UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PABLO JOSÉ PAVAN

Comportamento da E/S de Aplicações Paralelas em Sistemas de Alto Desempenho

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Philippe Olivier Alexandre Navaux

Co-orientador: Profa. Dra. Francieli Zanon Boito (Université de Bordeaux)

Porto Alegre
2021

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Pavan, Pablo José

Comportamento da E/S de Aplicações Paralelas em Sistemas de Alto Desempenho / Pablo José Pavan. – Porto Alegre: PPGC da UFRGS, 2021.

69 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2021. Orientador: Philippe Olivier Alexandre Navaux; Co-orientador: Francieli Zanon Boito.

1. Caracterização da Carga de Trabalho de E/S. 2. Comportamento de E/S. 3. E/S Paralela. 4. Computação de Alto Desempenho. I. Navaux, Philippe Olivier Alexandre II. Zanon Boito, Francieli. III. Título.

“Os que se encantam com a prática sem a ciência são como os timoneiros que entram no navio sem timão nem bússola, nunca tendo certeza do seu destino.”

— LEONARDO DA VINCI

AGRADECIMENTOS

Nesses anos de mestrado, de muito estudo, esforço e empenho, gostaria de agradecer a algumas pessoas que me acompanharam nessa jornada e foram fundamentais para sua conclusão. Por tanto, expresso aqui, a minha sincera gratidão a todas elas. Primeiramente agradeço a minha família, especialmente aos meus pais Adélio e Jussara, pelo profundo apoio. Obrigado por desejarem o melhor para mim e pelo esforço que fizeram para que eu pudesse superar cada obstáculo em meu caminho e chegar aqui. Minha gratidão especial ao meu orientador Prof. Dr. Philippe Olivier Alexandre Navaux, por ter sempre acreditado e depositado confiança em mim, por me ajudar e me orientar, pois sem seu apoio e confiança nada disso seria possível. Quero também agradecer a minha co-orientadora Profa. Dra. Francieli Zanon Boito e colega Jean Bez pelo suas contribuições à este trabalho. Um obrigado especial a minha namorada Sandra e ao amigo Matheus Serpa que sempre estiveram ao meu lado, me apoiando e torcendo por mim. Quero também agradecer a todos os colegas e professores que fizeram parte da minha rotina de estudos no Instituto de Informática da UFRGS, pelo companheirismo e pela amizade construída ao longo destes anos de trabalho. Por fim, agradeço a todos pelos conselhos, sugestões e interesse em contribuir para o desenvolvimento e conclusão deste trabalho.

RESUMO

Na Computação de Alto Desempenho (CAD), um grande número de aplicações disputam pelos recursos disponíveis, principalmente pelo sistema de arquivos paralelos. Esta concorrência muitas vezes causa contenção nas operações de Entrada/Saída (E/S) reduzindo o desempenho das aplicações. Portanto, caracterizar as operações de E/S dos supercomputadores é crucial para compreender carga de trabalho de E/S e assim identificar possíveis gargalos. Uma das formas de investigar os padrões de acesso das aplicações é através do uso de *profiles* de E/S das aplicações, porém, em um sistema CAD estes geram grandes quantias de dados.

Nesse contexto, a abordagem proposta nesta dissertação consiste em caracterizar a carga de trabalho de E/S em supercomputadores. Para tanto, foram combinados *profiles* de E/S de diferentes aplicações em um único conjunto de dados com o objetivo de identificar e caracterizar os principais comportamentos de E/S. Duas abordagens são aplicadas para caracterizar a carga de trabalho: (1) Análise da duração e da representatividade dos padrões de acesso a nível de aplicação; (2) Análise do comportamento das operações de E/S a nível de sistema.

Assim, analisou-se os dados fornecidos pela ferramenta Darshan coletados pela *Argonne Leadership Computing Facility* e disponibilizados ao público. Os dados são uma coleção de arquivos de registros anônimos referentes a 12 meses do ano de 2012 que resumem as características de E/S de 91.603 *jobs* de computação científica de produção executadas no supercomputador *IBM Intrepid Blue Gene/P*.

Foi apresentado o conceito de fase de E/S que define um intervalo de tempo onde uma aplicação realizou operações de E/S. A partir disso foi identificado que as fases de E/S das aplicações executaram em mediana por 1,2 microssegundos, se considerado o tempo ocioso entre as fases como parte da fase, identificando assim a estabilidade de uma fase de E/S na aplicação, esse valor sobe para 1 segundo. Quando considerado o comportamento global do sistema uma fase de E/S contém mais que uma operação e a duração dela em mediana é de 4 microssegundos. Além disso, conseguimos indicificar que em 50% do tempo, o sistema de arquivo paralelo trabalhou simultaneamente com 4 padrões de acesso distintos.

Palavras-chave: Caracterização da Carga de Trabalho de E/S. Comportamento de E/S. E/S Paralela. Computação de Alto Desempenho.

I/O Behavior of Parallel Applications in High Performance Computing Systems

ABSTRACT

In High-Performance Computing (HPC), multiple applications compete for the available resources, mainly for the parallel file system. This concurrency often causes contention in Input/Output (I/O) operations, reducing application performance. Therefore, characterizing supercomputer I/O operations is crucial to understanding the I/O workload and thus identifying potential bottlenecks. One way to investigate application access patterns is through the use of application I/O *profiles*; however, in an HPC system, these generate large amounts of data.

In this context, the approach proposed in this dissertation is to characterize the I/O workload in supercomputers. Therefore, we combined *profiles* of I/O from different applications into a single dataset in order to identify and characterize the main I/O behaviors. Two approaches are applied to characterize the workload: (1) Analysis of the duration and representativeness of access patterns at the application level; (2) Analysis of the behavior of I/O operations at the system level.

Thus, we analyzed the data provided by the Darshan tool collected by the *Argonne Leadership Computing Facility*. The data is a collection of anonymous log files for 12 months of the year 2012 that summarize the I/O characteristics of 91,603 production scientific computing *jobs* executed on the *IBM Intrepid Blue Gene/P supercomputer* .

The concept of the I/O phase was presented, which defines a time interval where an application performed I/O operations. We identified that the I/O phases of the applications ran on average for 1.2 microseconds. If we consider the idle time between phases part of the phase, thus identifying the stability of an I/O phase in the application rises to 1 second. When we consider the system's overall behavior, an individual I/O phase contains more than one operation, and its median duration is 4 microseconds. Furthermore, we were able to indicate that 50% of the time, the parallel file system worked simultaneously with four different access patterns.

Keywords: I/O workload characterization, I/O Behavior, Parallel I/O, High Performance Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na Computação de Alto Desempenho (CAD), especialmente em supercomputadores, existe um grande número de aplicações que são executadas de forma concorrente (YANG; GUO, 2005). Tais aplicações competem pelos recursos disponíveis, principalmente pelo sistema de arquivos paralelo. Esta concorrência muitas vezes causa contenção nas operações de entrada e saída (E/S) reduzindo o desempenho das aplicações paralelas (JI et al., 2019; YU et al., 2017). Isto está diretamente relacionado com o número de requisições de E/S e com o padrão de acesso de E/S (HERBEIN et al., 2016).

As operações de E/S podem ser caracterizadas pelos seus padrões de acesso, que representam a maneira como uma aplicação acessa os seus dados. Algumas características são consideradas na literatura para definir o padrão de acesso. Dentre elas destacam-se o tamanho das requisições, o tipo de operação (escrita ou leitura) e a espacialidade das solicitações (acesso contíguo ou *1D-strided*) (BERGMAN et al., 2008; BOITO et al., 2018). Nesse contexto, identificar a carga de trabalho através do padrão de acesso pode auxiliar na caracterização de operações de E/S um sistema de CAD além de auxiliar desenvolvedores a melhorar o desempenho das aplicações atuais e futuras (LI et al., 2018; CARON et al., 2018; RADFORD; METZ; CHINTALA, 2015; NATALE; MARTINELLI, 2019).

Uma das formas de investigar o padrão de acesso das aplicações é através do uso de *profiles* de aplicações visando a E/S, gerados por ferramentas especializadas. Tais ferramentas interceptam as requisições de E/S coletando informações das operações realizadas. No entanto a grande maioria das ferramentas produz um rastreamento de granulação grossa com informações agregadas sobre as requisições de E/S. Este comportamento é justificado, uma vez que o uso de um rastreamento refinado geralmente impõe uma sobrecarga ao sistema. Mesmo assim, a adoção de granulação grossa com agregação de informações em sistemas CAD gera grandes quantidades de dados de *profiles* de E/S, devido ao alto volume de *jobs* submetidos a um supercomputador.

Outra questão é que a maioria das ferramentas de análise de perfil de E/S produz dados de perfil separados para cada aplicação em execução. Esta metodologia ajuda a identificar o comportamento de E/S em aplicações individualmente. Como os dados são gerados separados, torna-se difícil analisar a concorrência imposta entre as aplicações e mais difícil identificar o comportamento do sistema como um todo. Assim, é necessária a utilização de técnicas para combinar as informações de perfis individuais para a análise

de todo o sistema.

Nesse contexto, nosso trabalho descreve uma abordagem para a caracterização da carga de trabalho de E/S em supercomputadores a partir do comportamento observado nas aplicações. Para tanto, foram combinados perfis de E/S de diferentes aplicações em um único conjunto de dados com o objetivo de identificar e caracterizar os principais comportamentos de E/S em uma máquina ao longo do tempo. Buscando demonstrar a aplicação da nossa abordagem, a empregamos em rastros de E/S das aplicações do supercomputador Intrepid Blue Gene/P de Argonne.

1.1 Contribuições da Pesquisa

No contexto do trabalho realizado, nossas principais contribuições são:

- Combinação de perfis de E/S de diferentes aplicações em um único conjunto de dados;
- Abordagem estatística das operações de E/S de um sistema de arquivos paralelos; e
- Análise a nível de aplicação do comportamento dos padrões de acesso.

As informações obtidas com a caracterização realizada podem orientar pesquisas futuras de otimizações de E/S. Além disso, pode fornecer um panorama de um cenário da real do sistema de arquivos, permitindo melhor avaliar técnicas de otimização de E/S utilizando padrões e comportamentos representativos das aplicações de CAD.

1.2 Organização do Documento

Este trabalho está organizado em cinco capítulos, sendo este o primeiro capítulo com a introdução do trabalho desenvolvido. A seguir, o Capítulo 2 apresenta o estado da arte sobre os tópicos desta dissertação e discute trabalhos relacionados operações de E/S. O Capítulo 3 abordará o desenvolvimento pesquisa, assim como quais foram as metodologias e funcionalidades utilizadas no trabalho. O Capítulo 4 apresenta os resultados alcançados do comportamento de E/S a nível das aplicações e a nível do sistema. E por fim o Capítulo 5 descreve conclusões com base em nossas descobertas e apresenta algumas ideias de trabalho futuro.

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

Este capítulo apresenta os principais conceitos de E/S paralela abordados neste trabalho. Serão discutidos os aspectos essenciais que servem de base para esta dissertação. Além disso, neste capítulo, serão detalhados os trabalhos relacionados visando apresentar os critérios que foram adotados por outros autores para proporem soluções na mesma área.

2.1 A Pilha de E/S Paralela

Para acessar os dados, aplicações utilizam-se de uma interface, na qual em alto nível pode se traduzir ao uso de uma biblioteca, exemplos de bibliotecas utilizadas em sistemas CAD que possibilitam o acesso de arquivos por aplicações são, POSIX (POLLAK, 1996) e MPI-IO (CORBETT et al., 1996).

Um acesso ao dado vindo de uma aplicações e caracterizado como uma operação de E/S, exemplos destas operações são *abrir*, *ler*, *escrever* e *fechar*. Estas operações produzem requisições para a pilha de E/S, que é responsável pelo tratamento de cada uma. Em sistemas de grande porte, a computação é dividida entre vários processos sendo estes distribuídos por um conjunto de nós de processamento. Assim, os dados também costumam ser divididos, buscando melhorar o desempenho. Desta forma, a adoção de um sistema de arquivos paralelos (SAP) (ROSS; THAKUR et al., 2000; MICROSYSTEMS, 2007; LI et al., 2018) é uma solução para este tipo cenário (LOCKWOOD et al., 2018).

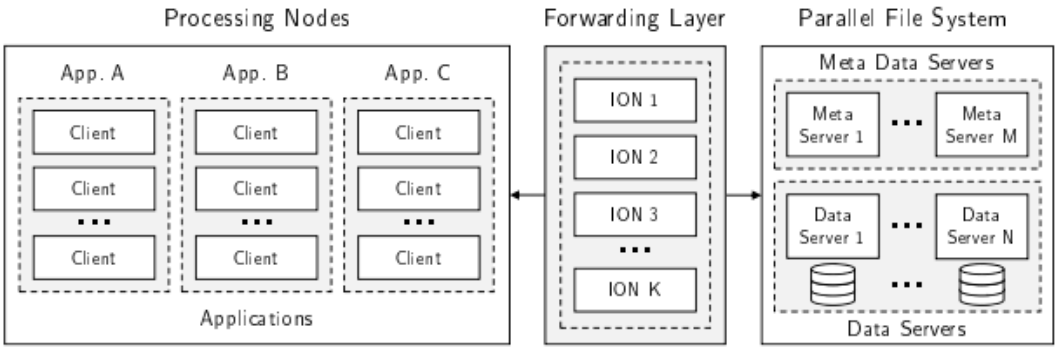
Os SAP produzem uma abstração do sistema de arquivos uma vez que os arquivos compartilhados e remotos podem ser acessados como se estivessem armazenados localmente. Geralmente no uso de SAP é empregado um conjunto de servidores para realizar o armazenamento de dados, permitindo assim que os dados possam ser acessados em paralelo, melhorando o desempenho das requisições de E/S. Na implementação de SAP são utilizados dois tipos de servidores especializados: os *servidores de dados* e os *servidores de metadados*. Em algumas implementações esses servidores podem até ser configurados nas mesmas máquinas físicas (ROSS; THAKUR et al., 2000).

Para possibilitar que as operações de E/S ocorram simultaneamente, o SAP aplica uma função chamada *striping* sobre os arquivos. O *striping* consiste em quebrar um arquivo em partes, de tamanho fixo, que são armazenados em diferentes servidores. Normalmente, uma abordagem *round-robin* é empregada para realizar a distribuição, porém

pode-se utilizar somente *subset* ou somente um servidor (DIBBLE; SCOTT, 1989).

Dada essa organização, as solicitações de E/S passam por diferentes camadas no SAP até o acesso dos dados. Na Figura 2.1 é ilustrada uma visão geral da pilha do SAP com os principais componentes. À esquerda da figura são representados os nós dedicados atuando como nós clientes. Eles representam os nós nos quais as aplicações executam seu processo de E/S. Estes nós clientes acessam o SAP através de bibliotecas de E/S, na qual cada acesso é uma requisição que o SAP precisa executar. Também são representados neste SAP a camada de *forwarding* e os servidores de metadados e de dados (BEZ et al., 2021).

Figura 2.1: Exemplo de pilha de um sistema arquivo paralelo.



Fonte: Boito et al. (2018)

2.1.1 Camada Física

A camada mais inferior é denominada de camada física, onde os dados são realmente armazenados. Existem diferentes tecnologias para o armazenamento dos dados, podendo ser através de fitas, discos HDDs (Hard-disk Drive) e SSDs (Solid State Drive). No entanto, cada tecnologia afeta o desempenho das requisições de diferentes maneiras, consequentemente, afetando o desempenho final das aplicações (PAVAN et al., 2019b).

Em um sistema CAD, alguns dados são arquivamento em fitas, buscando a durabilidade do armazenamento destes dados (PEASE et al., 2010). A tecnologia *Linear Tape-Open* (LTO) arquiva os dados em fitas magnéticas em formato linear; portanto, o acesso ocorre em um padrão sequencial. Como a durabilidade é maior do que outros dispositivos, e o preço por espaço de armazenamento é mais barato, esta tecnologia ainda é uma boa escolha quando arquivar dados em fitas é necessário. Hoje a última versão é a

LTO-8, que pode economizar 12 TB em um único dispositivo (STARR et al., 2020).

O dispositivo mais usado é o HDD, que salva os dados em discos com superfície magnética. Este dispositivo possui uma arquitetura de cilindro em que cada um é estruturado em faixas divididas em setores. Em geral, todos os setores têm o mesmo tamanho, que alterna de 512 a 4096 bytes. A construção das trilhas e setores é realizada através do processo denominado formatação física (KASAVAJHALA, 2011).

Para acessar um determinado dado em um disco rígido, primeiro precisa-se localizar a trilha e o setor onde os dados foram armazenados, movendo a cabeça para o local apropriado (CHEN et al., 2002). Esse processo resulta em um tempo de acesso que é influenciado por:

- **Tempo de busca:** tempo necessário para posicionar a cabeça de leitura/escrita no local apropriado;
- **Tempo de latência:** tempo necessário para chegar ao início do setor a ser lido ou escrito. Esse fator é definido pela velocidade do motor do dispositivo; e
- **Tempo de transferência:** tempo efetivo para ler ou escrever dados.

