo, que seja responsável pela comunicação e troca de mensagens entre os equipamentos desse sistema, trabalhando de forma simultânea e distribuída na execução de tarefas complexas, que seja eficiente, confiável, escalável e tolerante a falhas. Além disso, é imprescindível a utilização de uma plataforma de *hardware* com elevado poder de processamento, que seja de baixo custo de aquisição e possua uma excelente eficiência energética, como é o caso da *Raspberry Pi*, uma plataforma *SBC* (*Single Board Computer*) que possui todos os componentes integrados em uma única placa. Diante desse contexto, este artigo teve por objetivo realizar um mapeamento sistemático de estudos relacionados com a implementação de *clusters*, que utilizaram como plataforma de *hardware* a *Raspberry Pi* e o *Apache Hadoop* como biblioteca de paralelismo. A metodologia utilizada nesta pesquisa teve um caráter quantitativo, descritivo e exploratório, visando o aprofundamento do tema pesquisado.

Palavras-chave: *clusters*; alto desempenho; paralelismo; *Raspberry PI*; *Apache Hadoop*.

[1] Graduando em Ciência da Computação pelo Instituto Federal de Sergipe – IFS (2024). Técnico em Manutenção e Suporte em Informática pelo Instituto Federal de Sergipe – IFS (2020).

[2] Pós-Doutor em Engenharia e Computação Inteligente pelo Instituto Politécnico do Porto – ISEP/IPP, em Porto, Portugal (2024). Pós-Doutor em Engenharia de Produção e Sistemas pela Universidade do Minho – UNIMINHO, em Braga, Portugal (2023). Pós-Doutor em Ciência da Computação pela Universidade Federal de Sergipe - UFS (2022). Doutor em Ciência da Propriedade Intelectual pela Universidade Federal de Sergipe – UFS (2018). Mestre em Engenharia de Software pelo Centro de Estudos e Sistemas Avançados do Recife – C.E.S.A.R. EDU (2013). Especialista em Tecnologias da Informação, com ênfase em Cliente/Servidor, pela Universidade Federal do Ceará – UFC (2001). Graduado em Formação Pedagógica em Informática pelo Centro Universitário Leonardo Da Vinci – UNIASSELVI (2020). Graduado em Processamento de Dados pela Universidade Estadual do Piauí – UESPI (1997).

Esta obra está licenciada sob uma licença
Creative Commons Attribution 4.0 International (CC BY-NC-SA 4.0).

IMPLEMENTING LOW-COST, HIGH-PERFORMANCE CLUSTERS WITH RASPBERRY PI AND APACHE HADOOP

A systematic mapping

Abstract

Technological advances have led the computer industry to develop new, faster and more robust computer architectures, called parallel architectures, consisting of computers with multiple processors that collaborate to solve complex problems, exploring different levels of parallelism. An economically viable alternative, used to meet the high demand for data processing, as well as energy costs and the problems caused by the cost of acquiring and maintaining supercomputers, is the use of clusters, formed by groups of homogeneous and/or heterogeneous computers, which work in a distributed and collaborative manner, processing several simultaneous parallel tasks, behaving as a single system. However, to get the most out of a cluster, it is necessary to use a parallelism library, which is responsible for communication and message exchange between the equipment in this system, working simultaneously and distributed in the execution of complex tasks, which is efficient, reliable, scalable and fault tolerant. Furthermore, it is essential to use a hardware platform with high processing power, low acquisition cost and excellent energy efficiency, such as the Raspberry Pi, an SBC (Single Board Computer) platform that has all components integrated on a single board. Given this context, this article aimed to carry out a systematic mapping of studies related to the implementation of clusters, which used the Raspberry Pi as the hardware platform and Apache Hadoop as the parallelism library. The methodology used in this research was quantitative, descriptive and exploratory, aiming at deepening the research topic.

Keywords: clusters; high performance; parallelism; Raspberry PI; Apache Hadoop.

IMPLEMENTACIÓN DE CLÚSTERES DE ALTO RENDIMIENTO Y BAJO COSTO CON RASPBERRY PI Y APACHE HADOOP

Un mapeo sistemático

Resumen

Los avances tecnológicos han llevado a la industria informática a desarrollar nuevas arquitecturas de ordenadores más rápidas y robustas, llamadas arquitecturas paralelas, consistentes en ordenadores con múltiples procesadores que colaboran para resolver problemas complejos, explorando distintos niveles de paralelismo. Una alternativa económicamente viable, utilizada para satisfacer la alta demanda de procesamiento de datos, así como los costos energéticos y los problemas ocasionados por el coste de adquisición y mantenimiento de supercomputadoras, es el uso de clústeres, formados por grupos de ordenadores homogéneos y/o heterogéneos, que trabajan de forma distribuida y colaborativa, procesando varias tareas paralelas simultáneas, comportándose como un solo sistema. Sin embargo, para sacar el máximo provecho de un clúster, es necesario utilizar una librería de paralelismo, que se encarga de la comunicación e intercambio de mensajes entre los equipos de este sistema, trabajando de forma simultánea y distribuida en la ejecución de tareas complejas, que sea eficiente, fiable, escalable y tolerante a fallos. Además, es imprescindible utilizar una plataforma hardware con alto poder de procesamiento, bajo coste de adquisición y excelente eficiencia energética, como la Raspberry Pi, una plataforma SBC (Single Board Computer) que tiene todos los componentes integrados en una sola placa. En este contexto, el presente artículo tuvo como objetivo realizar un mapeo sistemático de estudios relacionados con la implementación de clústeres, que utilizaron como plataforma hardware Raspberry Pi y como biblioteca de paralelismo Apache Hadoop. La metodología utilizada en esta investigación fue cuantitativa, descriptiva y exploratoria, con el objetivo de profundizar en el tema de investigación.

Palabras clave: clústeres; alto desempeño; paralelismo; Raspberry PI; Apache Hadoop.

1 INTRODUÇÃO

Os avanços tecnológicos têm contribuído para o crescimento do volume de dados e da complexidade das tarefas realizadas pelos computadores, fazendo com que o poder de processamento desses equipamentos aumentasse de forma considerável ao longo dos anos (Lima *et al.*, 2016).

Porém, essa busca acelerada e constante tem deixado a indústria desse setor em alerta, devido às limitações impostas pelo aumento da frequência do *clock* dos processadores dos computadores, que eleva o consumo de energia e a produção de calor dos componentes eletrônicos (Santos, 2015), tornando-se um desafio manter um aumento contínuo no poder de processamento desses equipamentos.

Entretanto, para contornar esses problemas, causados pela limitação dos processadores, em função do aumento no quantitativo de transistores embutidos no núcleo desses componentes eletrônicos, a indústria passou a desenvolver circuitos integrados (CI´s) capazes de processar dados em paralelo, ao invés de continuarem a aumentar o poder de processamento em um único núcleo do processador, motivando assim o surgimento de diversos tipos de arquiteturas paralelas (Lima *et al.*, 2016).

As arquiteturas paralelas são estruturas de *hardware* que utilizam múltiplos processadores, trabalhando de forma colaborativa na resolução de algoritmos e problemas complexos, explorando os diferentes níveis de paralelismo do processador (Schepke, 2009).

De acordo com Bacellar (2009), uma alternativa economicamente viável para suprir a demanda elevada do processamento de grandes volumes de dados, os custos com gastos de energia e os problemas causados pelo custo de aquisição/manutenção de supercomputadores, é a utilização de *clusters*.

Os *clusters* são aglomerados de computadores, homogêneos e/ou heterogêneos, que trabalham de forma distribuída e colaborativa, efetuando o processamento de diversas tarefas paralelas, se comportando como um único sistema. Entretanto, a sua implementação deverá estar condicionada ao tipo de aplicação que será executada, tendo em vista que o seu funcionamento poderá variar conforme os problemas a serem solucionados, muitas vezes não sendo tão eficientes quanto se espera. Nesses casos, é indispensável o uso de um supercomputador.

Para montar um *cluster* é necessário ter um *hardware* com vários núcleos no processador, um sistema operacional e uma biblioteca de comunicação, que estabeleça a

comunicação entre os nós (*nodes*) que compõe o *cluster*, para efetuar o paralelismo das tarefas (Pitanga, 2004).

Durante a montagem de um *cluster* é possível escolher várias combinações de *hardware*, mas, é essencial utilizar componentes que possuam o desempenho adequado para as tarefas que serão realizadas. Dentre os componentes utilizados na implementação de um *cluster*, alguns possuem um destaque maior, pois influenciam diretamente no seu desempenho, são eles: a memória de armazenamento (memória secundária), que precisa ter altas velocidades de leitura e escrita, além de uma boa capacidade de armazenamento, visto que, armazenam grandes volumes de dados; a memória *RAM* (*Random-Access Memory*), que precisa ter frequências elevadas para não prejudicar o desempenho do processador e do *cluster*, tendo em vista que será responsável pelo armazenamento temporário dos processos durante a execução dos algoritmos de *benchmark*; a placa-mãe (*motherboard*), que precisa suportar a frequência máxima que as memórias *RAM* trabalham; e por fim, o processador, que precisa ter vários núcleos e uma frequência de processamento elevada.