Os HDDs são conhecidos por apresentarem o melhor desempenho quando os acessos são feitos sequencialmente, em vez de aleatoriamente, porque o primeiro minimiza o tempo de busca. Almejando equilibrar o pico das taxas de requisições com largura de banda de E/S, a maioria dos sistemas paralelos suporta várias matrizes redundantes de discos independentes (RAID) (THOMASIAN et al., 2004). Portanto, vários RAIDs fornecem largura de banda de E/S de pico igual ao produto do número de RAIDs e suas larguras de banda individuais. Nos RAIDs, os dados são distribuídos entre os discos, em faixas de tamanho fixo, e podem ser recuperados em paralelo, o que melhora o desempenho; além disso, o desempenho é limitado pela combinação de tamanho da faixa e acesso (JR et al., 1995).

Soluções mais recentes têm adotado o uso de SSD em alternativa aos HDDs. Os SSD são dispositivos de armazenamento construídos com componentes flash e, com o surgimento das tecnologias flash 3D de célula de nível triplo (TLC) e célula de nível quádruplo (QLC), os SSDs modernos podem oferecer uma capacidade de armazenamento de até dezenas de terabytes (TB) (CORNWELL, 2012). Por outro lado, como os SSDs usam memórias flash do tipo NAND, estas não podem ser alteradas. Para realizar este tipo de procedimento uma solicitação de gravação é realizada em um novo lugar, chamado de página limpa para armazenar os dados atualizados. Esses espaços não podem ser

substituídos diretamente, portanto, para gravar dados em um local, eles devem ser lidos, modificados e armazenados em outro espaço limpo. O local antigo não é mais válido e será tratado pela coleta de lixo em segundo plano. Esse processo causa à amplificação de gravação no SSD, o que reduz o desempenho de gravação e a resistência do SSD (YANG; PEI; YANG, 2019; PARK et al., 2021).

Cada dispositivo possui características físicas diferentes que podem produzir diferentes desempenhos de E/S. Uma das características que podem melhorar o desempenho em HDDs e SSDs é o uso de *cache* no hardware. Esses *caches* geralmente executam técnicas de pré-busca e leitura antecipada, que tentam prever dados que serão acessados no futuro e fazer essas solicitações mais cedo. Portanto, os acessos de leitura aleatória podem ter um desempenho pior que os sequenciais porque não exploram completamente esses mecanismos (BOITO et al., 2018).

2.1.2 Interfaces

As aplicações fazem suas requisições de E/S para os servidores do sistema de arquivos paralelos de diferentes maneiras, dependendo de como foram codificadas. Várias características, como o número de requisições, o tamanho destas e sua localização espacial no arquivo determinam o que chamamos de padrão de acesso da aplicação. Esse padrão tem um impacto direto no desempenho, portanto, muitas pesquisas buscam otimizar este acesso a dados modificando o padrão de acesso (LOFSTEAD et al., 2011; HE et al., 2013; YIN et al., 2013; KUO et al., 2014).

O padrão de acesso de E/S das aplicações pode ser classificado em global ou local. Segundo Yin et al. (2013) o padrão global descreve o comportamento de toda aplicação, enquanto o padrão local o faz no contexto de um processo ou tarefa. As informações do padrão de acesso local geralmente são empregadas para identificar e aplicar otimizações no lado do cliente, enquanto o padrão de acesso global é mais relevante para otimizações na camada de encaminhamento ou dos servidores do sistema de arquivos, pois possui uma visão geral dos acessos de dados.

Neste estudo, foram considerados os seguintes aspectos para descrever o padrão de acesso a dados do aplicação: o número de arquivos, a localização espacial dentro do arquivo, o tamanho dos acessos e a operação de E/S.

Em relação ao número de arquivos, foram considerados dois cenários que retratam como a maioria das aplicações de CAD fazem E/S. No primeiro, cada processo da aplica-

ção lê ou grava dados em seu arquivo individual. No segundo cenário, todos os processos compartilham um arquivo comum. Além disso, o parâmetro de localização espacial descreve se o acesso a um arquivo compartilhado é sequencial, ou seja, cada processo acessa partes contíguas do arquivo ou *1D-strided*, isto é, cada processo acessa partes do arquivo com uma lacuna de tamanho fixo entre as requisições de um mesmo processo (BOITO et al., 2019; SUGIHARA; TATEBE, 2020).

O tamanho da solicitação também tem grande impacto no desempenho de E/S devido à sensibilidade dos dispositivos de armazenamento para acessar tamanhos diferentes de requisições e dos custos de rede destas requisições (BOITO et al., 2015; KANG et al., 2020). Pequenas solicitações, por exemplo, sofrem mais devido à sobrecarga imposta pela latência da rede, que domina o custo de processamento da solicitação.

2.1.3 Ferramenta de E/S para Coleta de Dados Estatísticos

O Darshan (CARNS et al., 2009) foi projetado para capturar uma imagem precisa do comportamento de E/S do aplicação, incluindo propriedades como padrões de acesso nos arquivos, com sobrecarga mínima. Ele pode ser usado para investigar o comportamento de E/S de aplicações CAD complexas. Além disso, o projeto leve (CARNS et al., 2012) do Darshan o torna adequado para implantação em tempo integral para caracterização da carga de trabalho de grandes sistemas.

O Darshan foi desenvolvido originalmente na série de computadores *IBM Blue Gene/P* implantados no Argonne Leadership Computing Facility, mas é portátil em uma ampla variedade de plataformas, incluindo os clusters Cray XE6 e Cray XC30 (CARNS et al., 2011).

O Darshan é implementado como um conjunto de bibliotecas de espaço do usuário. Essas bibliotecas não requerem modificação do código-fonte e podem ser adicionadas de forma transparente durante a fase de link dos *scripts* do compilador MPI. Essa abordagem compromete o objetivo de transparência, pois os binários existentes devem ser recompilados (ou vinculados novamente) para usar o Darshan. No entanto, ele pode ser aplicado automaticamente a aplicações recém-compiladas ou introduzido como parte de uma atualização do PMPI. Em troca desse compromisso, a Darshan pode utilizar mecanismos portáteis de baixo custo para interceptar rotinas de E/S.

O Darshan captura rotinas MPI-IO (TSUJITA et al., 2018) usando a interface de criação de perfil (PMPI) para MPI (KARRELS; LUSK, 1994). As rotinas POSIX são

capturadas inserindo funções de *wrapper* através dos *linkers* GNU (CHAMBERLAIN; TAYLOR, 2010). Esses mecanismos foram testados com a implementação MPICH MPI para os compiladores GNU e IBM C, C ++ e Fortran. Ele também funciona corretamente para compilação estática e dinâmica, não requer infraestrutura de suporte adicional para instrumentação e é compatível com outras implementações e compiladores MPI.

Em vez de capturar um rastreamento completo de todas as operações de E/S, Darshan caracteriza a aplicação usando estatísticas e informações de tempo cumulativas. A vantagem dessa abordagem é que os dados podem ser armazenados compactamente usando uma quantidade limitada de memória. Os dados são gravados independentemente em cada processo em tempo de execução e, em seguida, agregados e armazenados quando o *job* é encerrado. Darshan não invoca rotinas de comunicação ou armazenamento até o final da execução. Portanto, reduz o escopo do desafio de escalabilidade para uma única rotina de desligamento.

2.2 Trabalhos relacionados

No contexto do CAD, a velocidade de acesso ao disco e a contenção desses, podem impactar negativamente no desempenho de E/S. Estes fatores foram avaliados em diversos estudos como em (ROSS et al., 2005; NISAR; LIAO; CHOUDHARY, 2008; ALFOROV et al., 2018; CARNEIRO et al., 2018; PAVAN et al., 2019b; PAVAN et al., 2019a). Além disso, o entendimento e a caracterização de uma plataforma podem fornecer intuições extras sobre como as aplicações devem realizar operações de E/S para obter o melhor desempenho e, assim, mitigar esses fatores negativos.

Zoll et al. (ZOLL; ZHU; FENG, 2010) estudaram um conjunto de rastros de E/S de aplicações do supercomputador da Advanced School for Computing and Imaging (ASCI). Os rastros foram obtidos utilizando a ferramenta *strace*, sendo que eles variavam de dezenas de segundos a meia hora, e incluíam duas aplicações científicas e três testes gerados com o *benchmark* IOR (Interleaved or Random). Eles concluíram que um modelo de Markov não poderia representar a taxa de chegada de pedidos dos fluxos de E/S das aplicações, pois eles apresentam autossimilaridade. Eles propuseram um modelo estocástico para prever essa métrica. Wang et al. (WANG et al., 2004) estudaram o comportamento do tamanho das requisições a partir dos mesmos rastros e mostraram que a maioria das aplicações realizou um grande número de pequenas requisições (de alguns *bytes* a 1 MB) em pequenos intervalos de tempo.

Kim et al. (KIM et al., 2010; KIM; GUNASEKARAN, 2015) caracterizaram a carga de trabalho do supercomputador da Oak Ridge Leadership Computing Facility (OLCF). Eles consideraram a utilização do sistema, as demandas das operações de leitura e escrita, o tempo ocioso e a distribuição dos pedidos de leitura para escrever os pedidos. O estudo demonstrou que a utilização da largura de banda e o tempo entre as chegadas dos pedidos poderia ser modelado como uma distribuição de Pareto.

Para motivar seu trabalho na coordenação de aplicações cruzadas, Dorier et al. (DORIER et al., 2014) utilizaram dados do *Parallel Workload Archive*, do período entre janeiro e setembro de 2009. Através de um modelo simples e otimista, eles utilizaram a distribuição de alguns *jobs* simultâneos para mostrar que havia uma alta probabilidade de ter múltiplas aplicações realizando simultaneamente operações de E/S, mesmo quando as aplicações gastavam tão pouco quanto 5% do seu tempo de execução em E/S.

Luu et al. (LUU et al., 2015) analisaram os *logs* do Darshan de um milhão de tarefas executadas durante 2013 em dois supercomputadores. Eles demonstraram que quase um terço dos *jobs* teve uma produção agregada de não mais de 256 MB/s. Além disso, apesar da existência de bibliotecas paralelas de alto nível, três quartos dos trabalhos utilizaram apenas o POSIX para realizar E/S.

Liu et al. (LIU et al., 2016) propuseram o AID (Automatic I/O Diverter), um sistema que realiza a caracterização automática de E/S da aplicação e o escalonamento levando E/S em consideração. O AID analisa o tráfego de E/S existente e o histórico de trabalhos em lote, sem qualquer conhecimento prévio sobre aplicações ou envolvimento do usuário/desenvolvedor. Eles avaliaram o AID no supercomputador Titan, usando aplicações reais, e confirmaram que o AID é capaz de (1) identificar aplicações E/S-intensivas e suas características detalhadas de E/S, e (2) reduzir significativamente a degradação/variação de desempenho de E/S destas aplicações, avaliando conjuntamente o padrão de E/S das aplicações em espera e a carga de E/S do sistema em tempo real.

Xie et al. (XIE et al., 2012) apresentaram uma caracterização do desempenho de armazenamento do supercomputador Cray XK6 Jaguar enquanto examinavam as implicações desses resultados para o desempenho da aplicação. Eles observaram e quantificaram limitações de tráfegos dos dados devido a concorrência, interferência e gargalos de operações de escrita em arquivos compartilhados.

2.3 Conclusão do Capítulo

Observa-se nos trabalhos relacionados que as propostas desenvolvidas adotaram uma análise de dados para entender o comportamento de aplicações quando realizam suas operações de E/S, estes dados foram providos, gerados ou coletados em sistemas de alto desempenho. Nesse contexto, nosso trabalho propõe caracterizar o comportamento de E/S, para isso desenvolvemos uma abordagem para combinar perfis das operações de E/S e aplicamos uma análise estatística sobre as operações.

Diferentemente de Zoll et al. (ZOLL; ZHU; FENG, 2010), Wang et al. (WANG et al., 2004), Dorier et al. (DORIER et al., 2014) e Xie et al. (XIE et al., 2012), que utilizaram dados providos de *benchmarks*, usamos como base para nossa análise os rastros obtidos de forma transparente pela ferramenta Darshan, sobre aplicações reais. Nos trabalhos de Kim et al. (KIM et al., 2010; KIM; GUNASEKARAN, 2015), os dados foram coletados com uma acurácia de 2 segundos durante 6 meses no ano de 2010, diferentemente, este trabalho apresenta uma acurácia de microssegundos.

Ao contrário de Luu et al. (LUU et al., 2015), que analisou os contadores gerados pelo Darshan diretamente, este trabalho, utiliza-se deles para estimar fases de E/S e assim reconstruir o *workload* de E/S. O trabalho de Liu et al. (LIU et al., 2016) coletou os dados de E/S e utilizou para medir o *throughput* das operações, enquanto este trabalho foca na duração das operações.

Neste capítulo observou-se a relevância do tema de pesquisa, endereçando os principais trabalhos relacionados que tentam mitigar a questão da caracterização das operações de E/S. No próximo capítulo, apresentamos nossa abordagem em detalhes.

3 PROPOSTA: UMA ABORDAGEM PARA ESTIMAR PADRÕES DE E/S

As operações de E/S são gargalos de desempenho para muitas aplicações em CAD devido a contenção de E/S que ocorre nos acessos simultâneos, e à diferença de velocidade entre o processamento e o acesso aos dados. A partir disso, o objetivo deste trabalho é caracterizar as operações de E/S, procurando compreender o comportamento de E/S. Neste capítulo será apresentado a abordagem de pré-processamento utilizada no conjunto de dados para caracterizar as operações de E/S das aplicações a partir da estimativa das fases de E/S. Esta abordagem é baseada em um *workflow* composto por três etapas: *parser* dos dados; extração dos contadores; e extração das fases de E/S.

Na Seção 3.1 são apresentadas informações do sistema utilizado para a caracterização do comportamento E/S. Na sequência são descritas as três etapas do *workflow* proposto. Na primeira etapa, apresentada na Seção 3.2, foram extraídas informações dos arquivos binários de rastros do Darshan para todas as aplicações. Em seguida, na segunda etapa, apresentada na Seção 3.3, buscou-se extrair contadores relevantes para análise e organizá-los em um formato eficiente para manipulação dos dados, JSON (Java Script Object Notation). Finalmente, na terceira etapa, apresentada na Seção 3.4, foi estimado as fases de E/S de cada aplicação, construindo um CSV para cada dia. A Figura 3.1 mostra o *workflow* com os passos e a descrição de cada etapa.

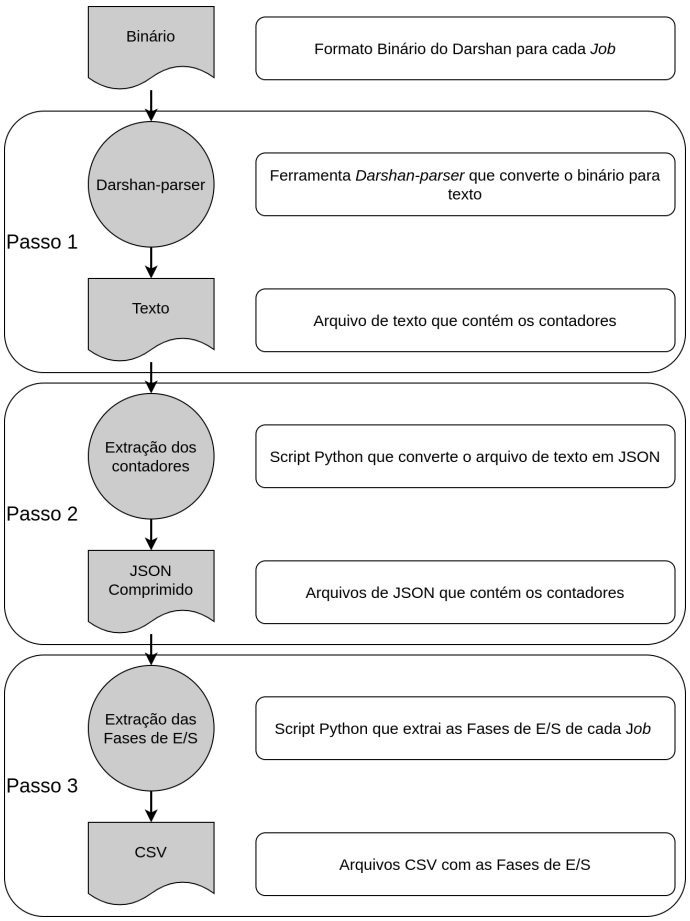
Com o objetivo de reduzir o tempo desta etapa de pré-processamento, foram implementadas versões paralelas dos *scripts* de parseamento. Nos *scripts* em Python, utilizamos uma abordagem orientada a tarefas. Onde foi utilizado um pool de tarefas que podiam ser executadas pelos processos. Cada tarefa consiste em um par formado por um *script* e um arquivo. Esses *scripts* e arquivos foram mudando durante as etapas do *workflow*.

3.1 Base de Dados

Para caracterizar o comportamento E/S, foram analisados os dados fornecidos pela ferramenta Darshan recolhidos pela Argonne Leadership Computing Facility e disponibilizados ao público online [1]. Os dados são uma coleção de arquivos de registo anônimo que agrupam as características de E/S das aplicações de computação científica de produção no Laboratório Nacional em Argonne no supercomputador IBM Intrepid Blue Gene/P.

[1] <http://www.mcs.anl.gov/research/projects/darshan/docs/tm-13.1_darshan-data-guide.pdf>

Figura 3.1: *Workflow* de pré-processamento executado antes da análise dos dados.



Fonte: Autor (2021).