Com base em estudos recentes, observa-se que os processadores baseados na arquitetura *ARM* (*Advanced RISC Machine*) ganharam destaque na implementação de *clusters* de alto desempenho (*High-Performance Computing – HPC*) e, em *benchmarks* da área computacional, motivado pelo alto poder de processamento desses processadores, pela arquitetura baseada em *RISC* (*Reduced Instruction Set Computer*), pelo baixo custo de aquisição/manutenção e pela eficiência energética (Kalapothas *et al*., 2023).

Os processadores *ARM* possuem uma arquitetura do tipo *RISC* (*Reduced Instruction Set Computer* - Conjunto Reduzido de Instruções) de 64 *bits*, uma *pipeline* em três estágios e registradores do tipo *load/store*, fazendo com que o processador não fique ocioso e esteja sempre executando diferentes estágios de instruções. Além disso, esses processadores possuem sete modos diferentes de execução, que são utilizados para o tratamento de exceções e de interrupções, entre outras funcionalidades. Além dessas características, esses processadores possuem ainda (Campbell Júnior, 2019): registradores de uso geral de 64 *bits*, *SP* (*stack pointer*) e *PC* (*program counter*); processamento de dados de 64 *bits*; endereçamento virtual estendido; e estados de execução que suportam três conjuntos de instruções principais: *AArch32* (conjunto de instruções de tamanho fixo de 32 *bits*); *T32*: (*Thumb* e *Thumb-2*); e *AArch64* (conjunto de instruções de comprimento fixo de 64 *bits*).

Por serem baseados na arquitetura *RISC*, os processadores *ARM* possuem um conjunto de instruções mais simplificada (reduzido), tornando-os processadores menos complexos e com um excelente desempenho energético (baixo consumo de energia), ideal para a criação de

soluções computacionais embarcadas e de alto desempenho (*HPC - High-Performance Computing*) (Zomaya; Lee, 2012).

Dentre os diversos tipos de equipamentos que utilizam arquitetura *RISC* e processadores do tipo *ARM*, o destaque com relação a implementação em soluções de *clusters* de alto desempenho e baixo custo, é para a plataforma *Raspberry Pi*, uma plataforma *SBC* (*Single Board Computer*), ou seja, um tipo de computador que possui todos os componentes integrados em uma única placa. Esse tipo de computador possui algumas características importantes, tais como: portabilidade, baixo custo de aquisição, baixo consumo de energia e um poder de processamento elevado, tornando-o interessante e acessível para a implementação de *clusters* de alto desempenho (*HPC*) e de baixo custo (Ariza; Baez, 2022; Basford *et al*., 2020).

Para tirar o máximo de eficiência do poder de processamento dos nós (*nodes*) que compõem um *cluster*, torna-se indispensável o uso de uma biblioteca da programação paralela, responsável pela comunicação e troca de mensagens entre os *nodes*, que trabalhe de forma simultânea e distribuída na execução de tarefas complexas, envolvendo grandes volumes de dados e atuando de forma eficiente no gerenciamento dos processos, além de oferecerem uma abstração para a criação e o gerenciamento de processos, sendo ainda confiáveis, escaláveis e tolerante a falhas, como é o caso da biblioteca *Apache Hadoop* (Ma; Zhao; Zhao, 2023).

Nesse contexto, a programação paralela surge como uma importante ferramenta para a solução de problemas computacionais que envolvem um grau elevado de complexidade, visando a exploração de diferentes técnicas de paralelismo, bem como o uso de bibliotecas que possam trabalhar com a execução explícita de processos ou *threads* de forma simultânea e distribuída.

Porém, o uso das bibliotecas de paralelismo traz alguns desafios, entre eles: a utilização de diferentes tipos de primitivas de comunicação e sincronização; as abstrações de processos ou *threads* distintas; os diferentes modelos de programação; os problemas de balanceamento de carga; e um maior impacto no desenvolvimento e desempenho das aplicações desenvolvidas, em função do tipo de plataforma e de biblioteca paralela utilizada na solução dos problemas computacionais.

Portanto, este artigo teve por objetivo realizar um mapeamento sistemático sobre os estudos relacionados à implementação de *clusters* de baixo custo e alto desempenho, que utilizam como plataforma de *hardware* a *Raspberry Pi* (*RPi*) e o *Apache Hadoop* como biblioteca de paralelismo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ARQUITETURAS PARALELAS

As arquiteturas paralelas são estruturas de *hardware* que utilizam múltiplas unidades de processamento paralelo, que cooperam entre si na resolução de problemas complexos, demandando um maior poder de processamento por parte dos computadores, explorando diferentes níveis de paralelismo (Schepke, 2009).

Essas arquiteturas podem ser classificadas e organizadas de acordo com suas características e aplicações, a saber: máquinas *multicore*, multiprocessadores, multicomputadores (*clusters*), vetorial e em sistema de *grid*, formado por computadores organizados em um sistema de grade, disponíveis em escala mundial na Internet (Navaux; Rose; Pilla, 2011).

Uma arquitetura multiprocessada consiste em um conjunto de processadores operando sobre um conjunto de memória, sendo que essas memórias podem ser acessadas de maneira uniforme, na mesma latência e compartilhada entre todos os processadores (*Uniform Memory Access - UMA*) ou não uniforme (*Non-Uniform Memory Access - NUMA*), onde a memória será dividida em vários módulos e cada processador possui um deles. Nesse caso, o tempo de acesso à memória poderá variar (Navaux; Rose; Pilla, 2011).

A arquitetura de multicomputadores pode ser vista como uma arquitetura constituída por um conjunto de computadores independentes, interconectados através de uma rede de computadores, onde cada computador possui sua própria unidade de processamento (CPU) e memória, sendo que a memória de um computador não poderá ser acessada diretamente por outro computador da arquitetura (Andrews, 2001; Navaux; Rose; Pilla, 2011).

Uma arquitetura vetorial tem por objetivo prover instruções de alto-nível, através da utilização de vetores de dados, permitindo dessa forma a realização de operações aritméticas em um vetor ou entre vetores, sem a necessidade da implementação de um laço (*loop*) no programa, como ocorre nas arquiteturas não vetoriais (Dongarra; Duff; Sorensen; Van Der Vorst, 1991; Navaux; Rose; Pilla, 2011).

As arquiteturas *multicore*, por sua vez, são formadas por processadores que possuem múltiplos núcleos de processamento. Dessa forma, múltiplos programas ou fluxo de execuções podem ser executados de forma simultânea e distribuída, conforme a quantidade de núcleos existentes na máquina. Sendo assim, é possível obter um nível maior de desempenho dos processadores durante a execução das tarefas. Além disso, os recursos da memória *cache* podem ser compartilhados entre os diferentes núcleos de processamento (Gepner; Kowalik,

2006; Navaux; Rose; Pilla, 2011).

Já uma arquitetura em *grid* (grade), consiste em uma enorme infraestrutura de *hardware* e *software*, capaz de prover um elevado poder de processamento para a execução de aplicações que envolvem grandes volumes de tarefas e recursos computacionais (Bernholdt, 2007; Navaux; Rose; Pilla, 2011). Porém, a construção desse tipo de arquitetura enfrenta vários desafios, entre eles: o gerenciamento dos recursos; a grande heterogeneidade de máquinas e de conexões de rede; a grande escalabilidade; o controle de acesso aos administradores e usuários; e as interfaces de programação (Schepke, 2009), além da largura de banda utilizada na rede (Internet) e o tipo de infraestrutura de rede utilizada nesse tipo de arquitetura.

2.2 ARQUITETURA EM *CLUSTER*

Uma das formas de explorar o paralelismo de *hardware* é através da construção de sistemas que realizam o processamento paralelo por meio da associação de dois ou mais computadores (*nodes*), conectados entre si por meio de uma rede local de computadores (*LAN - Local Area Network*), denominados de *clusters* (Costa, 2007).

Os *clusters* são sistemas fracamente acoplados, onde cada computador possui sua própria memória individual, vistos como um sistema único e coerente, realizando o gerenciamento de memória necessário para manter a consistência das atividades computacionais a serem realizadas. Além disso, são uma forma de contornar os problemas relacionados aos custos elevados de aquisição e manutenção de supercomputadores, tornando-se uma alternativa viável para o desenvolvimento de pesquisas, dentre outras atividades na área computacional (Costa, 2007).

O objetivo de um *cluster* é obter o máximo de poder de processamento dos computadores (*nodes*) conectados a ele, necessário para a execução de tarefas complexas, que envolvem grandes volumes de dados. Em um *cluster*, existe um computador principal denominado de mestre (*master*), responsável por coordenar a execução das tarefas e controlar os outros computadores, denominados de nós (*nodes*), ou *slaves*, que cooperam entre si para a realização das tarefas (Bacellar, 2009).

Para realizar a comunicação entre os *nodes* e a distribuição das tarefas, é necessário a utilização de um *software* (uma biblioteca de paralelismo) que seja capaz de fornecer a ideia de um sistema único para o usuário, totalmente transparente, responsável por permitir a comunicação entre todos os *nodes* que compõe o *cluster*, bem como a troca de mensagens entre eles, através de uma interconexão de rede (Bacellar, 2009).

A implementação de um *cluster* deverá levar em consideração a aplicação a ser executada, tendo em vista que esse tipo de sistema não é tão eficiente quanto um supercomputador, no caso da resolução de problemas que envolvem uma granulosidade fina (decomposição do problema em um grande número de pequenas tarefas). No entanto, para problemas que envolvem uma granulosidade grossa (decomposição do problema em um pequeno número de grandes tarefas), os *clusters* têm se mostrado tão eficientes quanto os supercomputadores (Bacellar, 2009).

O motivo dos *clusters* não terem se mostrado muito eficientes para tratar problemas de granulosidade fina, se deve ao fato da necessidade de troca de informações entre o *node* mestre e os *nodes slaves*. Esse tipo de atividade provoca um aumento significativo no tráfego da rede, interferindo de forma direta no tempo de execução das tarefas, comprometendo o objetivo proposto pelo paralelismo, que é o ganho de tempo na execução das atividades. Já no caso de problemas que envolvem granulosidade grossa, onde é exigido um elevado poder de processamento de dados, os *clusters* têm se tornado uma solução viável, uma vez que os *nodes* recebem, de forma independente, uma carga elevada de dados para processar, diminuindo a troca de informações na rede entre os *nodes* (Bacellar, 2009). Além disso, em um *cluster*, todo o gerenciamento de memória compartilhada, necessário para manter o sistema computacional operando de forma consistente, é realizado de maneira confiável e segura (Lima *et al.*, 2016).

Na computação de alto desempenho (*HPC*), os *clusters* são amplamente utilizados por possuírem acesso restrito e dedicado, além de serem uma arquitetura confiável, escalável e flexível. A gestão de um *cluster* envolve diversos fatores, que vão desde a instalação do sistema operacional até a definição de ferramentas para sua configuração, manutenção, monitoramento e agendamento de tarefas (Schepke, 2009). Além disso, os *clusters* possuem uma série de benefícios que contribuem para sua utilização em *HPC*, tais como: desempenho, expansibilidade, baixo custo, alta disponibilidade, balanceamento de carga e tolerância a falhas (Bacellar, 2009; Lima *et al.*, 2016).

Na literatura, são identificados diferentes tipos de *clusters*, onde cada modelo possui suas próprias características e situações em que podem ser aplicados, a saber:

> Alto desempenho: Consiste no agrupamento de computadores para a obtenção de um elevado poder de processamento, permitindo a execução de algoritmos mais complexos, que podem processar grandes volumes de dados em um curto espaço de tempo (Nascimento; Souza, 2020);

Balanceamento de carga: Utilizado para dividir as requisições de um sistema, onde as solicitações são divididas entre os *nodes* que fazem parte do *cluster*. Dessa forma, todos os recursos são utilizados igualmente, mantendo um baixo tempo de resposta das tarefas executadas (Sharma, 2020);

Alta disponibilidade: Utilizado para garantir que um serviço permaneça sempre ativo. Sendo assim, todos os *nodes* que fazem parte do *cluster* devem possuir as mesmas informações, e caso o computador principal (*master*) apresente uma falha, um outro computador será designado para manter a disponibilidade do sistema, até que o problema seja corrigido (Sharma, 2020).

2.3 BIBLIOTECAS DE PARALELISMO

Para resolver problemas computacionais em paralelo, os códigos (algoritmos) devem ser escritos de forma a explorar ao máximo o paralelismo, exigindo o domínio de técnicas de programação paralela, bem como o uso de bibliotecas que suportam o paralelismo, tais como: a *MPI* (*Message Passing Interface*), a *OpenMP* (*Open Multi-Processing*), a *PVM* (*Parallel Virual Machine*), o *Apache Hadoop*, entre outras.

Em um *cluster*, a biblioteca de paralelismo é responsável por gerenciar os recursos disponíveis no sistema, realizar a comunicação entre os nós (*nodes*) e coordenar a execução dos algoritmos paralelos. Cada biblioteca possui suas próprias características e peculiaridades, permitindo a obtenção do melhor aproveitamento dos recursos computacionais do *cluster*, sendo implementada e utilizada de forma distinta na resolução de cada problema, a fim de garantir que as suas funcionalidades sejam adequadas à necessidade da tarefa a ser executada (Barker, 2015; Santos, 2013).

2.4 *APACHE HADOOP*

O *Apache Hadoop* é uma biblioteca de paralelismo de código aberto (*open-source*), escrita em *JAVA* (linguagem de programação), utilizada na execução de tarefas que necessitam de um elevado poder de processamento dos computadores, para a manipulação de grandes volumes de dados (*big data*). Essa biblioteca é responsável por gerenciar os recursos de todos os computadores que compõem o sistema do *cluster*, bem como coordenar a execução dos algoritmos de processamento de dados (Menezes; Freitas; Parpinelli, 2016).

As vantagens da utilização desta biblioteca são: o fato dela ser uma ferramenta de código aberto (*open-source*); a economia na sua implantação, por não ser necessário gastos

com licenças e contratação de pessoal especializado; sua robustez, oferecendo estratégias de recuperação automática, no caso de situações de falhas nos equipamentos; escalabilidade, podendo aumentar a quantidade de máquinas utilizadas no processamento dos dados, quando necessário, sem a necessidade de grandes alterações na codificação; e simplicidade, pelo fato de retirar do programador a responsabilidade de ter que gerenciar questões referentes à computação paralela, tais como: tolerância a falhas, escalonamento e balanceamento de carga, ficando essas tarefas a cargo da própria ferramenta (Goldman *et al*., 2012).

O *Apache Hadoop* pode ser instalado em qualquer máquina *Linux*, com no mínimo 4 *GBytes* de memória *RAM* (*Random Access Memory*), podendo ser implementada em servidores ou em *clusters*, caracterizando-se como uma ferramenta de código aberto, econômica, robusta e escalável (Apache Hadoop, 2024).

Porém, o *Apache Hadoop* apresenta algumas desvantagens, entre elas destacam-se: o fato da ferramenta estar em constante evolução, não conseguindo dar suporte à todas as situações que está sendo submetida, devido a algumas de suas funcionalidades não estarem maduras; apesar de poder utilizar várias máquinas como *nodes*, para o processamento e armazenamento dos dados, possui apenas um único *node* mestre, o *master*, criando dessa forma um ponto crítico suscetível a falha, vital para o funcionamento da aplicação; dificuldade no gerenciamento dos *nodes* e na depuração de falhas, bem como na análise dos *logs*, que encontram-se distribuídos entre as diferentes pastas do ecossistema *hadoop*; problemas não paralelizáveis ou com alta dependência entre os dados, que possuem dificuldade de divisão de tarefas em tarefas menores, devido a impossibilidade de extração da porção paralelizável da aplicação, não sendo portanto, bons candidatos para o *Apache Hadoop*; e o processamento de arquivos pequenos (Goldman *et al*., 2012).

O *Apache Hadoop* possui semelhanças com os sistemas de arquivos distribuídos existentes, porém, é altamente tolerante a falhas e foi projetado para implementações em *hardware* de baixo custo, como a *Raspberry Pi*. Além disso, fornece uma taxa elevada de transferência de acesso aos dados da aplicação, sendo adequado para trabalhar com aplicações que envolvem grandes volumes de dados, como é o caso de *Big Data* (Apache Hadoop, 2024).

Sua estrutura possui alguns elementos essenciais para a seu funcionamento, tais como: o *HDFS* (*Hadoop Distributed File System*), sistema responsável pelo armazenamento de arquivos; e o *MapReduce*, estrutura responsável por processar os dados, além de alguns outros componentes que auxiliam durante a execução e processamento dos algoritmos paralelos (Goldman; *et al*., 2012; Bhosale; Gadekar, 2014).

A biblioteca *Apache Hadoop* é um *software* que fornece confiabilidade para o armazenamento de dados, possuindo uma estrutura de gerenciamento simplificada, pois a própria ferramenta é responsável por administrar a comunicação entre os *nodes* do *cluster* e a distribuição durante o processamento dos dados, além de possuir escalabilidade, pois não são necessárias grandes modificações nas configurações para adicionar mais *nodes* (computadores) ao *cluster*.