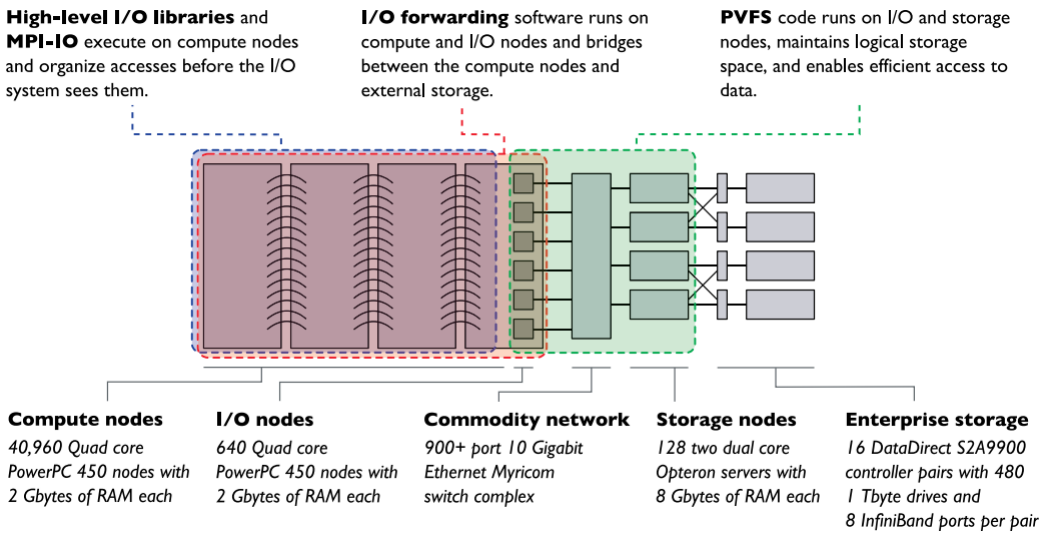
Este supercomputador ocupou o 38º lugar na lista Top500 no ano 2012[2]. Possuía 163.840 núcleos com 458,6 TFlops. O sistema de armazenamento tinha uma capacidade total de 5,2 PiB e uma taxa de pico de E/S de aproximadamente 78 GiB/s. A Figura 3.2 mostra a arquitetura da Intrepid, incluindo o hardware de armazenamento ligado externamente à máquina.

Os nós de computação e os nós de E/S são os mesmos PowerPC. Para cada conjunto de 64 nós de computação, existe um único nó de E/S que executa o encaminhamento de E/S para os nós de armazenamento. Existem 128 desses nós que estão ligados ao armazenamento pelo InfiniBand. Este processo executa RAID 6 que fornece tolerância a falhas e permite que o ambiente de software forneça garantias de alta disponibilidade para o armazenamento (LANG et al., 2009).

Os dados foram coletados de diferentes *jobs* utilizando três versões de Darshan nos anos 2010, 2012 e 2013. Os registros Darshan não cobriam 100% de todas as aplicações que executavam no Intrepid. A taxa de cobertura de Darshan variou de 20% a 80% de

[2]<https://www.top500.org/system/176322/>

Figura 3.2: Configuração do sistema de E/S do IBM Blue Gene/P.



Fonte: Lang et al. (2009)

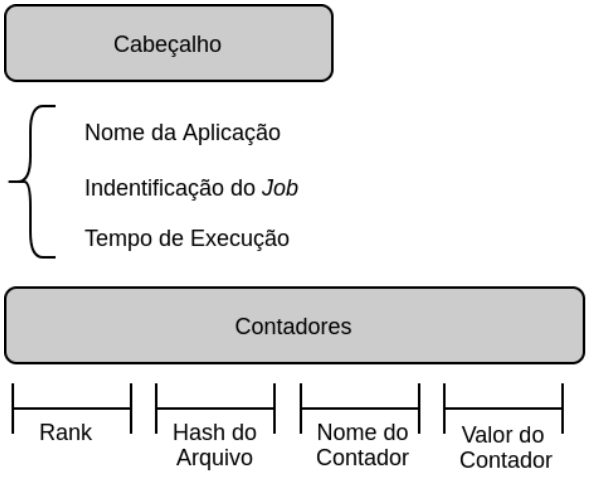
semana para semana. Para este trabalho foram utilizadas as observações de 2012, que foram geradas pelas versões 1.23, 1.24 e 2.0, totalizando 36, 359 e 91.603 observações com cada uma, respectivamente. Foram selecionados os dados obtidos com a versão 2.0 do Darshan, já que eles representam mais de 99,57% dos *jobs* reportados. O repositório contém informação sobre os *jobs* executados em 346 dias, a partir de 3 de janeiro até 31 de dezembro de 2012. Entre 28 de Novembro e 17 de dezembro não foram reportados dados de E/S.

3.2 Etapa 1 - Conversão de dados

Como o Darshan salva os dados coletados das aplicações em um formato binário, para extrair e transformar os dados necessários para análise, foi utilizado a ferramenta *darshan-parser*, disponibilizada junto com o Darshan. Essa ferramenta transforma o arquivo binário em um arquivo de texto, que contém todos os contadores relacionados às operações de E/S coletadas pelo *profiler*. O arquivo em formato de texto gerado pelo *darshan-parser* é organizado em duas seções, como mostra a Figura 3.3. No início do arquivo é armazenado em pares de chave-valor dados que descrevem a execução como: nome da aplicação, identificador da tarefa e o tempo de execução são alguns exemplos. O restante do arquivo contém as medidas de cada contador capturado por Darshan, em relação a cada arquivo aberto pela aplicação. Cada linha apresenta um formato tabular. Exemplos de contadores são: número de operações (leitura, gravação, *seek*, *stat*), usando

várias interfaces (POSIX, MPI-IO, STDIO), histogramas de tamanhos de acesso e tempo acumulado gasto em cada operação.

Figura 3.3: Organização do arquivo de texto gerado pelo *darshan-parser*.



Fonte: Autor (2021).

3.3 Etapa 2 - Extração de contadores relevantes dos traços do Darshan

A quantidade de dados gerados pelo Darshan é de aproximadamente 100GB. Assim, uma nova etapa foi utilizada para extrair os contadores relevantes para posterior análise e armazenamento em um formato JSON compactado. Este script, implementado em linguagem Python, percorre o arquivo texto gerado na etapa 1 pelo *darshan-parser* procurando os contadores e os armazenado em uma estrutura de dicionário. O formato JSON foi selecionado pela facilidade em converter a estrutura de dicionário do Python para ele, além disso, é um formato leve de compartilhamento de dados, que usa uma coleção de pares *nome/valor*. Isso ajuda na busca de um valor específico em consultas futuras. Os arquivos foram compactados com o objetivo de reduzir o espaço de armazenamento.

Existem uma gama de contadores que o Darshan coletou. Estes contadores foram divididos em seis categorias. A primeira refere-se aos *timestamp* de data e hora das operações, a segunda é o tempo acumulado, a terceira categoria refere-se aos bytes transferidos, a quarta é chamada de desempenho e relata algumas estatísticas de tempo gasto em E/S. A quinta categoria relata informações sobre o número de operações. Por último, a sexta categoria apresenta dados sobre os tamanhos de requisições usados pelas aplicações. Na Tabela 3.1 são apresentados todos os contadores.

Tabela 3.1: Relação de contadores coletados pela ferramenta Darshan.

Tipo	Nome do Contador	Descrição
Timestamp	CP_F_READ_START_TIMESTAMP	Timestamp de início da primeira operação de leitura.
	CP_F_READ_END_TIMESTAMP	Timestamp do final da última operação de leitura.
	CP_F_WRITE_START_TIMESTAMP	Timestamp de início se a primeira operação foi de escrita.
	CP_F_WRITE_END_TIMESTAMP	Timestamp de data/hora em que a última operação de escrita terminou.
	CP_F_OPEN_TIMESTAMP	Timestamp da primeira vez que o arquivo foi aberto.
	CP_F_CLOSE_TIMESTAMP	Timestamp da última vez em que o arquivo foi fechado.
Tempo	CP_F_POSIX_READ_TIME	Tempo cumulativo gasto lendo usando a interface POSIX.
	CP_F_POSIX_WRITE_TIME	Tempo cumulativo gasto em escrita, fsync e fdatasync usando a interface POSIX.
	CP_F_POSIX_META_TIME	Tempo cumulativo gasto em abrir, fechar, stat e seek usando a interface POSIX.
	CP_F_MPI_READ_TIME	Tempo cumulativo gasto lendo usando a interface MPI-IO.
	CP_F_MPI_WRITE_TIME	Tempo cumulativo gasto com gravação e sincronização usando a interface MPI-IO.
	CP_F_MPI_META_TIME	Tempo cumulativo gasto em abrir e fechar usando a interface MPI-IO.
Bytes	CP_BYTES_READ	Número total de bytes lidos do arquivo.
	CP_BYTES_WRITTEN	Número total de bytes gravados no arquivo.
Desempenho	CP_FASTEST_RANK	O rank MPI com o menor tempo gasto em E / S.
	CP_FASTEST_RANK_BYTES	O número de bytes transferidos pelo rank com o menor tempo gasto em E/S.
	CP_SLOWEST_RANK	O rank MPI com o tempo mais longo gasto em E/S.
	CP_SLOWEST_RANK_BYTES	O número de bytes transferidos pelo rank com o maior tempo gasto em E/S.
	CP_F_FASTEST_RANK_TIME	O tempo gasto pelo rank que passou menos tempo em E/S.
	CP_F_SLOWEST_RANK_TIME	O tempo gasto pelo rank que passou mais tempo em E/S.
Operações	CP_POSIX_OPENS	Número de vezes que o arquivo foi aberto.
	CP_POSIX_READS	Número de operações de leitura em POSIX.
	CP_POSIX_WRITES	Número de operações de escrita em POSIX.
	CP_POSIX_SEEKS	Número de operações de seek em POSIX.
	CP_POSIX_FOPENS	Número de operações de abertura.
	CP_POSIX_FREADS	Número de operações de leitura.
	CP_POSIX_FWRITES	Número de operações de escrita.
	CP_POSIX_FSEEKS	Número de operações de seek.
	CP_INDEP_OPENS	Número de aberturas independentes MPI.
	CP_INDEP_READS	Número de leituras independentes MPI.
	CP_INDEP_WRITES	Número de escritas independentes MPI.
	CP_COLL_OPENS	Número de aberturas coletivas MPI.
	CP_COLL_READS	Número de leituras coletivas MPI.
	CP_COLL_WRITES	Número de escritas coletivas MPI.
	CP_SPLIT_READS	Número de leituras coletivas (split) MPI.
	CP_SPLIT_WRITES	Número de escritas coletivas (split) MPI.
	CP_NB_READS	Número de leituras não-bloqueantes MPI.
	CP_NB_WRITES	Número de escritas não-bloqueantes MPI.
	CP_SYNCS	Número de sincronizações de arquivo MPI.
	CP_CONSEC_READS	Número de leituras consecutivas POSIX.
	CP_CONSEC_WRITES	Número de escritas consecutivas POSIX.
	CP_SEQ_READS	Número de leituras sequenciais POSIX.
	CP_SEQ_WRITES	Número de escritas sequenciais POSIX.
Tamanho / Acesso	CP_SIZE_READ_*	Histograma de tamanhos de acesso de leitura usando a interface POSIX.
	CP_SIZE_READ_AGG_*	Histograma do tamanho total dos acessos de leitura usando a interface MPI.
	CP_SIZE_WRITE_*	Histograma de tamanhos de acesso de escrita usando a interface POSIX.
	CP_SIZE_WRITE_AGG_*	Histograma do tamanho total de acessos de escrita usando a interface MPI.
	CP_ACCESS[1-4]_ACCESS	4 Tamanhos de acesso mais comum.
	CP_ACCESS[1-4]_COUNT	Contagem dos 4 tamanhos de acesso mais comum.

Fonte: O Autor (2021).

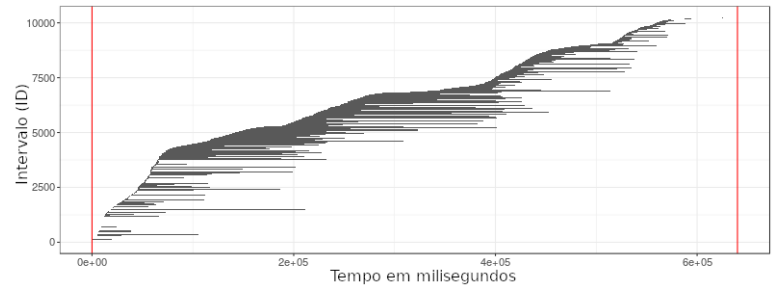
3.4 Etapa 3 - Estimando as fases de E/S a partir dos contadores extraídos

Um conceito importante na metodologia proposta é o de fase de E/S. Uma fase de E/S é composta por *timestamps* de início e fim e um padrão de acesso específico. Esses *timestamps* são coletados em microssegundos, essa é a menor medida de tempo disponível. Um padrão de acesso representa a maneira como uma aplicação acessa seus dados.

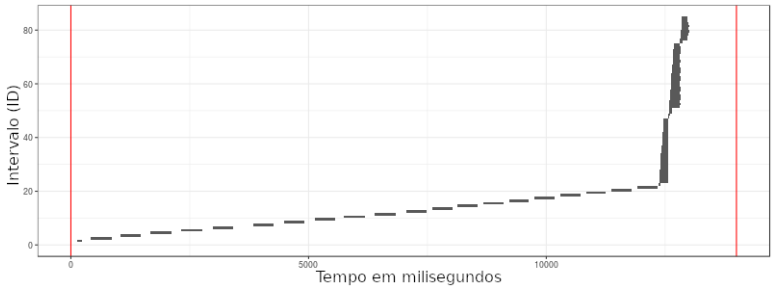
Como estamos usando os *timestamps* relatados por Darshan, que expressam a tempo da primeira e da última operação de um determinado padrão, não é garantido que durante todo esse período, as operações de E/S estejam realmente acontecendo. No entanto, é possível ter certeza de que, se alguma operação de E/S estiver acontecendo, esse padrão as caracterizará.

Como o Darshan relata informações agregadas sobre os *jobs*, como por exemplo os tamanhos médios de requisições e a quantidade total de dados acessados, não é fácil dividir a execução em fases temporais de E/S descritas por seus padrões de acesso. É porque o comportamento temporal é parcialmente perdido ao agregar estatísticas sobre a execução. No entanto, a geração de rastros mais refinados não é considerada viável na prática, devido à maior sobrecarga imposta às aplicações e a grande quantidade de dados gerados pelos rastros. O Darshan 3.0 apresenta uma funcionalidade chamada DXT, que realiza um rastreamento refinado, os dados gerados por essa funcionalidade poderiam ser utilizados em nossa abordagem, agregando o custo elevado de coletar esses dados (XU et al., 2017).

Figura 3.4: As fases de E/S de dois *jobs* diferentes executados no Intrepid em 2012.



(a) Job **A**.



(b) Job **B**

Fonte: Autor (2021)

Assim, usando as informações extraídas nas etapas anteriores e armazenadas no formato JSON compactado, é possível coletar cada padrão de acesso detectado para cada *job*, junto com o *timestamp* de início e fim. Essa informação foi coletada e salva em um arquivo no formato CSV.

A Figura 3.4(a) mostra as fases de E/S no *job* A executado no Intrepid. Cada ID é um intervalo que representa uma fase que contém operações de E/S neste *job* foram cerca de 10.000 fases. O eixo x apresenta o tempo de execução em milissegundos, a primeira linha vermelha representa o início e a segunda representa o final da execução. Além disso, a Figura 3.4(b) mostra as fases de E/S de outro *job* B. Este, apresentou menos fases que o anterior. Outra diferença entre os dois *jobs* é a sobreposição de fases. No segundo, esse comportamento aconteceu apenas perto do final da execução, enquanto no primeiro *job* foi comum durante toda a execução.

3.5 Conclusão do Capítulo

Neste capítulo foi apresentado a metodologia proposta para estimar as fases de E/S dos arquivos Darshan. Além disso, as três etapas da abordagem proposta e o supercomputador Intrepid foram apresentados. No próximo capítulo, a abordagem proposta é utilizada para caracterizar o comportamento de E/S do supercomputador utilizando os dados relativos a um ano inteiro.

4 RESULTADOS DA ANÁLISE DO COMPORTAMENTO DE E/S

Como pode ser visto no capítulo anterior, existem 91.603 *jobs* em 2012 que foram coletados pelo Darshan no supercomputador Intrepid. Em cada *job*, foi produzido um arquivo CSV, que contém as fases de E/S. Neste capítulo, nosso objetivo é discutir o comportamento de E/S com base nas fases de E/S. Este capítulo está organizado da seguinte forma. A Seção 4.1 apresenta uma análise do comportamento de E/S no nível das aplicações. Esta análise não considera a sobreposição de *jobs* que são executados em paralelo no sistema. Na Seção 4.2 é apresentado uma análise dos resultados no nível do sistema, considerando a sobreposição dos *jobs* para analisar o comportamento de E/S.

Cada fase de E/S contém um *timestamp* início e fim com um padrão de acesso específico. No conjunto de dados selecionados como caso de estudo - dados coletados do Intrepid supercomputador - existem 22 padrões de acesso únicos estimados a partir dos contadores disponibilizados pelo Darshan.