Entretanto, é possível encontrar alguns problemas ao utilizá-la, tais como: dificuldade para encontrar falhas, pois durante a execução das tarefas são criados vários arquivos de *logs*, que ficam espalhados em diferentes pastas do sistema; o processamento de arquivos pequenos, pois a biblioteca não é bem adaptada para esse tipo de situação; e a existência de apenas um nó mestre, que cria um ponto crítico para o funcionamento do *cluster* (Goldman *et al.*, 2012).

3 METODOLOGIA

A metodologia utilizada no desenvolvimento deste artigo, considerando sua natureza e abordagem, é classificada como uma pesquisa básica pura (focada na ampliação da base de conhecimento científico sobre um determinado tema de pesquisa (Lopes, 1991)) e quantitativa (focada na análise de dados brutos, utilizando instrumentos padronizados para a coleta de dados, onde os resultados poderão ser quantificados através de técnicas estatísticas (Mussi *et al.*, 2019).

Quanto aos objetivos, está classificada como uma pesquisa descritiva (descrevendo as características de uma determinada população ou fenômeno, bem como o estabelecimento de relações entre as variáveis, utilizando técnicas padronizadas para a coleta de dados (Gil, 2008), podendo observar os fatos, registrá-los, analisá-los, classificá-los, compará-los e interpretá-los, sem a interferência do pesquisador (Marcondes *et al.*, 2022)) e exploratória (visando o aprofundamento do tema pesquisado, tornando-o mais claro, levando a construção de questões importantes para a condução da pesquisa (Raupp; Beuren, 2006)).

Com relação aos procedimentos técnicos, foi realizado um mapeamento sistemático da literatura (identificando um conjunto de estudos finalizados sobre um determinado tema, avaliando os resultados desses estudos e evidenciando suas conclusões (Hulley; Cummings; Grady, 2015)), através de um levantamento do estado da arte, proporcionando uma visão panorâmica dos principais resultados das publicações acadêmicas (artigos científicos, teses, dissertações, livros, entre outros), utilizando-se de diversas bases de dados, tais como: *Scopus*, *Web of Science* e *Science Direct*, entre outras.

Por fim, os resultados coletados foram transportados para uma planilha eletrônica (*Microsoft Excel*), onde posteriormente foram tabulados, analisados e armazenados, gerando dados estatísticos que respondem às questões de pesquisa deste estudo.

Para conduzir este estudo, foi utilizada a seguinte questão principal de pesquisa: **Como desenvolver um *cluster* de baixo custo e alto desempenho utilizando a biblioteca paralela *Apache Hadoop* na plataforma computacional *Raspberry Pi*?**

Com base na questão principal desta pesquisa, foram definidas algumas questões secundárias, que contribuíram para responder à questão principal da pesquisa, a saber:

1) **QP1**: Quais algoritmos de *benchmark* foram utilizados nos testes dos *clusters*?
2) **QP2**: Qual modelo da *Raspberry Pi* foi utilizado no desenvolvimento dos *clusters*?
3) **QP3**: Qual a versão utilizada da biblioteca de paralelismo *Apache Hadoop*?
4) **QP4**: Qual a versão utilizada da Linguagem de Programação *JAVA*?
5) **QP5**: Quais *softwares* foram utilizados no gerenciamento e monitoramento dos *clusters*?
6) **QP6**: Quais métricas de desempenho foram utilizadas para a análise e avaliação dos *clusters*?

Neste mapeamento sistemático, foram utilizados os seguintes critérios para inclusão e exclusão das publicações identificadas nas buscas às bases de dados científicas:

- Critérios de Inclusão:

1) **I1:** Artigos publicados no período de 2019 a 2023;
2) **I2:** Artigos que possuíam a *string* principal de busca nos campus resumo, título ou palavras-chave, das respectivas bases de dados pesquisadas;
3) **I3:** Artigos escritos somente no idioma inglês e português;
4) **I4:** Artigos completos.

- Critérios de Exclusão:

1) **E1:** Artigos duplicados;
2) **E2:** Artigos de literatura cinza;
3) **E3:** Artigos que disponibilizaram apenas conceitos.

As buscas nas bases de dados acadêmicas da *Scopus*, *Science Direct* e *Web of Science* ocorreram durante o mês de junho de 2024, compreendendo o período de publicações entre os anos de 2019 e 2023.

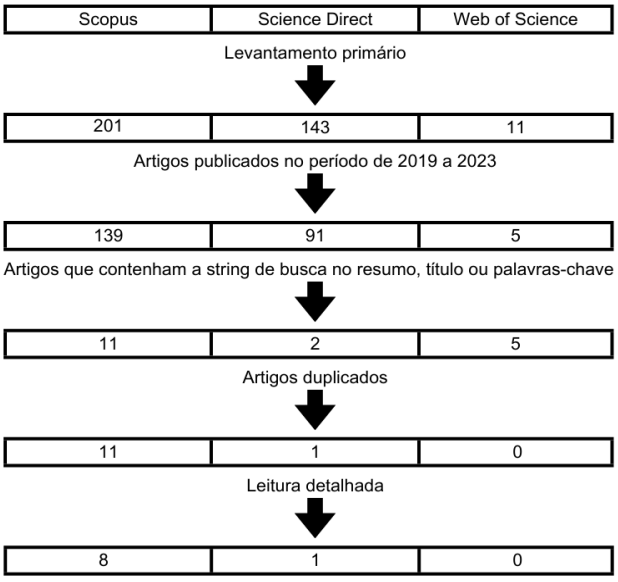
Como critério de busca, foi utilizada a expressão (*string*): "*hadoop*" *AND "cluster" AND "raspberry*" (*TITLE-ABS-KEY (“hadoop" AND "cluster" AND "raspberry") AND*

PUBYEAR > 2018 AND PUBYEAR < 2024), inserida nos campus resumo, título e palavras-chave, das respectivas bases de dados pesquisadas. Essa combinação de palavras-chave contou com a utilização de operadores lógicos (*AND* e *OR*), que foram usados para refinar as buscas, delimitando o escopo da pesquisa, de forma a garantir a relevância dos dados coletados.

Nas buscas realizadas no levantamento primário, foram identificadas 355 publicações que atenderam inicialmente o escopo da pesquisa. Desse quantitativo, 201 foram encontradas na base de dados da *Scopus*, 143 na *Science Direct* e 11 na *Web of Science*.

Na sequência, foram aplicados alguns critérios para filtrar os trabalhos mais relevantes para este estudo. Dessa forma, foram excluídos 120 documentos por estarem fora do período de publicação proposto nesta pesquisa, 217 por não apresentarem as palavras-chave da expressão de busca nos campos citados e 6 por estarem duplicados. Por fim, foi realizada uma leitura detalhada de 12 publicações, restando 9 para serem analisadas por completo, das quais 8 pertenciam a base de dados da *Scopus* e 1 a base de dados da *Science Direct*, como mostra a Figura 1.

Figura 1 – Etapas da seleção das publicações para a pesquisa.



Fonte: Elaborada pelos Autores, 2024.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nesta seção, serão apresentados e analisados os dados que respondem às questões de pesquisa norteadoras deste trabalho.

4.1 ALGORITMOS DE *BENCHMARK*

Com base nas buscas realizadas nas bases de dados citadas, foram identificados 9 tipos de *benchmarks* utilizados nos testes de desempenhos dos *clusters* implementados, sendo o *TestDFSIO* e o *TeraSort*, disponíveis no *Apache Hadoop*, os mais utilizados, como mostra a Tabela 1.

Tabela 1 – Algoritmos de *Benchmark*.

Algoritmos de *Benchmark*	**Referências**
TestDFSIO	(Alves Neto; Carneiro Neto; Moreno, 2022), (Lee; Oh; Park, 2021)
Terasort	(Alves Neto; Carneiro Neto; Moreno, 2022), (Lee; Oh; Park, 2021)
Wordcount	(Lee; Oh; Park, 2021)
Pi	(Lee; Oh; Park, 2021)
Sysbench	(Lee; Oh; Park, 2021)
Siege	(Nugroho; Widiyanto, 2020)
Regressão Linear	(Komninos *et al*., 2019)
Árvore de Decisão	(Komninos *et al*., 2019)
TPCx-HS	(Böther; Rabl, 2021)

Fonte: Elaborada pelos Autores, 2024.

O *Terasort* é um algoritmo composto por três fases, que permitem testar diferentes componentes do *cluster*. Nesse teste, o processador é intensivamente utilizado durante as operações de criação e ordenação dos dados. A transferência de dados possibilita avaliar a capacidade da rede, sendo possível analisar as velocidades de leitura e gravação do dispositivo de armazenamento utilizado.

Já o algoritmo *TestDFSIO* é utilizado para avaliar, principalmente, a velocidade de transferência de dados da unidade de armazenamento do *cluster*. Durante sua execução, são gerados grandes volumes de dados, para monitorar o desempenho de leitura e escrita do sistema (*cluster*).