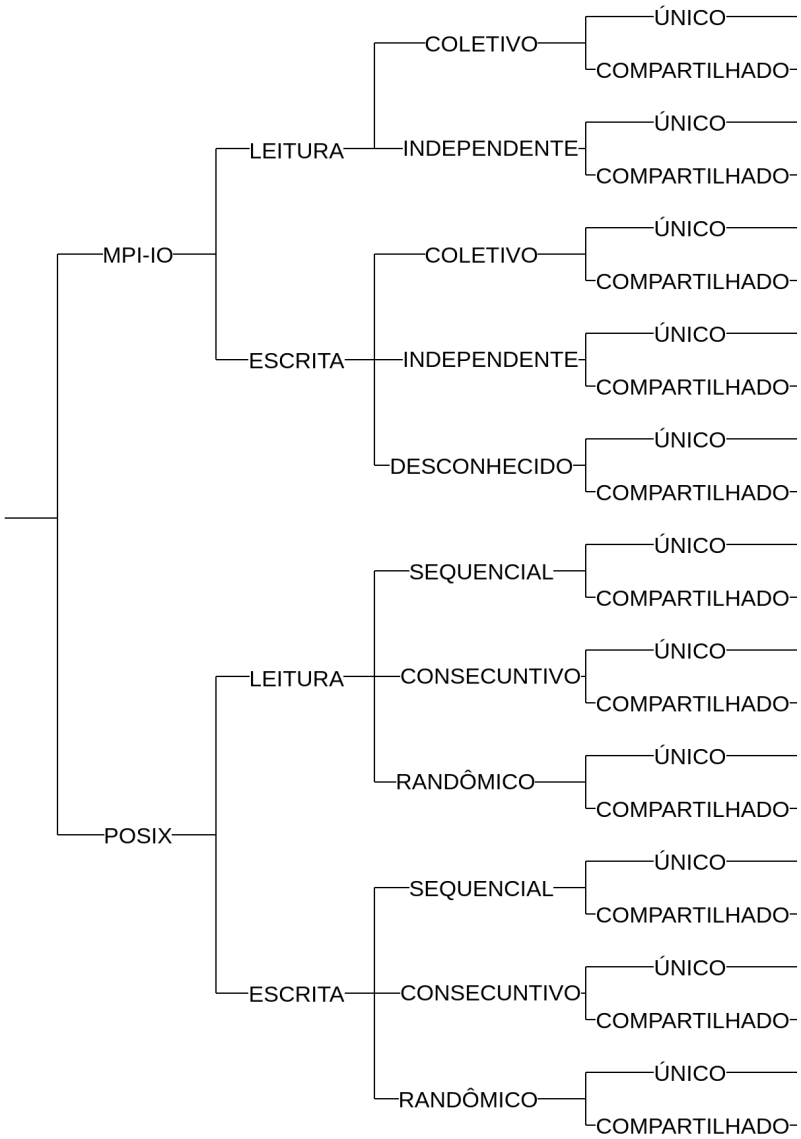
A Figura 4.1 mostra o diagrama em árvore desses padrões de acesso. As interfaces disponíveis são MPI-IO e POSIX. Para ambos, operações de leitura e escrita podem ser executadas. Para as operações de acesso MPI-IO, os processos podem acessar, de maneira coletiva ou independente, arquivos independentes (aqui chamados de "únicos") ou compartilhados. No entanto, para algumas fases de escrita, não foi possível estimar se os acessos eram coletivos ou independentes, devido à falta de dados nos traços do Darshan.

Para os acessos POSIX, em relação à espacialidade, os acessos podem ser consecutivos ou sequenciais ou randômico. A diferença entre acessos consecutivos e sequenciais é que o acesso consecutivo ocorre quando o acesso é imediatamente adjacente ao acesso anterior, enquanto o acesso sequencial ocorre em um deslocamento mais alto (*offset*) do que os acessos anteriores. Da mesma forma, como nas fases MPI-IO, os processos podem acessar arquivos únicos ou compartilhados.

4.1 Análise do comportamento de E/S no nível das Aplicações

Aqui, o objetivo foi analisar o comportamento geral de E/S das aplicações, e não no comportamento de processos individuais. Por esse motivo, antes de prosseguir, foi aplicado uma etapa extra de pré-processamento a esses dados. Isso foi necessário porque os traços do *Darshan* são contados para cada arquivo acessado, ou seja, uma fase em que vários processos acessam arquivos independentes (únicos) serão relatados várias vezes,

Figura 4.1: Diagrama em árvore dos 22 padrões de acessos.



Fonte: Autor (2021).

uma vez por processo envolvido. Isso significa que as fases que acessaram arquivos únicos seriam super-representadas no conjunto de dados. Para mitigar esse problema, nesta etapa adicional, todas as fases do mesmo *job* que correspondem ao mesmo padrão de acesso e que se sobrepõem no tempo foram agregadas. Portanto, obtemos o comportamento geral da aplicação, ignorando o comportamento dos processos individuais. Esta etapa foi implementada no Spark e executada no cluster LSD da Universidade de Bordeaux[1].

[1]https://lsd.labri.fr

4.1.1 Distribuição das Fases de E/S nas Aplicações

Considerando que cada fase de E/S apresenta um padrão de acesso distinto, nesta seção um padrão de acesso representa uma fase de E/S, e vice-versa. Com a intenção de observar quantos padrões as aplicações trabalham em sua execução, a Tabela 4.1 apresenta o número de padrões distintos que um *job* apresenta durante sua execução [2]. Em mediana uma aplicação apresentou 2 padrões de acesso, com 3 padrões no terceiro quadrante. O mínimo foi 1 e o máximo foi 14 padrões de acesso. Assim, a maioria dos *jobs* (75%) possui apenas 3 padrões de acesso ou menos; se considerarmos 99% dos *jobs*, eles apresentam 8 padrões ou menos.

Tabela 4.1: Distribuição dos padrões de acesso distintos presentes em uma aplicação.

Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
1	2	2	2,72	3	14

Fonte: Autor (2021).

Além disso, um *job* pode repetir o mesmo padrão várias vezes durante sua execução; dessa maneira, a Tabela 4.2 apresenta o número de padrões que se repetem durante a execução de uma aplicação. Como a mediana foi 1, 50% das vezes, um padrão que aconteceu em um *job* não se repete. 75% das vezes ele foi repetido apenas uma vez, já que o terceiro quadrante é 2. Em 99% da vezes, um padrão de acesso que aconteceu em um *job* foi repetido 251 vezes ou menos.

Tabela 4.2: Distribuição dos padrões de acesso repetidos presentes em uma aplicação.

Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
1	1	1	30,30	2	154613

Fonte: Autor (2021).

Analisou-se a repetição dos padrões de acesso, considerando as duas operações, escrita e leitura. Também se analisou as interfaces, desta forma, para os padrões caracterizados como POSIX e MPI-IO. Estes resultados são apresentados na Tabela 4.3. Para operações de escrita, o valor máximo foi maior que para as leituras. No entanto, as medianas e quadrantes são iguais entre escrita e leituras. Comparando 99% das vezes, as escritas são repetidas 971 vezes, enquanto as leituras são repetidas 207 vezes. Comportamento

[2]Todas as tabelas apresentadas nos resultados segue o formato desta tabela. Os dados são o aprestados de forma a analisar a distribuição das variáveis, divido a distribuição em formato de quartil.

semelhante ocorre com o MPI-IO e POSIX, com medianas e quadrantes semelhantes. A média é mais baixa para MPI-IO porque seu máximo é muito menor que o máximo para POSIX. O terceiro quadrante é um pouco mais alto para o MPI-IO, o que indica que essas fases tendem a se repetir mais vezes na mesma aplicação.

Tabela 4.3: Distribuição dos padrões de acesso repetidos presentes em uma aplicação separado pelas operações e interface.

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
Leitura	1	1	1	10,90	2	14865
Escrita	1	1	1	52,02	2	154613
MPI-IO	1	1	1	12,73	5	1017
POSIX	1	1	1	31,80	2	154613

Fonte: Autor (2021).

4.1.2 Distribuição da Duração das Fases de E/S

A carga de trabalho de E/S de um sistema pode ser caracterizada pela duração das requisições que ocorrem a ele. Essas requisições são caracterizadas por padrões de acesso, por exemplo, padrão de escrita ou de leitura. Nesse trabalho, as fases de E/S representam um padrão de acesso. Portanto analisou-se a duração destas para entender a carga de trabalho de E/S que ocorreu no supercomputador Intrepid.

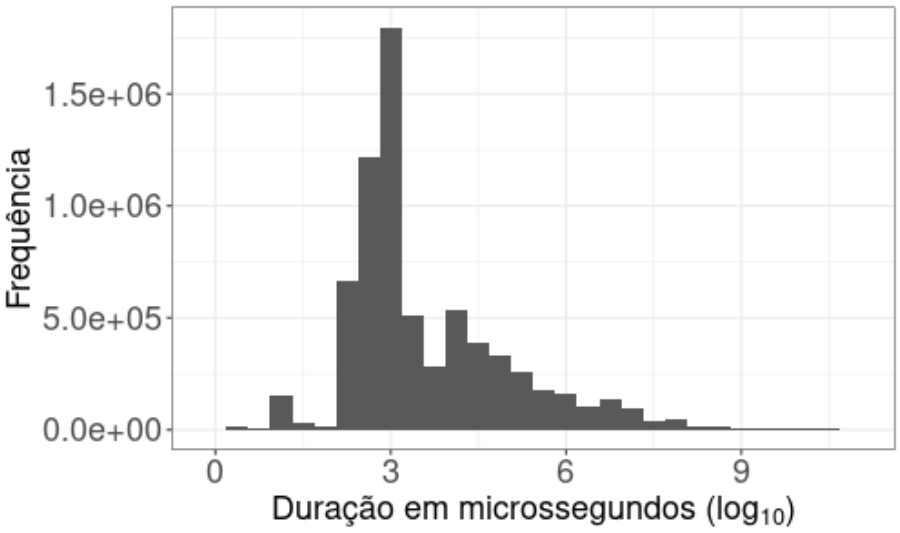
A primeira análise mostra a duração em geral, considerando todas as fases, sem distinguir os diferentes padrões que as compõe. Assim, a Tabela 4.4 apresenta a distribuição da duração das fases em microssegundos. Além disso, a Figura 4.2 mostra o histograma da duração. Todos os gráficos dos resultados são apresentados em forma de histograma, onde foram gerados com *bins* de tamanho 30, portanto os dados foram agregados em 30 grupos de intervalos idênticos. Além disso o eixo x que representa a duração é apresentado no formato de log_{10} para melhorar a visualização, visto que os valores de mínima e máxima são distantes.

Tabela 4.4: Distribuição da duração das fases de E/S (μs).

Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
0,99	622,99	1151,00	58828394,30	21930,99	75981935004,99

Fonte: Autor (2021).

Figura 4.2: Histograma da duração das fases de E/S em log_{10} (μs).



Fonte: Autor (2021).

Em geral, as fases têm uma ampla faixa de duração, de 1 microssegundo a 21 horas. 50% das fases ocorreram por 1.2ms e 75% ocorreram por 22ms ou menos. Em média, as fases foram executadas por 58,8s, a média é superior à mediana, devido a algumas fases muito longas que aumentaram a média.

Esses resultados indicam que as fases de E/S das aplicações são curtas. Tais fases curtas são plausíveis; por exemplo com uma largura de banda de 1 GB/s, uma fase de 1 ms pode acessar 1MB, com 10 GB/s, 10MB. No supercomputador Intrepid, a largura de banda era de aproximadamente 78GB/s. Portanto, uma fase de 1 ms poderia acessar 78MB. Considerando que, como as aplicações tiveram acessos curtos, possivelmente elas produziram acessos pequenos, porém uma fase pode se repetir durante uma execução, em 75% das aplicações isso ocorre pelo menos 2 vezes (Tabela 4.2).

4.1.3 Duração e Representatividade das Fases de Leitura e Escrita

Na seção anterior, constatou-se que as fases de E/S das aplicações são curtas, na escala de milissegundos ou microssegundos. Ainda assim, queríamos saber se esse comportamento é o mesmo para diferentes padrões de acesso. Além disso, estamos interessados na representatividade das diferentes características consideradas pela nossa estimativa de fase de E/S.

Nesta seção, analisou-se a representatividade das operações de leitura e escrita. Assim, a Tabela 4.5 mostra a distribuição da duração das fases de leitura e escrita. A

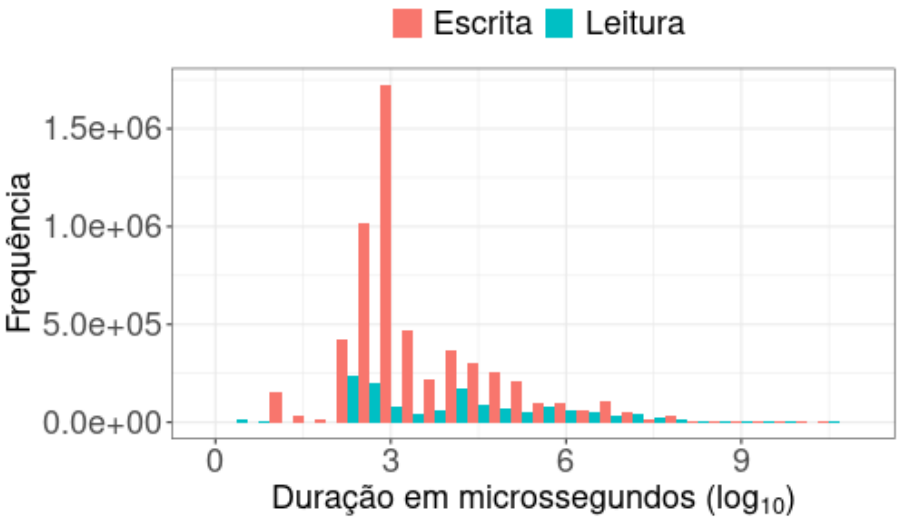
Figura 4.3 mostra o histograma da duração. As cores representam o padrão de acesso (escrita em vermelho e leitura em azul).

Tabela 4.5: Distribuição da duração das fases de leitura e escrita (μs).

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
Leitura	1,99	369,99	10240,00	117503824,22	210500	42148333124
Escrita	0,99	631,00	1066,99	45068655,80	14353	75981935005

Fonte: Autor (2021).

Figura 4.3: Histograma da duração das fases de leitura e escrita em log_{10} (μs).



Fonte: Autor (2021).

As fases de escrita tiveram uma duração mediana de 1ms, enquanto as fases de leitura tiveram uma duração mediana de 10ms. Portanto, as fases de leitura são maiores em 10 vezes se comparadas às de escrita. As fases de leitura ocorreram 1.332.261 vezes, enquanto as fases de escrita ocorreram 5.681.139 vezes. Portanto, 81% das fases foram escritas. É importante observar que essas são fases do lado das aplicações e não representam necessariamente a carga de trabalho do ponto de vista do servidor. No entanto, podemos dizer que os *jobs* tiveram mais fases de escrita do que de leitura (81% vs. 19%).

O tempo total gasto no ano inteiro nas fases de leitura foi de 43484 horas e nas fases de escrita foram de 71122 horas. Essa proporção 62% versus 38% não é a mesma que a observada ao se analisar o número de fases (81% vs 19%). Isso é explicado por uma maior duração das fases de leitura (10ms vs. 1ms).

Até agora, analisou-se a diferença entre operações de leitura e escrita. Para verificar se esse comportamento é mantido para as duas interfaces de acesso disponíveis,

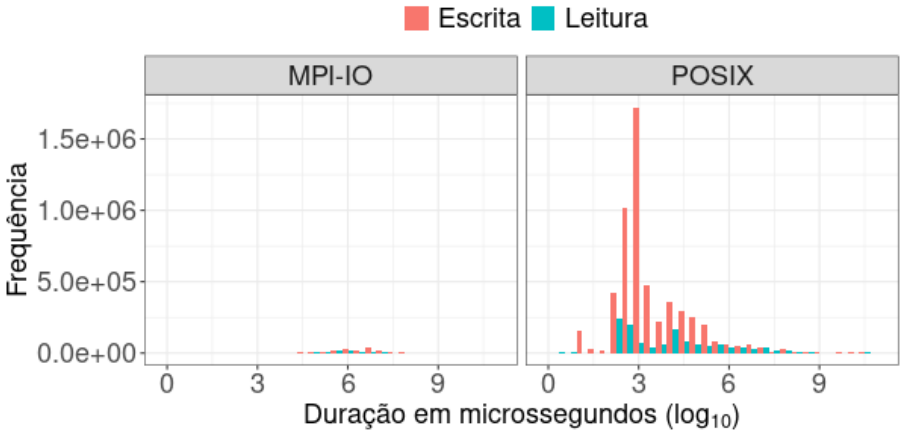
MPI-IO e POSIX, a Tabela 4.6 mostra a distribuição da duração de leitura e escrita para MPI-IO e POSIX. Além disso, a Figura 4.4 mostra o histograma da duração. As cores representam o padrão de acesso (escrita em vermelho e leitura em azul) e cada faceta mostra as interfaces MPI-IO e POSIX.

Tabela 4.6: Distribuição da duração das fases de leitura e escrita para cada interface (μs).

	Op.	Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
MPI-IO	**Leitura**	208,99	396896,00	663400,00	86196719,05	1851388,00	37889646904
	Escrita	397,99	622347,75	2365935,50	182778052,41	6330947,25	42792257294
POSIX	**Leitura**	1,99	350,99	8624	119305593,16	102233,25	42148333124
	Escrita	0,99	628,00	1015	41080469,00	10739,00	75981935005

Fonte: Autor (2021).

Figura 4.4: Histograma da duração das fases de escrita e leitura para cada interface em log_{10} (μs).



Fonte: Autor (2021).