A aplicação de cálculo do *Pi* é utilizada para estimar o valor do *Pi*, aplicando o método estatístico *quasi-Monte Carlo*. A execução desse algoritmo permite obter dados sobre a utilização e o desempenho do processador dos computadores utilizados no *cluster*.

O *Wordcount* é um algoritmo que permite analisar um arquivo de texto e indicar quantas vezes uma palavra foi encontrada neste arquivo. Esse algoritmo é utilizado para aplicar uma carga de trabalho intensiva ao processador, permitindo analisar o seu desempenho durante a execução da tarefa.

O *Siege* consiste em uma ferramenta utilizada para estressar um servidor, ao simular um número de usuários e a quantidade de requisições realizadas por cada um. Ele permite registrar o número de requisições, bem como o total de *bytes* transferidos, o tempo de resposta e a duração de cada requisição. A partir disso, é possível analisar o desempenho da CPU e a capacidade de transferência da rede.

Um dos tipos de teste que merece um destaque é o *Sysbench*, aplicado em um dos trabalhos encontrados durante a pesquisa, que consiste em uma ferramenta utilizada para analisar o desempenho do sistema de forma individual, ou seja, de cada máquina e dos seus respectivos componentes, onde são disponibilizados diferentes testes para executar cargas de trabalho (teste de carga), que permitem analisar o desempenho dos componentes de cada equipamento pertencente ao *cluster*, como processador e memória *RAM*.

4.2 MODELOS DE *RASPBERRY PI*

Nos trabalhos selecionados, foram utilizados 4 modelos de *Raspberry Pi* para o desenvolvimento dos *clusters*. As versões mais utilizadas foram a *4B* e a *2B*, presentes em 4 e 3 trabalhos, respectivamente. Em dois estudos foram desenvolvidos dois *clusters* utilizando diferentes modelos de *Raspberry Pi*, a *3B* e a *3B+*, para analisar a evolução do desempenho das plataformas, como mostra a Tabela 2.

Tabela 2 – Modelos de *Raspberry Pi* utilizados na implementação dos *clusters*.

Modelos de *Raspberry Pi*	**Referências**
4B	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021), (Komninos *et al*., 2019)
2B	(Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019)
3B+	(Böther; Rabl, 2021), (Nugroho; Widiyanto, 2020)
3B	(Lee; Oh; Park, 2021), (Nasir *et al*., 2019)

Fonte: Elaborada pelos Autores, 2024.

A *Raspberry Pi 2B*, lançada em 2015, possui um processador *quad-core* de 32 *bits*, com frequências de até 900 *MHz*. Essa versão possui 1 *GByte* de memória *RAM LPDDR2*, tem suporte a cartão de memória de no máximo 32 *GBytes* e a conexão de rede é *Fast Ethernet* (100 *Mbps*).

Já a *Raspberry 3B*, lançada em 2016, possui um processador *quad-core* 64 *bits*, que pode atingir até 1.2 *GHz* de frequência. Esse modelo também possui 1 *GByte* de memória *RAM* do tipo *SDRAM* e uma placa de rede com conexão *Fast Ethernet* (100 *Mbps*).

O modelo *3B+*, lançado em 2018, possui um processador *quad-core*, com uma arquitetura 64 *bits*, podendo atingir uma frequência de até *1.4 GHz*. Esse modelo é equipado com 1 *GByte* de memória *RAM* do tipo *LPDDR2* e uma placa de rede com suporte à conexão *Gigabit Ethernet* (1000 *Mbps*).

Por fim, a *Raspberry Pi 4B*, lançada em 2019, que também possui um processador *quad-core*, mas sua arquitetura é de 64 *bits*, podendo atingir até 1.8 *GHz*. Sua memória *RAM* é do tipo *LPDDR4*, com 4 modelos, sendo eles de: 1, 2, 4 ou 8 *GBytes*. O tamanho máximo do cartão de memória que esse modelo suporta é 1 *TByte* e a conexão de rede dessa *Raspberry Pi* é do tipo *Gigabit Ethernet* (1000 *Mbps*). Esse modelo de *Raspberry* foi a mais utilizada dentre os trabalhos pesquisados, pelo fato de possuir mais recursos disponíveis no seu *hardware* e, consequentemente, apresentar um melhor desempenho, tornando-se a opção ideal para o desenvolvimento de um *cluster* com a proposta apresentada neste trabalho.

4.3 VERSÃO DO *APACHE HADOOP*

De acordo com os resultados dos trabalhos analisados, foi possível identificar a utilização de três versões da biblioteca de paralelismo *Apache Hadoop*, sendo as versões 3.2.0 e 3.1.3, as mais utilizadas, presentes em dois trabalhos cada, como pode ser evidenciado na Tabela 3.

Tabela 3 – Versões do *Apache Hadoop* utilizadas na implementação dos *clusters*.

Versões do *Hadoop*	**Referências**
3.2.0	(Alves Neto; Carneiro Neto; Moreno, 2022), (Komninos *et al*., 2019)
3.1.3	(Böther; Rabl, 2021), (Lee; Oh; Park, 2021)
1.2.1	(Nugroho; Widiyanto, 2020)

Fonte: Elaborada pelos Autores, 2024.

Na versão 1.2.1 do *Hadoop*, o *JobTracker* é o componente utilizado para gerenciar os recursos do sistema e a execução das tarefas, sendo um ponto crucial de falhas, pois, caso falhe, não será possível rastrear as tarefas. Essa versão foi lançada em 2013, não utiliza os recursos disponíveis no *hardware* com eficiência e possui uma escalabilidade limitada e, por isso, era indicada para projetos mais simples.

Já a versão 3.1.3, lançada em 2019, apresenta algumas melhorias na utilização dos recursos do sistema, ao substituir o *JobTracker* pelo *YARN*. O *YARN* é um gerenciador de recursos criado com o objetivo de separar o mecanismo de processamento e a função de gerenciamento do *MapReduce* (modelo de programação que permite o processamento de

dados massivos em um algoritmo paralelo e distribuído). Ele faz o monitoramento e o gerenciamento de cargas de trabalho, dos recursos de alta disponibilidade do *Hadoop*, além de implementar controles de segurança no sistema. Com essa mudança, a escalabilidade e a eficiência da biblioteca *Hadoop* foram aprimoradas, permitindo o desenvolvimento de projetos mais avançados e complexos, para a execução de cargas de trabalho maiores, como é o caso do *Big Data*.

A versão 3.2.0, também lançada em 2019, trouxe melhorias significativas na utilização dos recursos do sistema, tais como: escalabilidade, segurança, tolerância a falhas, flexibilidade no armazenamento de dados e balanceamento de carga, proporcionando uma maior eficiência da biblioteca.

4.4 *SOFTWARES* DE MONITORAMENTO E GERENCIAMENTO

Com relação às ferramentas de monitoramento e gerenciamento de desempenho dos *clusters* implementados nos trabalhos pesquisados, foram identificados três tipos de ferramentas: o *Zabbix Server*, o *Grafana* e o *Prometheus*, sendo as duas últimas as mais utilizadas pelos autores, como mostra a Tabela 4.

Tabela 4 – Ferramentas de monitoramento e gerenciamento dos *clusters*.

***Softwares* de monitoramento e gerenciamento**	**Referências**
Grafana	(Alves Neto; Carneiro Neto; Moreno, 2022), (Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019)
Prometheus	(Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019)
Zabbix	(Alves Neto; Carneiro Neto; Moreno, 2022)

Fonte: Elaborada pelos Autores, 2024.

O *Grafana* é uma ferramenta utilizada para monitorar e visualizar os dados de um sistema. A sua interface *web* permite criar *dashboards* com vários tipos de gráficos e personalizações, bem como gerenciar as configurações disponíveis no sistema.

O *Prometheus* é uma ferramenta utilizada para o monitoramento de dados de séries temporais. Em sua estrutura, existem exportadores responsáveis por coletar métricas dos dispositivos e disponibilizá-las, para que sejam exibidas através de *dashboards*.

Já o *Zabbix* é uma ferramenta utilizada para monitorar o desempenho e a disponibilidade de dispositivos instalados em uma rede de computadores. Sua estrutura possui um servidor, chamado *Zabbix Server*, que é responsável por coletar os dados enviados através da rede e dispositivos monitorados, que podem ser configurados através do protocolo *SNMP*

(*Simple Network Management Protocol*) ou utilizando o *Zabbix Agent*, que facilita a instalação nos equipamentos e amplia a quantidade de dados que podem ser enviados para o servidor.

Todas essas ferramentas são úteis para monitorar e gerenciar a utilização dos recursos de *hardware* do sistema durante a execução das tarefas de um *cluster*, permitindo a realização de análises com base em métricas que permitem compreender, analisar e melhorar o desempenho do *cluster*.

4.5 MÉTRICAS DE DESEMPENHO

A partir da análise dos artigos selecionados, foi possível identificar a utilização de diferentes tipos de métricas de desempenho dos *clusters* implementados, com destaque para as seguintes métricas: utilização de *CPU*, utilização de *RAM*, *throughput*, tráfego de rede e *CPU time*, como mostra a Tabela 5.

Tabela 5 – Métricas de desempenho.

Métricas	**Referências**
Utilização de *CPU*	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021), (Komninos *et al*., 2019), (Scolati *et al*., 2019), (Nasir *et al*., 2019)
Throughput (MB/s)	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021), (Nugroho; Widiyanto, 2020), (Nasir *et al*., 2019)
Utilização de *RAM*	(Alves Neto; Carneiro Neto; Moreno, 2022), (Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019), (Nasir *et al*., 2019)
Tráfego de rede	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021), (Nasir *et al*., 2019)
CPU time (ms)	(Alves Neto; Carneiro Neto; Moreno, 2022), (Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019)
Real Time (s)	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021),
Temperatura	(Alves Neto; Carneiro Neto; Moreno, 2022), (Nasir *et al*., 2019)
Consumo de Energia	(Böther; Rabl, 2021), (Lee; Oh; Park, 2021)
Average I/O rate (MB/s)	(Alves Neto; Carneiro Neto; Moreno, 2022), (Lee; Oh; Park, 2021)
Map task time (ms)	(Alves Neto; Carneiro Neto; Moreno, 2022)
Reduce task time (ms)	(Alves Neto; Carneiro Neto; Moreno, 2022)
Average I/O (std deviation)	(Alves Neto; Carneiro Neto; Moreno, 2022)
Test Exec time (s)	(Alves Neto; Carneiro Neto; Moreno, 2022)
Transaction	(Nugroho; Widiyanto, 2020)
Response Time	(Nugroho; Widiyanto, 2020)
Transactions Rate	(Nugroho; Widiyanto, 2020)

Fonte: Elaborada pelos Autores, 2024.

A utilização dessas métricas permite analisar fatores que influenciam diretamente no desempenho do *cluster* durante a execução das tarefas, como processador, memória RAM, consumo de energia, temperatura e tráfego de rede. A partir das informações coletadas, é possível verificar a utilização dos principais recursos do *cluster* e realizar o gerenciamento do sistema, bem como fazer intervenções no sistema, caso seja necessário.

4.6 QUANTIDADE DE NÓS (*NODES*)

Como pode ser observado na Tabela 6, foram utilizadas diversas combinações de *hardware* para compor os nós escravos (*slaves*) dos *clusters* implementados nos trabalhos analisados. Dentre as opções de *clusters* identificadas, pode-se observar que as mais utilizadas foram formadas pelas seguintes quantidades de *nodes slaves* (nós escravos): 1, 4, 5 e 7 nós. Em quatro trabalhos analisados, foram aplicadas duas ou mais combinações de tamanho de *clusters* e, em três trabalhos, foram mencionados a implementação de um *cluster* formado por apenas uma única máquina (*cluster* de nó único), onde essa máquina se comporta, ao mesmo tempo, como nó *master* e *slave* da solução de *cluster* implementada.

Tabela 6 – Quantidade de *nodes* do *cluster*.

Quantidade de nós	Referências
1	(Lee; Oh; Park, 2021), (Nugroho; Widiyanto, 2020), (Nasir *et al*., 2019)
2	(Alves Neto; Carneiro Neto; Moreno, 2022), (Lee; Oh; Park, 2021)
3	(Lee; Oh; Park, 2021), (Nugroho; Widiyanto, 2020)
4	(Alves Neto; Carneiro Neto; Moreno, 2022), (Böther; Rabl, 2021), (Lee; Oh; Park, 2021)
5	(Lee; Oh; Park, 2021), (Nasir *et al*., 2019), (Komninos *et al*., 2019)
7	(Penchalaiah *et al*., 2022), (Scolati *et al*., 2020), (Scolati *et al*., 2019)
8	(Alves Neto; Carneiro Neto; Moreno, 2022)

Fonte: Elaborada pelos Autores, 2024.

A utilização de mais de uma combinação de máquinas em um *cluster* serve para demonstrar a evolução de desempenho do sistema, que pode ser obtida à medida que ocorre um aumento da escalabilidade dos nós do *cluster*, tornando-o mais robusto e capaz de executar tarefas de forma mais rápida e eficiente.

4.7 RESUMO DOS TRABALHOS ANALISADOS

Alves Neto, Carneiro Neto e Moreno (2022) desenvolveram um guia para a instalação de um *cluster* de baixo custo utilizando o *Apache Hadoop* e o *Raspberry Pi 4B*. Para analisar o desempenho do sistema, foram utilizados os algoritmos *Terasort* e *TestDFSIO*. Os testes

foram executados com 2, 4 e 8 nós escravos, utilizando arquivos com tamanho entre 250 *MBytes* e 1 *GByte*. Para monitorar o consumo dos recursos do *cluster* durante os testes, os autores utilizaram as ferramentas *Zabbix* e *Grafana*, instaladas em uma *Raspberry Pi 3B+*. A partir dos resultados obtidos, eles concluíram que, em razão do baixo custo e do bom desempenho da plataforma, a *Raspberry Pi* pode ser uma boa alternativa para desenvolver um *cluster* para trabalhar com *Big Data*.

Penchalaiah *et al*. (2022) desenvolveram um *cluster* utilizando a plataforma *Raspberry Pi 2B*, em conjunto com as tecnologias *Apache Hadoop*, *Spark* e *Docker*. As ferramentas selecionadas para monitorar o sistema e gerar as análises de desempenho do *cluster*, foram: o *Prometheus* e o *Grafana*. Para avaliar o desempenho do *cluster*, os autores executaram uma aplicação *Nodejs*, para criar um conjunto de arquivos e uma aplicação em *Python*, para analisar os dados que foram gerados. O objetivo deste estudo foi explorar os benefícios que podem ser obtidos ao utilizar *containers* para o desenvolvimento de *clusters*, como por exemplo, a melhoria na tolerância a falhas e na manutenção do sistema.

Böther e Rabl (2021) estudaram a viabilidade na utilização de plataformas *ARM* para o desenvolvimento de sistemas capazes de processar grandes volumes de dados (*Big Data*). Com esse objetivo, foram desenvolvidos dois *clusters*, utilizando as plataformas *Raspberry Pi 3B+* e *4B*, ambos equipados com quatro nós escravos. O desempenho dos *clusters* foi analisado com a execução do *benchmark TPCxHS*, baseado no algoritmo *Terasort*, utilizando arquivos de 1 *GByte* e 10 *GBytes*. Com base nos dados coletados, os autores concluíram que a *Raspberry Pi 4B* apresenta bons resultados, considerando a relação custo/desempenho, e que os problemas encontrados no modelo 3B+, foram resolvidos ao aumentar a quantidade de memória *RAM* disponível para o sistema.

Lee, Oh e Park (2021) estudaram os impactos causados pela velocidade do dispositivo de armazenamento no desempenho do *cluster*. Para isso, desenvolveram três *clusters* com diferentes plataformas: *Desktop*, *Raspberry Pi 3B* e *4B*. Com o objetivo de analisar o desempenho do *cluster*, os autores utilizaram, para testes de *benchmark*, os seguintes algoritmos: *Wordcount*, *Teragen*, *TestDFSIO* e *Pi*. Para obter mais informações sobre o desempenho do *cluster*, os autores também avaliaram o desempenho individual das plataformas, utilizando para isso a ferramenta *Sysbench*. Todos os testes foram realizados utilizando entre um e cinco nós escravos, associados a três tipos de armazenamento (memória secundária). Os resultados mostraram que o desempenho do *cluster* foi significativamente afetado pelos dispositivos de armazenamento utilizados.

Nugroho e Widiyanto (2020) avaliaram uma alternativa baseada em computação paralela para a implementação de servidores *IoT* (Internet das Coisas). Para isso, desenvolveram um *cluster* utilizando a *Raspberry Pi 3B+* e o *Apache Hadoop*. Os testes foram executados com o *Siege*, uma ferramenta utilizada para realizar solicitações para um servidor. Para analisar o desempenho do sistema em diferentes situações, os autores fizeram 10, 50, 100 e 150 solicitações para o *cluster*, além de terem utilizado um e três nós escravos para a montagem dos *clusters*.

Scolati *et al*. (2020) investigaram os benefícios da utilização de *containers* para o desenvolvimento de *clusters* para o processamento de dados. A partir desse objetivo, foi desenvolvido um *cluster* composto por sete nós escravos, utilizando a *Raspberry Pi 2B* como plataforma de *hardware* e os *softwares Spark*, *Docker* e *Apache Hadoop*. Além disso, os autores instalaram uma tecnologia *Blockchain*, utilizada para aumentar a confiabilidade do sistema, chamada *TOM Orchestrator*. O desempenho do *cluster* foi analisado com a execução de duas aplicações, uma delas responsável por gerar um conjunto de arquivos e a outra por analisá-los. Durante a execução dos testes, os autores utilizaram o *Grafana* e o *Prometheus* para monitorar o desempenho e o consumo dos recursos do *cluster*.