Entre as fases MPI-IO, 69% das fases foram escritas e 31% foram leituras (159.900 a 72.501). As fases de escrita foram mais longas das que as fases de leitura, cerca de 3,5 vezes. Entre as fases POSIX, 81% das fases foram escritas contra 18% de fases de leitura. Ao contrário das fases MPI-IO, onde as escritas foram mais longas que as leituras, agora, para as fases em POSIX, as fases leituras duraram mais do que as de escritas cerca de 8 vezes. A duração e a proporção das fases de leitura e gravação do POSIX são semelhantes ao comportamento geral mostrado na Tabela 4.2 porque existem mais fases do POSIX do que do MPI-IO.

As fases POSIX representam 97%, enquanto apenas 3% das fases são MPI-IO (6.780.999 a 232.401). Os *jobs* gastaram 104.753 horas nas fases POSIX e 9,854 horas nas fases MPI-IO; portanto, as fases POSIX executaram 10,6 vezes mais durante o ano.

A diferença no tempo (MPI-IO com 8,5% e POSIX com 91,5%) é menor que a diferença no número de fases (MPI-IO com 3% e POSIX com 97%) porque as fases do POSIX são mais curtas. Sua duração mediana foi de 1ms, contra 1s das fases MPI-IO.

Essa disparidade entre o uso de POSIX e MPI-IO leva a perceber que as aplicações não estão utilizando interfaces de mais alto nível que é o caso do MPI-IO, onde por exemplo, é possível fazer operações coletivas, agregações e aplicar algumas otimizações de E/S. Estes tipos de operações se tornam bastante custosas para implementar em POSIX, deixando a cargo do desenvolvedor coordenar os processos e realizar as comunicações.

4.1.4 Duração e Representatividade de Fases com Acessos a Arquivos Únicos e Compartilhados

Nas fases de E/S, os processos podem acessar arquivos independentes (únicos) ou compartilhados. Nesta seção, foi investigado se a duração das fases é diferente, dependendo desse aspecto. A Tabela 4.7 mostra a distribuição da duração separada por tipo de acesso ao arquivo (único ou compartilhado). Além disso, a Figura 4.5 mostra o histograma da duração. As cores representam o tipo de acesso ao arquivo (compartilhados em vermelho e únicos em azul).

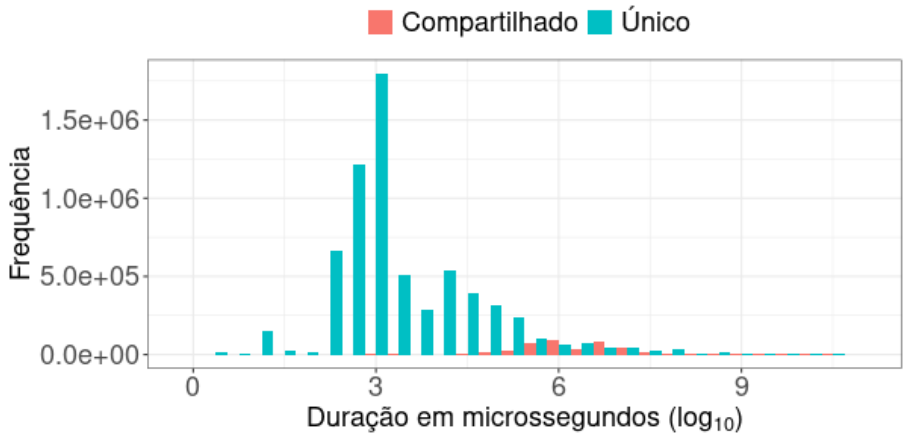
Tabela 4.7: Distribuição da duração das fases com acessos a arquivos únicos e compartilhados (μs).

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
Únicos	0,99	612,99	1035	43727018,07	12891	75981935005
Compartilhados	15,00	597593,00	1361261	298674318,07	6400153	58781866060

Fonte: Autor (2021).

A maioria dos acessos foram a arquivos únicos, com um total de 6.597.973 (94% das fases). No entanto, as aplicações passaram no total 80.141 (70%) horas acessando arquivos únicos, enquanto gastaram 34.465 (30%) horas acessando arquivos compartilhados. Portanto, as fases, onde os arquivos compartilhados são acessados, são mais longas. De fato, a duração mediana foi de 1,3s, enquanto as fases de *arquivo único* duraram 1ms. Isso remete a um senso comum: acessos em arquivos únicos acontecem de forma mais rápida, onde os processos abrem, acessam, e fecham o arquivo logo em seguida. Porém, em acessos com arquivos compartilhados, o processo de fechamento pode ser mais difícil de ser coordenado, devido a sincronizações, já que isso envolve sincronizações entre os

Figura 4.5: Histograma da duração das fases com acessos a arquivos únicos e compartilhados em log_{10} (μs).



Fonte: Autor (2021).

Tabela 4.8: Distribuição da duração das fases com acessos a arquivos únicos e compartilhados para cada interface (μs).

	Arquivo	Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
MPI-IO	**Únicos**	208,99	62616,74	213075	44259794,88	2670401	37889646904
	Comp.	285,00	659971,00	1578860	183289485,18	6264227	42792257294
POSIX	**Únicos**	0,99	611,99	1018,00	43722849,77	11999	75981935005
	Comp.	15,00	522519,00	1199586,50	387920407,25	6583367	58781866060

Fonte: Autor (2021).

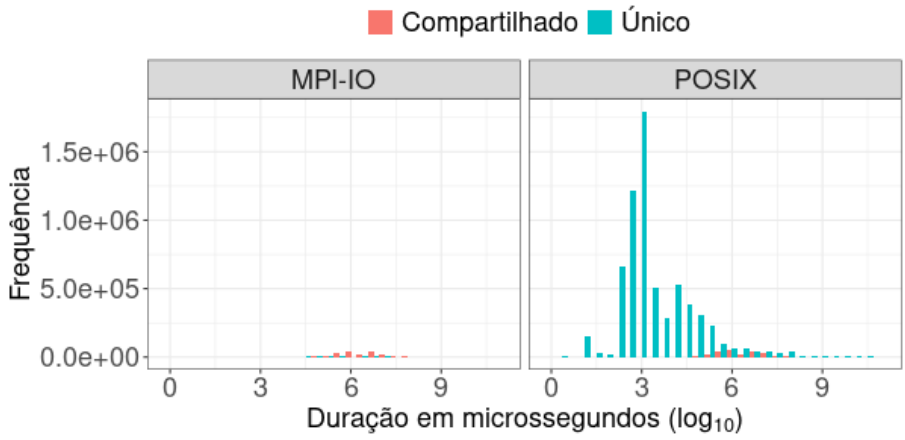
processos, especialmente quando os acessos são realizados em POSIX.

Para investigar se a duração de fases com acessos a arquivos únicos e compartilhados é semelhante para as duas interfaces, POSIX e MPI-IO, a Tabela 4.8 mostra a duração da distribuição por interface (MPI-IO e POSIX). Além disso, a Figura 4.6 mostra o histograma da duração. As cores representam o tipo de acesso ao arquivo (compartilhados em vermelho e únicos em azul) e cada e faceta representa as interfaces (MPI-IO e POSIX).

Entre as fases que usaram o MPI-IO, as *fases de arquivos compartilhados* foram mais comuns, elas representam 78% (181.181 de 232.401 acessos) das fases e 94% (9.224 horas) do tempo gasto. Além disso, para as fases MPI-IO, em que arquivos únicos foram acessados, estas foram mais longas que o observado em geral (Tabela 4.7), com uma mediana de 213,07 milissegundos. As fases POSIX em que arquivos únicos são acessados representam 99,2% de todos os acessos, é por isso que o POSIX tem um comportamento semelhante ao observado em geral.

Separou-se esses resultados pela operação para ver se o comportamento seria diferente do que foi observado em geral. A Tabela 4.9 apresenta a duração das fases separadas

Figura 4.6: Histograma da duração das fases com acessos a arquivos únicos e compartilhados para cada interface em log_{10} (μs).



Fonte: Autor (2021).

por tipo de acesso ao arquivo e por operação. Além disso, a Figura 4.7 apresenta o histograma da duração. As cores representam o tipo de acesso ao arquivo (compartilhados em vermelho e únicos em azul) e cada faceta representa as operações (escrita e leitura).

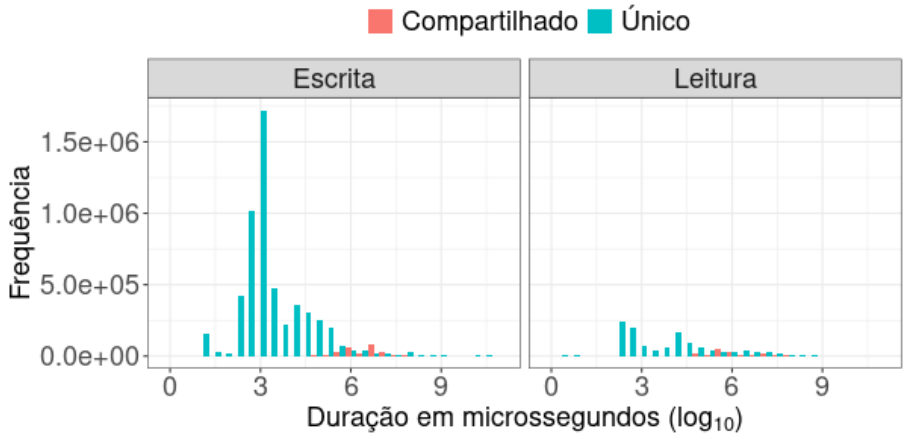
Tabela 4.9: Distribuição da duração das fases com acessos a arquivos únicos e compartilhados para cada operação (μs).

	Arquivo	Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
Escrita	**Únicos**	0,99	626,00	980	39490251,19	9168	75981935005
	Comp.	15,00	776099,99	3805274	161446573,93	6681542	58781866060
Leitura	**Únicos**	1,99	321,99	5555,00	63246682,09	44848,50	41699453454
	Comp.	94,00	409617,75	662482,50	527902676,58	2291544,75	42148333124

Fonte: Autor (2021).

As aplicações passaram no total 11.653 horas escrevendo arquivos compartilhados e 59.468 horas escrevendo em arquivos únicos. No entanto, as fases, onde os arquivos compartilhados foram escritos, eram 388.193% mais longas que as fases de escrita em arquivos únicos. O motivo desse contraste (as fases mais curtas são responsáveis pela maior parte do tempo) é que 94% das fases de escrita eram arquivos únicos. Por outro lado, nas fases de leitura, um total de 22812,14 horas foram gastas lendo arquivos compartilhados 20672,7 horas lendo arquivos únicos. As fases dos arquivos únicos ainda são mais frequentes (656,40%), mas a proporção é menor do que o observado nas fases de escrita.

Figura 4.7: Histograma da duração das fases com acessos a arquivos únicos e compartilhados para cada operação em log_{10} (μs).



Fonte: Autor (2021).

4.1.5 MPI-IO: Acessos Independentes e Coletivos

Os processos MPI-IO podem acessar de duas formas os arquivos. Os acessos podem ser independentes ou coletivos. Os acessos independentes são caracterizados por cada processo manipulando sua E/S independentemente de outros processos. Os acessos coletivos são caracterizados por chamadas de E/S que devem ser feitas por todos os processos que participam de uma operação de E/S específica. Estes processos compartilham o mesmo comunicador MPI para sincronizar os acessos. A Tabela 4.10 apresenta a distribuição da duração das fases independentes, coletivas e desconhecidas. Ainda, a Figura4.8 demonstra o histograma da duração. As cores representam o tipo de acesso ao arquivo que os processos MPI-IO fizeram.

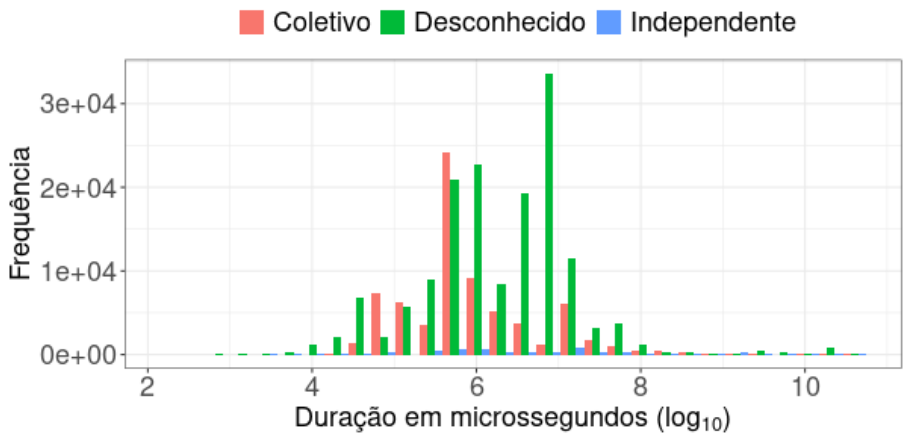
Tabela 4.10: Distribuição da duração das fases MPI-IO independentes, coletivas e desconhecida (μs).

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
Independente	208,99	557890,75	5443121,50	1234374901,25	39196527,25	42523536027,00
Coletiva	285,00	381863,25	636187,00	76210601,07	1743994,75	40757145653,00
Desconhecida	397,99	643558,49	2461551,00	148806142,89	6306651,00	42792257294,00

Fonte: Autor (2021).

Uma parcela significativa das fases MPI-IO não é coletiva ou independente; elas são caracterizadas como desconhecidas (66,35%). No entanto, o MPI-IO, em geral, não representa o maior comportamento, apenas 3% dos acessos são MPI-IO. As fases em que os acessos independentes ocorreram foram executadas por 5,44s, enquanto as fases

Figura 4.8: Histograma da duração das fases MPI-IO independentes, coletivas e desconhecida em log_{10} (μs).



Fonte: Autor (2021).

com acessos coletivos foram executadas por 636.1 milissegundos e as desconhecidas são executadas por 2.4s. As fases com processos acessando de forma independente foram as mais longas, mas elas representaram apenas 2,44% das fases MPI-IO.

4.1.6 POSIX: Acessos Consecutivos, Sequenciais e Randômicos

Diferente da informação disponibilizada para o MPI-IO, onde caracterizou-se os acessos pela organização dos processos, para o POSIX, caracterizou-se pela espacialidade na ordem dos acessos a um arquivo. Três espacialidades nos acessos são caracterizadas nas fases POSIX. Os acessos consecutivos são caracterizados por acessos que acontecem imediatamente adjacentes aos acessos anteriores. Por outro lado, os acessos sequenciais acontecem em um deslocamento mais alto do que onde o último acesso anterior parou, ou seja, consecutivos também são sequenciais. No entanto, a diferença entre eles é o deslocamento do acesso. Por fim, acessos randômicos são aqueles que não foram caracterizados por consecutivos ou sequenciais. A Tabela 4.11 mostra a distribuição da duração das fases em que a espacialidade é caracterizada por acessos consecutivos, sequenciais e randômicos. Além da tabela, a Figura 4.9 demonstra o histograma da duração. As cores representam a espacialidade dos acessos POSIX.

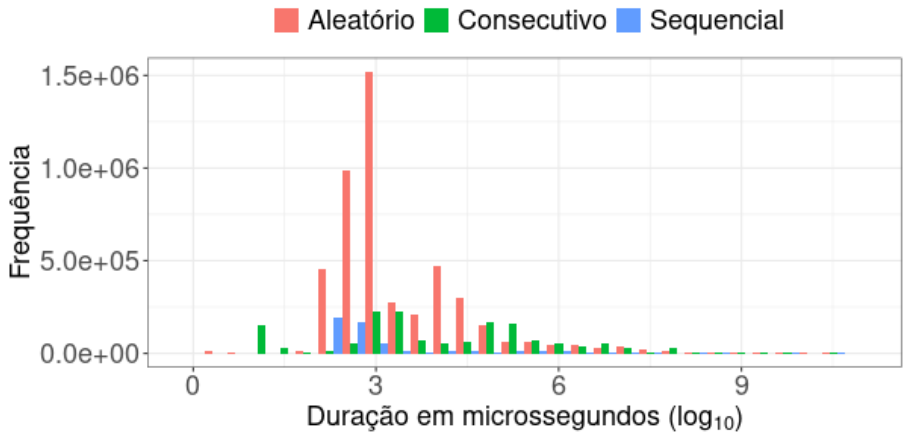
Os acessos randômicos representaram o maior número de fases, um total de 4.720.461 em 6.780.999, o que representa 70% dos acessos. No entanto, não são os acessos mais longos; neste caso, os consecutivos representam 70% do tempo total gasto em acessos POSIX durante o ano todo, embora estes representam apenas 22% no número total de

Tabela 4.11: Distribuição da duração das fases POSIX considerando a espacialidade (μs).

	Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
Consecutivos	13,99	1203	5919,99	101005931,36	164042,00	75981935005
Sequenciais	119,00	212,99	381,99	163234475,49	997,99	42895687341
Randômicos	0,99	626	877,00	28691695,89	10047,00	58781866060

Fonte: Autor (2021).

Figura 4.9: Histograma da duração das fases POSIX considerando a espacialidade em log_{10} (μs).



Fonte: Autor (2021).

fases. O motivo de representaram 70% do tempo gasto com somente 22% de fases foi fases que executaram por mais tempo. De fato, sua duração mediana foi de 6ms contra 381 e 877 microssegundos dos acessos sequenciais e randômicos, respectivamente.