Komninos *et al*. (2019) desenvolveram um *cluster* composto por cinco nós escravos, utilizando a *Raspberry Pi 4B* e os *softwares Spark* e o *Apache Hadoop*. O propósito foi avaliar a utilização dessa plataforma em aprendizado de máquina (*machine learning*), para executar previsões em aplicações turísticas. Para avaliar o desempenho do *cluster*, os autores executaram um algoritmo de regressão linear simples, utilizando um conjunto de dados de carros usados para prever os preços de venda e um algoritmo de classificação de árvore de decisão, utilizando dados de *check-ins* para recomendar locais de visita para um usuário. A partir dos resultados, os autores concluíram que o *cluster* de *Raspberry Pi* possui um bom desempenho para o treinamento e a execução de modelos de *Machine Learning* (*ML*).

Scolati *et al*. (2019) desenvolveram um *cluster* com sete nós escravos, utilizando a *Raspberry Pi 2B* e os *softwares Docker*, *Apache Hadoop* e *Spark*. O objetivo foi avaliar uma alternativa capaz de coletar e processar dados localmente, bem como, as vantagens da utilização de *containers* no desenvolvimento de *clusters*. Os autores utilizaram as ferramentas *Prometheus* e *Grafana* para monitorar o sistema durante a execução dos testes de *benchmark*. Para analisar o desempenho do *cluster*, os autores utilizaram uma aplicação para gerar um conjunto de dados e outra para analisá-los.

Nasir *et al*. (2019) propuseram uma estrutura para processamento de vídeos de vigilância para cidades inteligentes baseada na plataforma *Raspberry Pi*. O objetivo desse

sistema era gerar um resumo dos vídeos coletados a partir das câmeras e, então, enviá-los para o armazenamento na nuvem, reduzindo o alto consumo de banda gerado pela transferência de grandes arquivos. Com essa proposta, os autores desenvolveram um *cluster* utilizando a *Raspberry Pi 3B* em conjunto com os *softwares Spark* e *Apache Hadoop*. Os testes foram realizados com a execução de um algoritmo de sintetização de vídeo, utilizando arquivos com tamanho entre 100 *MBytes* e 1 *GByte*. Com base nos resultados, os autores chegaram à conclusão de que o sistema desenvolvido se apresenta como uma solução viável para atender às necessidades de uma cidade inteligente.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi elaborado um mapeamento sistemático para identificar e compreender o desenvolvimento de *clusters* de baixo custo e alto desempenho, utilizando a plataforma de *hardware Raspberry Pi* e a biblioteca de paralelismo *Apache Hadoop*, bem como os algoritmos paralelos utilizados como *benchmark* para medir e avaliar o desempenho do *cluster*, com o auxílio de ferramentas para o monitoramento e gerenciamento desses sistemas.

Para responder às questões norteadoras desta pesquisa, foram realizadas buscas nas bases de dados científicas da *Scopus*, *Science Direct* e *Web of Science*. Durante o levantamento inicial das buscas foram identificadas 355 publicações relacionadas ao tema pesquisado, porém, após a aplicação dos critérios de inclusão e exclusão estabelecidos na metodologia deste trabalho, restaram apenas 9 publicações para serem analisadas por completo.

Com relação aos algoritmos de *benchmark* utilizados para gerar os testes nos *clusters*, foram identificados nove tipos de algoritmos, dentre eles, os destaques vão para os algoritmos *TeraSort*, *TestDFSIO* e *Siege*, presentes em dois estudos cada.

Dentre os modelos de *Raspberry Pi* utilizados na implementação dos *clusters* analisados, observa-se uma maior utilização dos modelos 2B e 4B, sendo está última um modelo mais robusto de *hardware*, com um processador de maior desempenho em relação ao modelo 2B, além de uma capacidade de memória *RAM* mais elevada. Para a implementação de um *cluster*, é importante que a plataforma de *hardware* utilizada possua características significativas em relação ao desempenho do processador e a memória *RAM*. Dessa forma, quanto maior forem esses recursos de *hardware*, melhor será o desempenho do *cluster*.

Durante a análise dos trabalhos analisados, observa-se a implementação de *clusters* formado pela combinação de 1, 4, 5 e 7 *nodes slaves* (nós escravos), sendo que no caso de

uma máquina apenas, essa se comporta tanto como *master* e *slave* da solução, não sendo muito indicada para testes com grandes volumes de dados (*Big Data*) e com tarefas mais complexas.

Em um *cluster*, a escalabilidade de máquinas permite a obtenção de um melhor resultado durante a execução dos testes, principalmente se essas máquinas forem dotadas de processadores robustos e velozes, bem como de uma boa capacidade de memória *RAM*. Além disso, cabe mencionar, que é importante, em um *cluster*, as máquinas possuírem uma memória secundária capaz de armazenar um grande volume de dados processados.

Já com relação a biblioteca de paralelismo utilizada na implementação dos *clusters* pesquisados, o *Apache Hadoop*, verifica-se a utilização de três versões: 1.2.1, 3.1.3 e 3.2.0. Cabe mencionar, que a biblioteca *Hadoop* encontra-se em constante atualização (evolução), pelo fato de ser *open source*. Por esse motivo, é interessante que os usuários procurem sempre por versão mais estável (*stable*), na hora de implementar um *cluster*, evitando assim possíveis problemas de configuração e/ou de execução das tarefas.

Em todos os estudos analisados, observa-se a utilização de ferramentas de monitoramento e gerenciamento dos *clusters*, sendo as mais utilizadas o *Grafana*, o *Prometheus* e o *Zabbix*. Essas ferramentas permitem fazer uma coleta das métricas de desempenho pré-estabelecidas, bem como uma análise, em tempo real, do que está acontecendo no *cluster*, durante a execução das tarefas.

Com base nos trabalhos pesquisados, foram evidenciados 16 tipos de métricas utilizadas para medir e analisar o desempenho dos *clusters*, com destaque para as métricas que avaliam a utilização da CPU, o tráfego da rede (*throughput*) e a utilização da memória *RAM*.

Além disso, esse estudo mostrou que os processadores baseados na arquitetura *ARM* (*Advanced RISC Machine*), como é o caso da *Raspberry Pi*, constituem uma excelente plataforma de *hardware* para a implementação de *clusters* de baixo custo e de alto desempenho.

Por fim, os resultados apresentados neste trabalho fornecem um conjunto de informações relevantes para a compreensão do tema abordado, servindo de subsídio para novas pesquisas nessa área.

Como trabalho futuro, sugere-se a implementação de *clusters* baseados em novas plataformas de *hardware*, bem como a utilização de outras bibliotecas de paralelismo, a fim de realizar estudos comparativos entre essas soluções.

REFERÊNCIAS

ALVES NETO, A. J.; CARNEIRO NETO, J. A.; 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, parte A, p. 108403, December 2022.

ANDREWS, G. R. **Foundations of Multithreaded, Parallel, and Distributed Programming**. Reading, Massachussets: Addison-Wesley, 2001. 664 p.

APACHE HADOOP. **Apache Hadoop**. 2024. Disponível em: https://hadoop.apache.org/. Acesso em: 26 abr. 2024.

APACHE HADOOP. **Apache Hadoop 3.1.3 - Release Notes.** 2024. Disponível em: https://hadoop.apache.org/docs/r3.1.3/hadoop-project-dist/hadoop-common/release/3.1.3/RELEASENOTES.3.1.3.html. Acesso em: 20 jun. 2024.

APACHE HADOOP. **Apache Hadoop 3.2.0 - Release Notes.** 2024. Disponível em: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/release/3.2.0/RELEASENOTES.3.2.0.html. Acesso em: 20 jun. 2024.

APACHE HADOOP. **Apache Hadoop 1.2.1 - Release Notes.** 2024. Disponível em: https://hadoop.apache.org/docs/r1.2.1/releasenotes.html. Acesso em: 20 jun. 2024.

ARIZA, J. A.; BAEZ, H. Understanding the role of single-board computers in engineering and computer science education: A systematic literature review. **Computer Applications in Engineering Education**, v. 30, n. 1, p. 304-329, 2022.

ASSUNÇÃO, R.; SILVA. A Implementação do Zabbix com Segurança: Um Estudo de Caso Zabbix Safely. **Revista Foco**, v. 17, n. 4, p. e4851–e4851, 11 abr. 2024.

BACELLAR, H. V. **Cluster**: Computação de alto desempenho. Campinas, São Paulo: Instituto de Computação, Universidade Estadual de Campinas, 2009.

BARKER, B. Message passing interface (mpi). *In:* WORKSHOP: High Performance Computing on Stampede. New York: Cornell University, Center of Advanced Computing, 2015.