4.1.7 Estabilidade dos Padrões de Acesso

Nas seções anteriores, a representatividade dos padrões de acesso foi analisada. No entanto, considerando que o objetivo é entender o comportamento de E/S, surge a pergunta: "por quanto tempo um *job* continua com o mesmo padrão de acesso?" e, para responder isso, teria que ser investigada a estabilidade de uma fase de E/S. Para tanto, modificou-se o conjunto de dados, para que períodos inativos entre fases de E/S idênticas sejam ignorados. Ou seja, intervalos inativos onde as fases anterior e posterior sejam idênticas são caracterizados como parte de uma única fase de E/S. Esta etapa foi implementada no Spark juntamente com Python e foi executada no cluster LSD.

Identicamente como foi analisado posteriormente, analisou-se a duração das fases de E/S, dado que pode caracterizar a carga de trabalho de E/S dos padrões de acesso das

aplicações. Da mesma forma que foram realizadas análises de distribuição e de representatividade nas Seções de 4.1.2 até 4.1.6, nesta seção realizou-se as mesmas análises, porém somente os resultados mais relevantes serão apresentados.

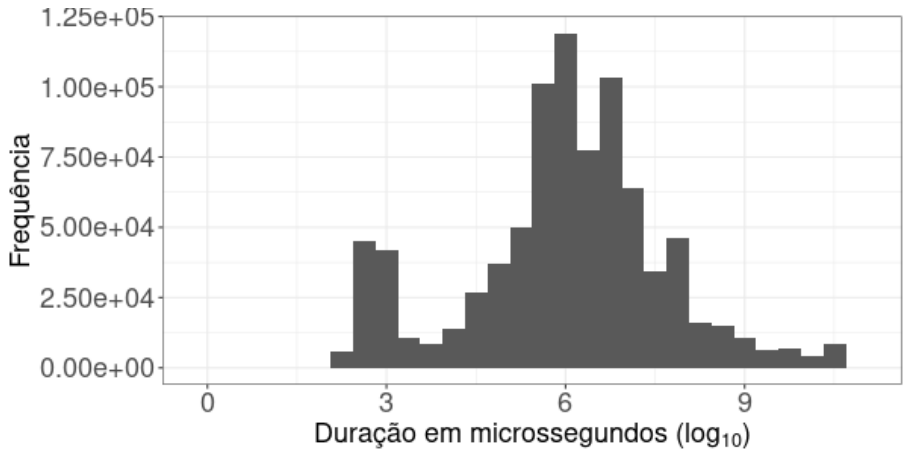
A primeira análise mostra a duração em geral, considerando todas as fases de E/S. A Tabela 4.12 apresenta a distribuição da duração das fases em microssegundos. Além disso, a Figura 4.10 mostra o histograma da duração.

Tabela 4.12: Distribuição da duração das fases agregadas de E/S (μs).

Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
0,99	198084,74	1036162,99	514619337,49	8377390,74	75981935004,99

Fonte: Autor (2021).

Figura 4.10: Histograma da duração das fases agregadas de E/S em log_{10} (μs).



Fonte: Autor (2021).

Os comportamentos são constantes por um tempo relativamente longo quando comparado com a análise da Seção 4.1.2. Tem-se que 75% das vezes o mesmo padrão é mantido por 8.377ms ou menos. A duração mediana considerando a estabilidade é de aproximadamente 1s. Comparando com a análise anterior a mediana da duração das fases foi de 1ms. A grande diferença entre essa e a duração da fase (1s vs. 1ms de mediana) vem de duas frentes: primeiro, a maneira como os dados do Darshan foram coletados e a nossa metodologia de estimar as fases. Segundo, e mais importante, as aplicações em CAD tendem a repetir os padrões de acesso periodicamente, como demonstrado na literatura (BOITO et al., 2018). Por exemplo, o uso de *checkpointing* faz com que o mesmo padrão de gravação aconteça a cada **N** segundos durante toda a vida útil da aplicação (SHAHZAD et al., 2013)

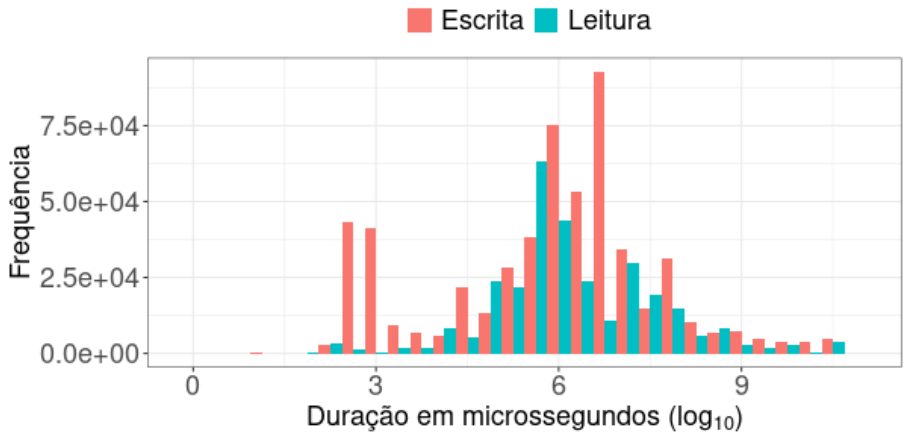
Buscando entender as diferentes características que compõem uma fase de E/S, analisou-se a representatividade das operações de leitura e escrita da mesma forma que foi realizado na Seção 4.1.3. Desta forma, a Tabela 4.13 mostra a distribuição da duração das fases de leitura e escrita. A Figura 4.11 mostra o histograma da duração. As cores representam o padrão de acesso (escrita em vermelho e leitura em azul).

Tabela 4.13: Distribuição da duração das fases agregadas de escrita e leitura (μs).

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
Escrita	0.99	74460	1312424	495148332.91	6867399	75981935005
Leitura	88,00	333815	810675	550833070.87	14739903	42148333124

Fonte: Autor (2021).

Figura 4.11: Histograma da duração das fases agregadas de escrita e leitura em log_{10} (μs).



Fonte: Autor (2021).

A proporção de leitura para escrita é maior quando comparado com análise anterior (35% a 65% em vez de 19% a 81%), o que nos leva a concluir que, ao ignorar as fases ociosas, mesclamos mais escritas do que leituras, ou seja, escritas são mais periódicas do que leituras. Isso também é justificado pelo aumento mediano da duração das fases de escrita que passaram de 1ms para 1,3s. Além disso, este comportamento corresponde ao senso comum: as aplicações CAD lerão no início e escreverão periodicamente durante a execução (LUU et al., 2013; BEHZAD et al., 2014).

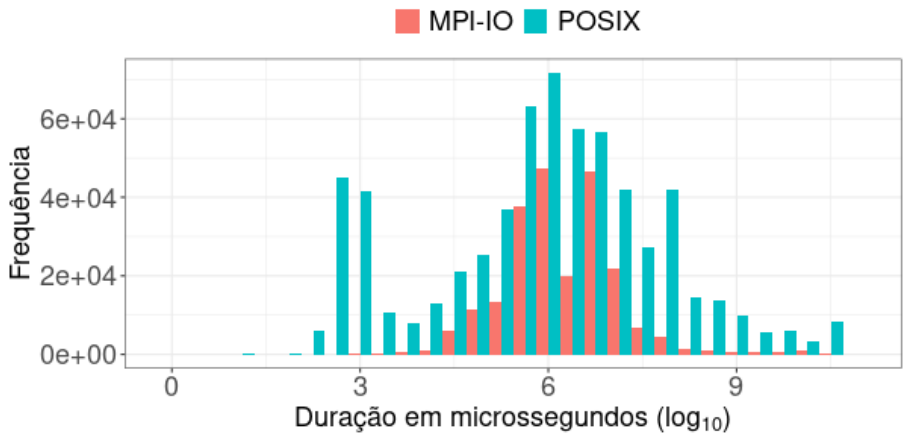
Além de considerar as operações de leitura e escrita, analisou-se a distribuição e a representatividade das fases em que a interface utilizada foi POSIX e MPI-IO. A Tabela 4.14 mostra a distribuição da duração das fases MPI-IO e POSIX. A Figura 4.12 apresenta o histograma da duração. As cores representam o padrão de acesso (MPI-IO em vermelho e POSIX em azul).

Tabela 4.14: Distribuição da duração das fases agregadas MPI-IO e POSIX (μs).

	Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
MPI-IO	208,99	526786,25	1187747	160319449,85	6155900,50	42792257294
POSIX	0,99	78251,25	985841,50	640600094,53	12284308,50	75981935005

Fonte: Autor (2021).

Figura 4.12: Histograma da duração das fases agregadas MPI-IO e POSIX em log_{10} (μs).



Fonte: Autor (2021).

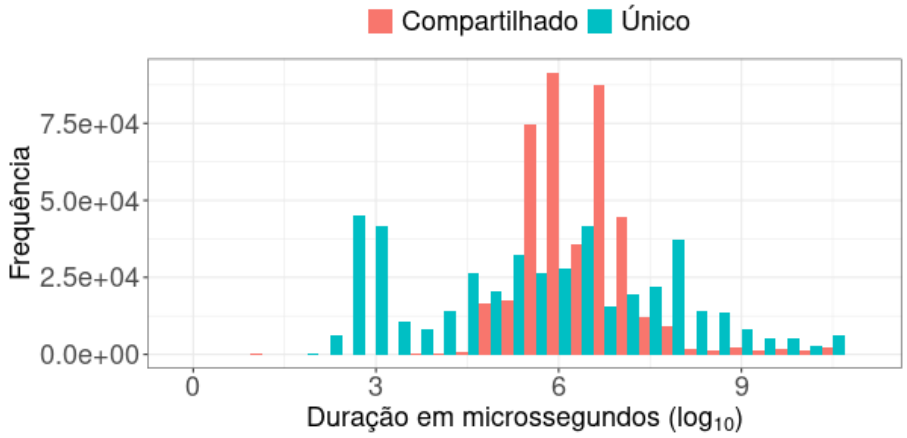
Na Seção 4.1.3 foi apresentada a proporção entre o número de fases POSIX e MPI-IO, onde 97% das fases foram caracterizadas como POSIX e apenas 3% representaram fases MPI-IO. Com a análise da estabilidade, houve uma mudança nesta proporção, passando-a para 74% POSIX e 26% MPI-IO. Apesar do aumento de fases MPI-IO, a duração mediana não foi modificada na mesma proporção, sendo que esta aumentou em aproximadamente 18% em relação à análise anterior, onde o tempo de duração mediano foi de 1 segundo, contra o atual de 1,18s. Já para as fases POSIX o aumento foi mais significativo, um total de 90.612% em relação às fases anteriores, passando de 1ms para 985 ms. Isso indica que as fases POSIX são geralmente mais repetidas com intervalos inativos entre elas, isso é justificado também pelo fato em que originalmente as fases POSIX eram mais representativas nos dados, o que indica que MPI-IO foi pouco utilizado nas aplicações. Uma característica dos padrões de acesso analisada foi a forma em que os processos acessam os seus arquivos. Neste caso, os processos podem acessar arquivos de maneira compartilhada ou única. A Tabela 4.15 mostra a distribuição da duração das fases em que os processos acessavam arquivos compartilhados e únicos. A Figura 4.13 mostra o histograma da duração. As cores representam o padrão de acesso (vermelho para compartilhado e azul para único).

Tabela 4.15: Distribuição da duração das fases agregadas com acessos compartilhados e únicos (μs).

	Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
Compartilhado	15,99	622592	1539359,00	309917415,41	6464084,00	58781866060
Único	0,99	9511	508884,50	697610932,11	25872326,50	75981935005

Fonte: Autor (2021).

Figura 4.13: Histograma da duração das fases agregadas com acessos compartilhados e únicos em log_{10} (μs).



Fonte: Autor (2021).

A proporção entre os acessos era de 94% e 6% para os acessos únicos e compartilhados, respectivamente (Seção 4.1.3). Agora na análise de estabilidade destes acessos, a proporção foi de 52% e 48%. Houve um aumento na duração de ambos os acessos, 13% para os acessos em arquivos compartilhados e 49.067% para os acessos à arquivos únicos. Isso indica que é mais comum ter intervalos inativos entre acessos únicos do que em acessos compartilhados. Porém os acessos compartilhados ainda são mais longos do que os acessos únicos em cerca de 202%.

4.2 Análise do Comportamento de E/S no Nível do Sistema

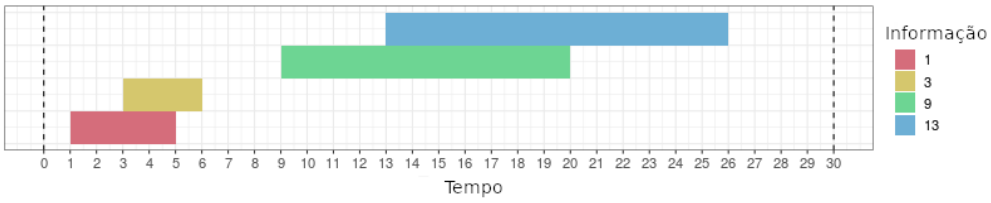
A Seção 4.1 investigou o comportamento de E/S no nível das aplicações sem considerar a sobreposição de *jobs* paralelos. Considerar a sobreposição é essencial para entender a carga de trabalho que foi enviada ao SAP. Com base nisso, esta seção procura explicar o comportamento de E/S, considerando a sobreposição dos *jobs*.

Portanto, para criar um conjunto de dados que considere a sobreposição dos *jobs*, criou-se um algoritmo para tratar os CSVs que continham as fases de E/S dos *jobs* divi-

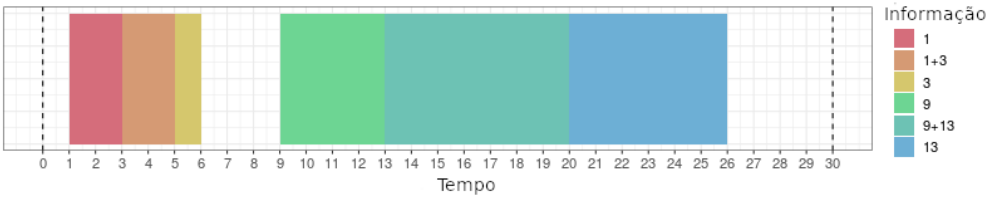
didos por dia. Estes CSVs foram criados na metodologia explicada no Capítulo 3. Esse algoritmo foi implementado em C++ utilizando uma estrutura *heap* mínima, resultando em um código com complexidade $O(nlog(n))$, onde n é número de fases de E/S que estavam presente nos CSVs. Aqui, a Fase de E/S contém um timestamp de início e de fim e os padrões de acesso que foram caracterizados naquele período.

O objetivo era em que cada nova fase contenha informações sobre todas as fases de E/S que se sobrepõem no tempo. Portanto, a Figura 4.14(a) apresenta exemplos de fases, cada um com seus respectivos padrões de acesso. Por exemplo, pode-se dizer que as fases 1 e 9 são o mesmo *job*, e os 3 e 13 são outro *job*. A Figura 4.14(b) demonstra o resultado do processo de sobreposição usando o código C++. A saída desse programa é armazenada em um arquivo CSV, em que cada entrada representa um novo intervalo.

Figura 4.14: Exemplo de Fases no Processo de Sobreposição.



(a) Exemplo de fases de E/ de *jobs*.



(b) Resultados do exemplo de fases.

Fonte: Autor (2021).

Após a criação do arquivo final que continha todos as fases, foi necessário executar mais um passo para combinar as fases consecutivas idênticas. Isso foi necessário, pois o algoritmo executado para criar as fases não mesclava fases consecutivas idênticas, o que é importante para entendermos a estabilidade do sistema. Para combinar as fases foi criado um algoritmo implementado em C++ com complexidade $O(n)$, onde n é o número de fases.

4.2.1 Distribuição da Duração das Fases

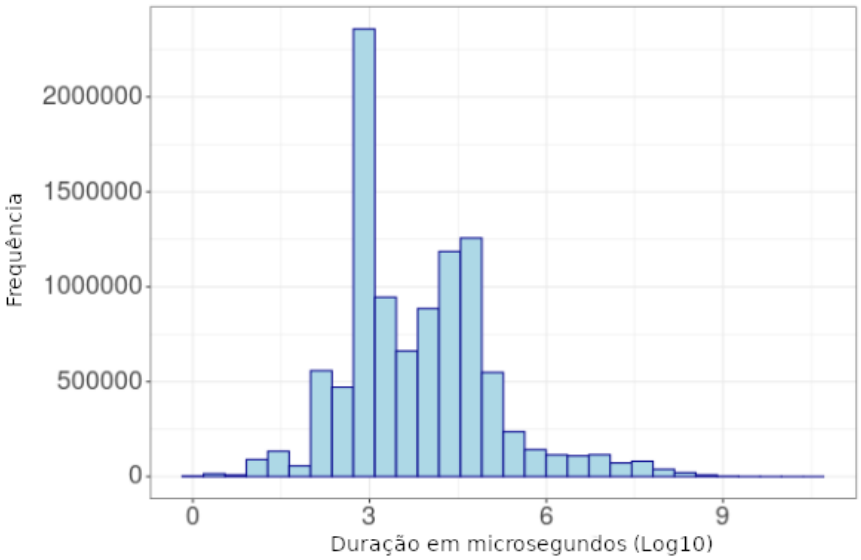
Considerando uma visão do sistema, diversas fases de E/S podem ocorrer de forma paralela, ou seja, existe mais de um padrão de acesso em uma fase de E/S. Portanto, cada fase contém todos os padrões de acessos que ocorrem naquele instante de tempo. Caracterizar a duração das fases agora, responde a estabilidade dos acessos enviados ao sistema, isto é, o tempo que o sistema recebe o mesmo comportamento. A Tabela 4.16 apresenta a distribuição da duração das fases de E/S. Além da tabela, a Figura 4.15 demonstra o histograma da duração.

Tabela 4.16: Distribuição da duração das fases de E/S a nível de sistema (μs).

Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
1.000e+00	6.970e+02	4.743e+03	2.281e+06	3.922e+04	3.442e+10

Fonte: Autor (2021).

Figura 4.15: Histograma da duração das fases de E/S a nível de sistema em log_{10} (μs).



Fonte: Autor (2021).

As fases executam em mediana 4,7ms, isso é maior do que o que foi observado na análise no nível de aplicações (Seção 4.1.1), é menor do que o observado na análise no nível de aplicações mas agregando fases consecutivas idênticas (Seção 4.1.7). A mediana é maior que a nível de aplicação, pois aqui, temos a estabilidade das fases a nível de sistema. Ela é menor, porque aqui uma fase pode conter mais do que um padrão de acesso, então é provável que existam mais mudanças de comportamento, visto que basta

um *job* paralelo mudar seu padrão de acesso, que a caracterização da fase muda. Isso se aplica também para os *jobs*, uma fase pode conter inúmeros *jobs*, uma mudança de *jobs* afeta a caracterização das fases.

Essa propriedade de mudança da característica da fase com o começo e o fim de novos padrões de acessos ou *jobs*, faz com que quando ocorra as mudanças de fases, a carga de trabalho não necessariamente mude completamente.

Portanto, concluímos que a duração das fases, quando analisada no nível do sistema está conectada ao número de processos e aplicações que executam E/S, ou seja, está ligada ao uso da máquina do que com os padrões de acesso.

4.2.2 Análise do Número de Padrões de Acesso Simultâneos

Como foi visto na seção anterior, a duração das fases de E/S quando analisada no nível de sistema está mais ligado ao número de aplicações/*jobs*, com isso nesta seção analisou-se a distribuição do número de padrões de acesso que compõe uma fase de E/S. Este número corresponde a concorrência que ocorre no sistema de E/S.

Para a análise da distribuição considerou-se uma análise de variável quantitativa discreta dos valores, isso porque os dados são frutos da contagem dos padrões de acesso de cada fase de E/S. A Tabela 4.17 apresenta a distribuição do número de padrões de acesso nas fases de E/S. Além da tabela, a Figura 4.16 apresenta nas barras a frequência relativa e na linha a frequência cumulativa ambas em porcentagem. Os valores de frequência entre 13 e 17 representam uma escala pequena (menos de 1%) em relação ao restante dos valores, por isso eles não apresentam valores na figura.

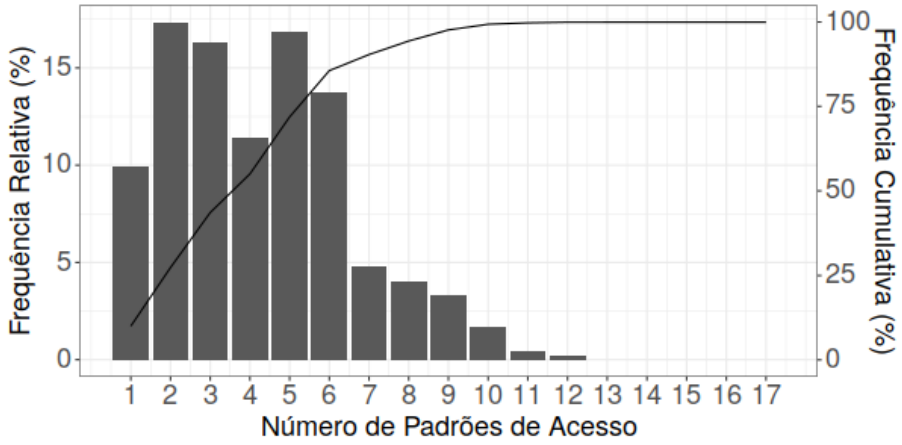
Tabela 4.17: Distribuição do número de padrões de acesso simultâneos de E/S.

Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
1.00	2.00	4.00	4.24	6.00	17.00

Fonte: Autor (2021).

O número de padrões de acesso contidos nas fases de E/S estão entre 1 e 17 padrões. A mediana foi 4 o que indica que os sistema trabalhou em 50% das vezes com 4 padrões de acesso simultâneos ou menos. Selecionando a maior frequência relativa (17,34%) as fases possuíam dois padrões de acesso concorrentes. Além disso, quando analisado a frequência acumulativa percebemos que em 99,36% das vezes a carga de tra-

Figura 4.16: Frequência relativa e cumulativa dos padrões de acesso simultâneos.



Fonte: Autor (2021).

balho do sistema foi formada por e 10 ou menos padrões simultâneos.

Outro ponto que foi considerado nesta análise foi a ponderação das fases pela duração delas, isso faz com que as fases sejam representadas pelo seu tempo de duração e não pela quantidade de vezes que ocorreram. Desta forma a Tabela 4.18 apresenta a distribuição do número de padrões de acesso nas fases de E/S. Além da tabela, a Figura 4.17 apresenta nas barras a frequência relativa e na linha a frequência cumulativa ambas em porcentagem. Os valores de frequência para 16 e 17 padrões de acesso representam uma escala pequena (menos de 1%) em relação ao restante dos valores, por isso eles não apresentam valores na figura.

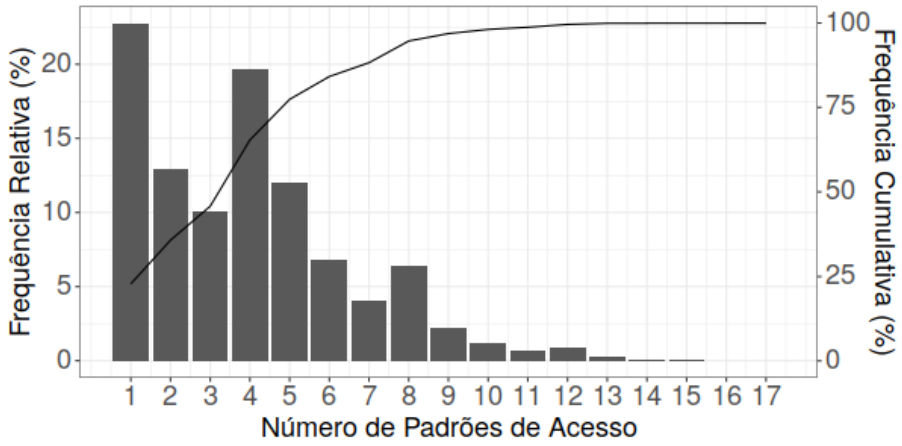
Tabela 4.18: Distribuição ponderada do número de padrões de acesso nas fases de E/S.

Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
1.00	2.00	4.00	3.92	5.00	17.00

Fonte: Autor (2021).

Analisando de forma ponderada pela duração, isso faz com que as fases sejam representadas pelo seu tempo de duração e não pela quantidade de vezes que ocorreram. Assim, podemos dizer que isso representa o tempo em que o sistema recebeu requisições de um determinado conjunto de padrões de acesso. Portanto, com a mediana igual a 4 o mesmo que foi visto anteriormente, temos que em 50% do tempo, o sistema trabalhou com 4 ou menos padrões de acesso simultâneos. A maior frequência relativa foi de 22,77%, ou seja, em aproximadamente 23% do tempo o sistema trabalhou com somente um padrão de acesso. Comparando esta distribuição ponderada com a anterior que considerava as ocorrências, percebe-se que fases com menos padrões simultâneos são mais

Figura 4.17: Frequência relativa e cumulativa dos padrões de acesso simultâneos ponderados.



Fonte: Autor (2021).

representativas na carga do trabalho.

4.2.3 Análise do Número de *jobs* Simultâneos

Dado que a duração das fases de E/S a nível de sistema é mais influenciada pela carga de trabalho do sistema, o número de *jobs* que compõem uma fase é diretamente ligado a mudança de comportamento das fases, dado que quando a mudança de *jobs* nas fases muda-se o caracterização dela, podendo mudar a carga de trabalho do sistema. Da mesma forma que se analisou o número de padrões de acesso simultâneos dentro de uma fase de E/S, buscou-se analisar o número de *jobs* simultâneos. Foi realizado o mesmo tipo de análise de variável quantitativa discreta.

A Tabela 4.19 apresenta um resumo da distribuição do número de *jobs* que compõem uma fase de E/S. Como pode ser visto, uma fase pode conter de 1 até 256 *jobs* simultâneos. A mediana foi 4 *jobs* simultâneos, ou seja, em 50% das vezes, as fases foram compostas por 4 ou menos *jobs*. A média foi de 8,7 *jobs* por fase, este valor é maior que a mediana pelo fato de existirem fases com até 256 *jobs* simultâneos.

Tabela 4.19: Distribuição do número de *jobs* que compõe uma fase de E/S.

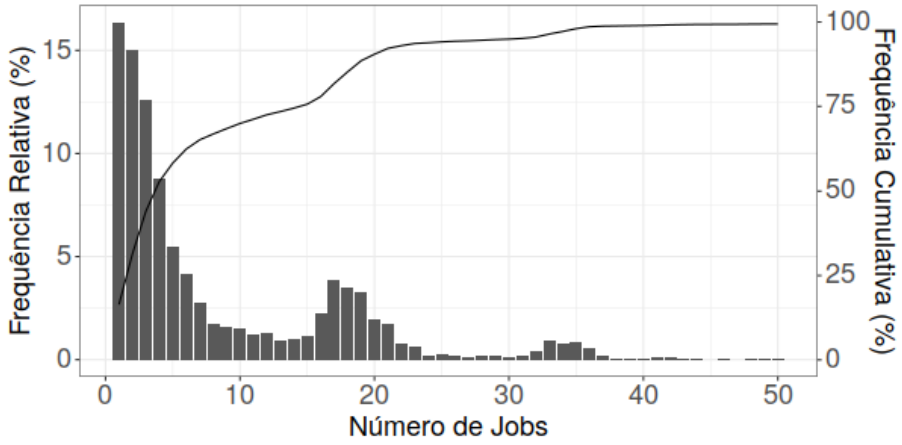
Mín.	1º Q.	Mediana	Média	3º Q.	Máx.
1.00	2.00	4.00	8.74	15.00	256.00

Fonte: Autor (2021).

A Figura 4.18 apresenta a frequência relativa no eixo y da esquerda e que é re-

presentado pelas barras, também apresenta a frequência cumulativa no eixo y da direita, representado pela linha. No eixo x apresenta o número de *jobs* entre 1 e 50. Quantidades entre 51 e 256 foram omitidas para auxiliar a visualização, por representarem menos de 1% das ocorrências.

Figura 4.18: Frequência relativa e cumulativa do número de *jobs*.



Fonte: Autor (2021).

Uma análise ponderada também foi aplicada para os *jobs*, assim, o número de *jobs* presentes em cada fase são representados pela duração da fase e não mais pela quantidade de vezes que ocorreram. Isso implica que, mesmo ocorrências que aconteceram poucas vezes, porém executaram por mais tempo, tenham maior peso. Desta forma, a Tabela 4.20 apresenta a distribuição do número de *jobs* que compõe uma fase de E/S ponderados pela sua duração. Além da tabela, a Figura 4.19 apresenta nas barras a frequência relativa e na linha a frequência cumulativa ambas em porcentagem. Da mesma forma que apresentado na Figura 4.17, alguns valores foram removidos para melhor visualização.

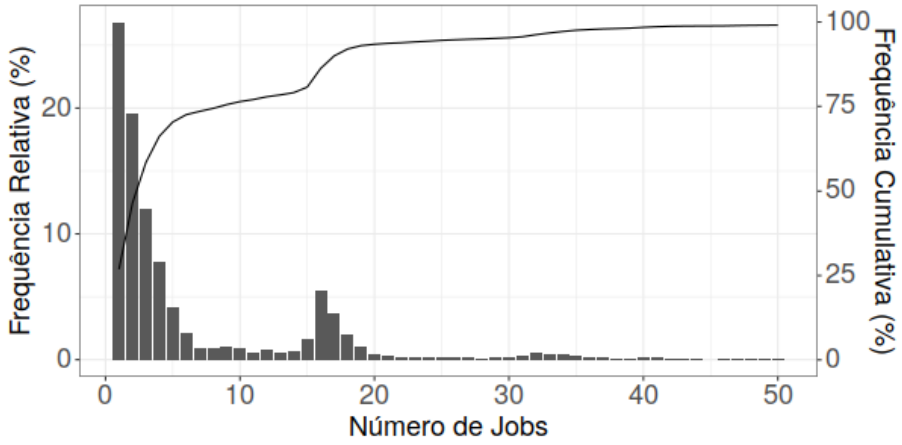
Tabela 4.20: Distribuição ponderada do número de *jobs* que compõe uma fase de E/S.

Mín.	**1º Q.**	**Mediana**	**Média**	**3º Q.**	**Máx.**
1.00	1.00	3.00	7.17	9.00	256.00

Fonte: Autor (2021).

A distribuição ponderando os número de *jobs* pela duração da fase (Tabela 4.19) fez com que o primeiro quartil, mediana, media e o terceiro quartil diminuíssem seus valores. Agora, a mediana apresenta é de 3, ou seja, em 50% do tempo 3 *jobs* realizaram operações de E/S. Observando a Figura 4.17, percebe-se que fases com somente 1 *job* representam 26,83% do tempo. e que em 99% do tempo 47 ou menos *jobs* fizeram parte de uma fase de E/S.

Figura 4.19: Frequência relativa e cumulativa do número de *jobs* ponderada.



Fonte: Autor (2021).

4.2.4 Correlação Entre o Número Padrões de Acesso e de *Jobs* Presentes em uma Fase de E/S

Aplicou-se o coeficiente de correlação de Pearson (FREEDMAN; PISANI; PURVES, 2007) para duas variáveis. Neste caso, o número de padrões de acesso e o número de *jobs* presentes nas fases de E/S. O coeficiente é um valor entre -1 e +1, que explica a correlação entre as duas variáveis. O valor de +1 é a correlação linear positiva total, 0 é não correlação linear e -1 e correlação linear negativa total. A formula do coeficiente é expressa por:

$$\rho_{X,Y} = \frac{\mathcal{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

onde:

σ_X é o desvio padrão de X;

σ_Y é o desvio padrão de Y;

μ_X é a média aritmética de X;

μ_Y é a média aritmética de Y;

$\mathcal{E}$ é o Valor esperado (MCAULIFFE, 2015).

O objetivo desta correlação era responder à pergunta: "Quanto mais padrões de acesso em uma fase, mais *jobs* fazem parte desta?". Para aplicar a correlação, utilizou-se da linguagem R e da biblioteca GGgaly[3] que implementa esta correlação pela função

[3]https://cran.r-project.org/web/packages/GGally/index.html

ggpairs[4]. Como resultado, o coeficiente da correlação foi de 0,37, ou seja, é uma correlação linear positiva, porém baixa. Isso responde nossa pergunta da seguinte forma: "sim, quanto mais padrões mais *jobs*, mas o crescimento não é igual para as duas variáveis", ou seja, pode ocorrer inúmeros *jobs* realizando um conjunto menor de padrões de acesso.

4.3 Conclusão do Capítulo

Neste capítulo, analisou-se 91.603 *jobs* que foram coletados pelo Darshan no supercomputador Intrepid no ano de 2012. Foram apresentadas duas análises sobre estes *jobs*, a primeira com o foco no comportamento individual de cada *job* e a segunda considerando o comportamento de todos os *jobs* sobre o sistema de E/S. Podemos identificar que as fases de E/S executam em mediana por 1,2 microssegundos.

Se considerado o tempo ocioso entre as fases como parte da fase, identificando assim a estabilidade de uma fase de E/S na aplicação, esse valor sobe para 1 segundo. Quando considerado o comportamento a nível do sistema uma fase de E/S contém mais que uma operação e a duração dela em mediana é de 4 microssegundos. No próximo capítulo são apresentadas as considerações finais deste trabalho.

[4]https://www.rdocumentation.org/packages/GGally/versions/1.5.0/topics/ggpairs

5 CONCLUSÃO E TRABALHOS FUTUROS

O comportamento E/S é um aspecto significativo do desempenho global das aplicações CAD. Além disso, a E/S é o fator limitador de desempenho para muitas aplicações paralelas. Assim, é crucial compreender a carga de trabalho de E/S de aplicações reais em plataformas CAD. Neste trabalho, propôs-se caracterizar o comportamento e a carga de trabalho das operações de E/S de aplicações paralelas. Foram utilizados dados de 91.603 *jobs* executados no supercomputador *Intrepid* no ano de 2012. A partir da análise dos dados, foram identificadas a duração das fases de E/S e os diferentes padrões de acesso.

Identificaram-se 22 padrões de acesso que se dividem em operações de escrita e leitura, com diferentes espacialidades e que utilizam as interfaces POSIX e MPI-IO, sendo que 97% das fases utilizam o POSIX. Analisando a distribuição das fases de E/S nas aplicações, concluíu-se que 50% dos *jobs* possuem 2 padrões de acesso durante sua execução e que o restante dos *jobs* apresentou no máximo 14 padrões de acesso. Além disso, estes padrões se repetiram até 251 vezes em 99% dos *jobs* e que esse comportamento ocorre 70% mais vezes em *jobs* que utilizam a interface MPI-IO.