BASFORD, P. J.; JOHNSTON, S. J.; PERKINS, C. S.; GARNOCK-JONES, T.; TSO, F. P.; PEZAROS, D.; COX, S. J. Performance analysis of single board computer clusters. **Future Generation Computer Systems**, v. 102, p. 278-291, 2020.

BERNHOLDT, D. E. Component architectures in the next generation of ultrascale scientific computing: challenges and opportunities. *In:* Compframe ’07: The 2007 Symposium on Component and Framework Technology in High-Performance and Scientific Computing, 2007, New York, NY, USA**. Proceedings […]** New York: ACM, 2007. p.1–10.

BHOSALE, H. S.; GADEKAR, D. P. A review paper on big data and hadoop. **International Journal of Scientific and Research Publications**, v. 4, n. 10, p. 1-7, 2014.

BÖTHER, M.; RABL, T. Scale-down experiments on TPCx-HS. *In:* International Workshop on Big Data in Emergent Distributed Environments - BiDEDE’21, 2021, China. **Proceedings […]** China, 2021. p. 1-6.

COSTA, R. A. G. **Desempenho e Consumo de Energia de Algoritmos Criptográficos do MiBench em Sistemas Móveis**. Amazonas: Universidade Estadual do Amazonas, 2007.

DONGARRA, J. J.; DUFF, I. S.; SORENSEN, D. C.; VAN DER VORST, H. A. **Solving Linear Systems on Vector and Shared Memory Computers**. Philadelphia, PA: SIAM, 1991.

FULMER, J. **Siege**. [S. l. : s. n.], 2024. Disponível em: https://github.com/JoeDog/siege. Acesso em: 25 jun. 2024.

GEPNER, P.; KOWALIK, M. F. Multi-core processors: New way to achieve high system performance. *In:* International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06), 2006, Poland. **Proceedings […]** Poland: IEEE, 2006. p. 9-13.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Editora Atlas SA, 2008.

GOLDMAN, A.; KON, F.; JÚNIOR, F. P.; POLATO, I.; FÁTIMA, P. R. Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. *In:* Jornadas de atualizações em informática, XXXI, 2012, Porto Alegre. **Anais [...]** Porto Alegre: SBC, 2012. p. 88-136.

HULLEY, S. B.; CUMMINGS, S. R.; BROWNER, W. S.; GRADY, D. G.; NEWMAN, T. B. **Delineando a pesquisa clínica**. 4. ed. Porto Alegre: Artmed Editora, 2015.

ISSA, J. A. Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount. **IEEE Access**, v. 3, p. 2784–2793, 2015.

CAMPBELL JUNIOR, R. M. **Avaliação de desempenho de aplicações hadoop em nuvens computacionais**. 2019. 51 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal Fluminense, Niterói, 2019.

KALAPOTHAS, S.; GALETAKIS, M.; FLAMIS, G.; PLESSAS, F.; KITSOS, P. A survey on risc-v-based machine learning ecosystem. **Information**, v. 14, n. 2, p. 64, 2023.

KOMNINOS, A.; SIMOU, I.; GKORGKOLIS, N.; GAROFALAKIS, J. Performance of raspberry pi microclusters for edge machine learning in tourism. **Network (Mbps)**, v. 100, n. 100, p. 100, 2019.

LIMA, F. D. A; MORENO, E. D.; DIAS, W. R. A. Design and Performance of a Low Cost Cluster using ARM-based Platform. *In:* International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), 22, 2016, Las Vegas, USA**. Proceedings […]** Las Vegas, USA: CSREA Press, 2016. p. 175.

LEE, E.; OH, H.; PARK, D. Big data processing on single board computer clusters: Exploring challenges and possibilities. **IEEE Access**, v. 9, p. 142551-142565, 2021.

LOPES, O. U. Pesquisa básica versus pesquisa aplicada. **Estudos avançados**, v. 5, p. 219-221, 1991.

LTD, R. P. **Buy a Raspberry Pi 3 Model B+.** 2024. Disponível em: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Acesso em: 21 jun. 2024.

MA, C.; ZHAO, M.; ZHAO, Y. An overview of Hadoop applications in transportation big data. **Journal of traffic and transportation engineering (English edition)**, v. 10, n. 5, p. 900-917, October 2023.

MARCONDES, M. M.; ARAÚJO, M. A. D.; SOUZA, W. J.; SILVA, M. G. K. Observatórios sociais e desigualdades no Brasil: Uma análise exploratória e descritiva. **Cadernos Gestão Pública e Cidadania**, v. 27, n. 86, p. 1-18, 2022.

MENEZES, S. L.; FREITAS, R. S.; PARPINELLI, R. S. Mineraçao em grandes massas de dados utilizando hadoop mapreduce e algoritmos bio-inspirados: Uma revisão sistemática. **Revista de Informática Teórica e Aplicada**, v. 23, n. 1, p. 69-101, 2016.

MUSSI, R. F. F.; MUSSI, L. M. P. T.; ASSUNÇÃO, E. T. C.; NUNES, C. P. Pesquisa Quantitativa e/ou Qualitativa: distanciamentos, aproximações e possibilidades. **Revista Sustinere**, v. 7, n. 2, p. 414-430, jul./dez. 2019.

NASCIMENTO, E. B.; SOUZA, M. S. Cluster e Balanceamento de Carga: Sistemas essenciais de alta disponibilidade. **RIOS - Revista Científica do Centro Universitário do Rio São Francisco,** v. 14, n. 27, nov. 2020.

NASIR, M.; MUHAMMAD, K.; LLORET, J.; SANGAIAH, A. K.; SAJJAD, M. Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. **Journal of Parallel and Distributed Computing**, v. 126, p. 161-170, 2019.

NAVAUX, P.; ROSE, C.; PILLA, L. Fundamentos das arquiteturas para processamento paralelo e distribuído. *In:* Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul, XI, 2011, Porto Alegre. **Anais [...]** Porto Alegre, 2011. p. 22-59.

NUGROHO, S.; WIDIYANTO, A. Designing parallel computing using raspberry pi clusters for IoT servers on Apache Hadoop. **Journal of Physics: Conference Series**, v. 1517, p. 012070, 2020.

PITANGA, M. **Computação em cluster:** o estado da arte da computação. Rio de Janeiro: Brasport, 2004.

PENCHALAIAH, N.; AL-HUMAIMEEDY, A. S.; MAASHI, M.; BABU, J. C.; KHALAF, O. I.; ALDHYANI, T. H. Clustered Single-Board Devices with Docker Container Big Stream Processing Architecture. **Computers, Materials & Continua**, v. 73, n. 3, 2022.

RASPBERRY PI LTD. **Buy a Raspberry Pi 3 Model B**. 2024. Disponível em: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/. Acesso em: 21 jun. 2024.

RAUPP, F. M.; BEUREN, I. M. Metodologia da pesquisa aplicável às ciências. **Como elaborar trabalhos monográficos em contabilidade:** teoria e prática. São Paulo: Atlas, p. 76-97, 2006.

SANTOS, A. P. C. Criar um Cluster de Processamento Paralelo MPI com Raspberrys. **Revista Programar**, mar. 2015. Disponível em: https://www.revista-programar.info/artigos/criar-um-cluster-de-processamento-paralelo-mpi-com-raspberrys/. Acesso em: 17 mar. 2024.

SCHEPKE, C. **Ambientes de programação paralela**. Trabalho Individual I. Rio Grande do Sul: Universidade Federal do Rio Grande do Sul/PPGC, 2009.

SCOLATI, R.; FRONZA, I.; EL IOINI, N.; SAMIR, A.; PAHL, C. A Containerized Big Data Streaming Architecture for Edge Cloud Computing on Clustered Single-board Devices. *In:* International Conference on Cloud Computing and Services Science - CLOSER, 9, 2019, Greece. **Proceedings [...]** Greece, 2019. v. 1. p. 68-80.

SCOLATI, R.; FRONZA, I.; EL IOINI, N.; SAMIR, A.; PAHL, C. A containerized edge cloud architecture for data stream processing. *In:* Cloud Computing and Services Science: International Conference, CLOSER, 9, 2019, Greece. **Proceedings [...]** Greece: Springer International Publishing, 2020. p. 150-176.

SHARMA, N. To Perform and Join Cluster High Availability Using Load Balancing Approach. **International Journal of Engineering Applied Sciences and Technology**, v. 5, n. 4, p. 637-640, 2020.

SILVA, G. O.; DIAS, W. R. A.; ESCUDERO, D. P. Análise do Desempenho Computacional da Raspberry Pi Executando o Benchmark SysBench. *In:* Simpósio em Sistemas Computacionais de Alto Desempenho, XXIII, 2022, Florianópolis. **Anais estendidos [...]** Florianópolis: SBC, 2022. p. 65-72.

ZOMAYA, A. Y.; LEE, Y. C. **Energy-efficient distributed computing systems**. [S.l]: John Wiley & Sons, 2012.