Ao analisar a duração das fases de E/S concluíu-se que em 50% dos casos, a duração foi de 1,2ms. Além disso, as durações variaram entre 1us e 21h. Quando comparadas as leituras com as escritas, foi verificado que as fases de leitura duram até 10 vezes mais que as de escrita. Por fim, tem-se que as fases de escrita representam 81% de todas as fases de E/S das aplicações. Considerando a estabilidade dos padrões de acesso, a duração das fases que era de 1ms aumentou para 1s. Isso ocorre porquê aplicações paralelas em sua maioria utilizam de uma abordagem de *checkpoints* para suas operações de E/S.

Quando o comportamento de E/S à nível de sistema é analisado, as fases de E/S representam mais de 1 padrão de acesso devido à concorrência das aplicações. Assim, a duração mediana de uma fase é 4,7ms, sendo esse valor menor do que a mediana considerando a estabilidade dos padrões. Isso ocorre, pois se uma aplicação mudar o seu padrão de acesso, isso impactará na duração da fase do sistema. Em mediana, as fases contêm 4 padrões de acesso e 4 *jobs* simultâneos.

As análises apresentadas neste trabalho podem auxiliar pesquisadores da área de E/S paralela. Em particular, os que buscam saber quais são os padrões de acesso representativos, encontrados em aplicações reais de CAD. Além disso, neste trabalho descreveu-se uma metodologia que pode ser aplicada à outros dados de outras máquinas e a quantificação da estabilidade dos padrões de acesso, que não foi encontrada na literatura, ajuda na

elaboração de técnicas que se adaptam dinamicamente aos padrões de acesso, para saber qual deve ser a frequência de operação dessas técnicas e de detecção dos padrões.

5.1 Trabalhos Futuros

Os trabalhos futuros incluem a utilização desta abordagem em outros conjuntos de dados, acrescentando mais informações nas fases, como por exemplo, o tamanho da requisição e os dados transferidos. Além disso, a implantação de um benchmark sintetizador da carga de trabalho baseada em dados reais.

5.2 Publicações

Os seguintes trabalhos foram produzidos durante esta dissertação. A enumeração inicia-se por aqueles que estão fortemente relacionados com este trabalho:

- **Pablo J. Pavan**, Jean Luca Bez, Matheus S. Serpa, Francieli Z. Boito, Philippe O. A. Navaux: Caracterização do Comportamento de E/S Utilizando Aprendizado não Supervisionado. ERAD 2020.
- **Pablo J. Pavan**, Jean Luca Bez, Matheus S. Serpa, Francieli Z. Boito, Philippe O. A. Navaux: Uma abordagem de aprendizagem não supervisionada para a caracterização do comportamento I/O. SBAC-PAD 2019. (Qualis B1).
- **Pablo J. Pavan**, Jean Luca Bez, Matheus S. Serpa, Francieli Z. Boito, Philippe O. A. Navaux: Uma Caracterização da Carga de Trabalho I/O em Supercomputadores utilizando a Aprendizagem Não Supervisionada. WSPPD 2019.

Os seguintes artigos foram também publicados durante esta dissertação:

- Matheus S. Serpa, **Pablo J. Pavan**, Eduardo H. M. Cruz, Rodrigo L. Machado, Jairo Panetta, Antônio Azambuja, Alexandre S. Carissimi, Philippe O. A. Navaux: Energy Efficiency and Portability of Oil and Gas Simulations on Multicore and Graphics Processing Unit Architectures. Concurrency and Computation: Practice and Experience, 2021. (Qualis A2)
- Jean Luca Bez, Andre R. Carneiro, **Pablo J. Pavan**, Valéria S. Girelli, Francieli Z. Boito, Bruno A. Fagundes, Carla Osthoff, Pedro L. S. Dias, Jean-François Méhaut, Philippe O. A. Navaux: I/O performance of the Santos Dumont supercomputer.

International Journal of High Performance Computing Applications 2020. (Qualis B1).

- Matheus S. Serpa, **Pablo J. Pavan**, Jairo Panetta, Alexandre S. Carissimi, Philippe O. A. Navaux: Melhorando o Desempenho e a Eficiência Energética do Método Fletcher para Simulação de Extração de Petróleo. ERAD 2020.
- Gessica M. Azevedo, Jean Luca Bez, **Pablo J. Pavan**, Francieli Z. Boito, Philippe O. A. Navaux: Tamanhos de Requisições de E/S de Aplicações HPC em um Supercomputador. ERAD 2020.
- **Pablo J. Pavan**, Ricardo K. Lorenzoni, Vinícius Machado, Jean Luca Bez, Edson L. Padoin, Francieli Z. Boito, Philippe O. A. Navaux, Jean-François Méhaut: Energy efficiency and I/O performance of low-power architectures. Concurrency and Computation: Practice and Experience 2019. (Qualis A2)
- Edson L. Padoin, Andressa T. Diefenthaler, Matheus S. Serpa, **Pablo J. Pavan**, Emmanuell D. Carreño, Philippe O. A. Navaux, Jean-François Méhaut: Optimizing Water Cooling Applications on Shared Memory Systems. CARLA 2019.
- Matheus S. Serpa, **Pablo J. Pavan**, Jairo Panetta, Antônio Azambuja, Alexandre S. Carissimi, Philippe O. A. Navaux: Portabilidade e Eficiência do Método Fletcher de Aplicações Sísmicas em Arquiteturas Multicore e GPU. WSCAD 2019.
- **Pablo J. Pavan**, Matheus S. Serpa, Edson L. Padoin, Alexandre Carissimi, Philippe O. A. Navaux: Melhorando as Operações de E/S do Algoritmo RTM. ERAD 2019.
- **Pablo J. Pavan**, Matheus S. Serpa, Víctor Martínez, Edson L. Padoin, Philippe O. A. Navaux, Jairo Panetta: Strategies to Improve the Performance and Energy Efficiency of Stencil Computations for NVIDIA GPUs. WPerformance 2018.
- **Pablo J. Pavan**, Matheus S. Serpa, Emmanuell Diaz Carreño, Víctor Martínez, Edson L. Padoin, Philippe O. A. Navaux, Jairo Panetta, Jean-François Méhaut: Improving Performance and Energy Efficiency of Geophysics Applications on GPU Architectures. CARLA 2018 (Best Paper).
- Víctor Martínez, Matheus S. Serpa, **Pablo J. Pavan**, Edson L. Padoin, Philippe O. A. Navaux: Performance Evaluation of Stencil Computations Based on Source-to-Source Transformations. CARLA 2018.
- **Pablo J. Pavan**, Matheus S. Serpa, Edson L. Padoin, Lucas Mello Schnorr, Philippe O. A. Navaux, Jairo Panetta: Improving I/O Performance of RTM Algorithm for Oil and Gas Simulation. WSCAD 2018.

REFERÊNCIAS

ALFOROV, Y. et al. Towards green scientific data compression through high-level i/o interfaces. In: IEEE. **2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)**. [S.l.], 2018. p. 209–216.

BEHZAD, B. et al. Automatic generation of i/o kernels for hpc applications. In: IEEE. **2014 9th Parallel Data Storage Workshop**. [S.l.], 2014. p. 31–36.

BERGMAN, K. et al. Exascale computing study: Technology challenges in achieving exascale systems. **Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep**, v. 15, 2008.

BEZ, J. L. et al. Arbitration policies for on-demand user-level i/o forwarding on hpc platforms. In: **IEEE International Parallel & Distributed Processing Symposium (IPDPS 2021)**. [S.l.: s.n.], 2021.

BOITO, F. Z. et al. A checkpoint of research on parallel I/O for high-performance computing. **ACM Computing Surveys (CSUR)**, ACM, v. 51, n. 2, p. 23, 2018.

BOITO, F. Z. et al. Automatic I/O scheduling algorithm selection for parallel file systems. **Concurrency and Computation: Practice and Experience**, 2015. ISSN 1532-0634. Available from Internet: <http://dx.doi.org/10.1002/cpe.3606>.

BOITO, F. Z. et al. On server-side file access pattern matching. In: IEEE. **2019 International Conference on High Performance Computing & Simulation (HPCS)**. [S.l.], 2019. p. 217–224.

CARNEIRO, A. R. et al. Collective i/o performance on the santos dumont supercomputer. In: IEEE. **2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)**. [S.l.], 2018. p. 45–52.

CARNS, P. et al. Understanding and Improving Computational Science Storage Access Through Continuous Characterization. **Trans. Storage**, ACM, New York, NY, USA, v. 7, n. 3, p. 8:1–8:26, oct. 2011. ISSN 1553-3077.

CARNS, P. et al. **Performance analysis of Darshan 2.2. 3 on the Cray XE6 platform.** [S.l.], 2012.

CARNS, P. et al. 24/7 characterization of petascale i/o workloads. In: IEEE. **2009 IEEE International Conference on Cluster Computing and Workshops**. [S.l.], 2009. p. 1–10.

CARON, M. et al. Deep clustering for unsupervised learning of visual features. In: **Proceedings of the European Conference on Computer Vision**. [S.l.: s.n.], 2018. p. 132–149.

CHAMBERLAIN, S.; TAYLOR, I. L. **Using LD the GNU linker**. [S.l.]: Jeffrey Osier, 2010.

CHEN, B. M. et al. **Hard disk drive servo systems**. [S.l.]: Springer New York, 2002.

CORBETT, P. et al. Overview of the mpi-io parallel i/o interface. In: **Input/Output in Parallel and Distributed Computer Systems**. [S.l.]: Springer, 1996. p. 127–146.

CORNWELL, M. Anatomy of a solid-state drive. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 12, p. 59–63, 2012.

DIBBLE, P. C.; SCOTT, M. L. Beyond striping: The bridge multiprocessor file system. **ACM SIGARCH Computer Architecture News**, ACM New York, NY, USA, v. 17, n. 5, p. 32–39, 1989.

DORIER, M. et al. Calciom: Mitigating i/o interference in hpc systems through cross-application coordination. In: IEEE. **Parallel and Distributed Processing Symposium, 2014 IEEE 28th International**. [S.l.], 2014. p. 155–164.

FREEDMAN, D.; PISANI, R.; PURVES, R. Statistics (international student edition). **Pisani, R. Purves, 4th edn. WW Norton & Company, New York**, 2007.

HE, J. et al. I/O acceleration with pattern detection. In: 22ND INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE PARALLEL AND DISTRIBUTED COMPUTING, 2013, New York, New York, USA. **Proceedings...** [S.l.]: ACM, 2013. (HPDC '13), p. 25–36. ISBN 978-1-4503-1910-2.

HERBEIN, S. et al. Scalable i/o-aware job scheduling for burst buffer enabled hpc clusters. In: **Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing**. [S.l.: s.n.], 2016. p. 69–80.

JI, X. et al. Automatic, application-aware i/o forwarding resource allocation. In: **17th USENIX Conference on File and Storage Technologies (FAST 19)**. Boston, MA: USENIX Association, 2019. p. 265–279. ISBN 978-1-939133-09-0. Available from Internet: <https://www.usenix.org/conference/fast19/presentation/ji>.

JR, J. V. H. et al. Ppfs: A high performance portable parallel file system. In: CITESEER. **International Conference on Supercomputing**. [S.l.], 1995. p. 385–394.

KANG, Q. et al. Improving mpi collective i/o for high volume non-contiguous requests with intra-node aggregation. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 31, n. 11, p. 2682–2695, 2020.

KARRELS, E.; LUSK, E. Performance analysis of mpi programs. **Environments and Tools for Parallel Scientific Computing**, Citeseer, p. 195–200, 1994.

KASAVAJHALA, V. Solid state drive vs. hard disk drive price and performance study. **Proc. Dell Tech. White Paper**, p. 8–9, 2011.

KIM, Y.; GUNASEKARAN, R. Understanding i/o workload characteristics of a peta-scale storage system. **The Journal of Supercomputing**, Springer, v. 71, n. 3, p. 761–780, 2015.

KIM, Y. et al. Workload characterization of a leadership class storage cluster. In: **2010 5th Petascale Data Storage Workshop (PDSW '10)**. [S.l.: s.n.], 2010. p. 1–5. ISSN 2157-7250.

KUO, C.-S. et al. How file access patterns influence interference among cluster applications. In: **2014 IEEE International Conference on Cluster Computing (CLUSTER)**. [S.l.]: IEEE, 2014. p. 185–193. ISSN 1552-5244.

LANG, S. et al. I/o performance challenges at leadership scale. In: IEEE. **Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis**. [S.l.], 2009. p. 1–12.

LI, S. et al. A flattened metadata service for distributed file systems. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 29, n. 12, p. 2641–2657, 2018.

LIU, Y. et al. Server-side log data analytics for i/o workload characterization and coordination on large shared storage systems. In: IEEE. **SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis**. [S.l.], 2016. p. 819–829.

LOCKWOOD, G. K. et al. A year in the life of a parallel file system. In: IEEE. **SC18: International Conference for High Performance Computing, Networking, Storage and Analysis**. [S.l.], 2018. p. 931–943.

LOFSTEAD, J. et al. Six degrees of scientific data: Reading patterns for extreme scale science IO. In: 20TH INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 2011, San Jose, California, USA. **Proceedings...** [S.l.]: ACM, 2011. (HPDC '11), p. 49–60. ISBN 978-1-4503-0552-5.

LUU, H. et al. A multi-level approach for understanding i/o activity in hpc applications. In: IEEE. **2013 IEEE International Conference on Cluster Computing (CLUSTER)**. [S.l.], 2013. p. 1–5.

LUU, H. et al. A multiplatform study of I/O behavior on petascale supercomputers. In: ACM. **Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing**. [S.l.], 2015. p. 33–44.

MCAULIFFE, R. E. Expected value. In: ____. **Wiley Encyclopedia of Management**. American Cancer Society, 2015. p. 1–1. ISBN 9781118785317. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118785317.weom080154>.

MICROSYSTEMS, S. **LUSTRE file system - high-performance storage architecture and scalable cluster file system**. 2007.

NATALE, C. D.; MARTINELLI, E. Data analysis. In: **Breath Analysis**. [S.l.]: Elsevier, 2019. p. 81–94.

NISAR, A.; LIAO, W.-k.; CHOUDHARY, A. Scaling parallel i/o performance through i/o delegate and caching system. In: IEEE PRESS. **Proceedings of the 2008 ACM/IEEE conference on Supercomputing**. [S.l.], 2008. p. 9.

PARK, J. et al. Reducing solid-state drive read latency by optimizing read-retry. In: **Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems**. [S.l.: s.n.], 2021. p. 702–716.

PAVAN, P. J. et al. An unsupervised learning approach for i/o behavior characterization. In: IEEE. **2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)**. [S.l.], 2019. p. 33–40.

PAVAN, P. J. et al. Energy efficiency and i/o performance of low-power architectures. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 31, n. 18, p. e4948, 2019.

PEASE, D. et al. The linear tape file system. In: IEEE. **2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)**. [S.l.], 2010. p. 1–8.

POLLAK, B. Portable operating system interface (posix)-part 1x: real-time distributed systems communication application program interface (api). **IEEE Standard P**, v. 1003, 1996.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. **arXiv preprint**, 2015.

ROSS, R. et al. Implementing mpi-io atomic mode without file system support. In: IEEE. **CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.** [S.l.], 2005. v. 2, p. 1135–1142.

ROSS, R. B.; THAKUR, R. et al. Pvfs: A parallel file system for linux clusters. In: **Proceedings of the 4th annual Linux showcase and conference**. [S.l.: s.n.], 2000. p. 391–430.

SHAHZAD, F. et al. An evaluation of different i/o techniques for checkpoint/restart. In: IEEE. **2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum**. [S.l.], 2013. p. 1708–1716.

STARR, M. et al. Comparing tape and cloud storage for long-term data preservation. 2020.

SUGIHARA, K.; TATEBE, O. Design of locality-aware mpi-io for scalable shared file write performance. In: IEEE. **2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.], 2020. p. 1080–1089.

THOMASIAN, A. et al. A performance evaluation tool for raid disk arrays. In: IEEE. **First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.** [S.l.], 2004. p. 8–17.

TSUJITA, Y. et al. Improving collective MPI-IO using topology-aware stepwise data aggregation with I/O throttling. In: ACM. **Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region**. [S.l.], 2018. p. 12–23.

WANG, F. et al. **File system workload analysis for large scale scientific computing applications**. [S.l.], 2004.

XIE, B. et al. Characterizing output bottlenecks in a supercomputer. In: **High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for**. [S.l.: s.n.], 2012. p. 1–11.

XU, C. et al. **Dxt: Darshan extended tracing**. [S.l.], 2017.

YANG, J.; PEI, S.; YANG, Q. Warcip: write amplification reduction by clustering i/o pages. In: ACM. **Proceedings of the 12th ACM International Conference on Systems and Storage**. [S.l.], 2019. p. 155–166.

YANG, L. T.; GUO, M. **High-Performance Computing**. [S.l.]: Wiley Online Library, 2005.

YIN, Y. et al. Pattern-direct and layout-aware replication scheme for parallel I/O systems. In: 2013 IEEE 27TH INTERNATIONAL SYMPOSIUM ON PARALLEL DISTRIBUTED PROCESSING (IPDPS), 2013, Boston, MA. **Proceedings...** [S.l.]: IEEE, 2013. p. 345–356. ISSN 1530-2075.

YU, J. et al. On the load imbalance problem of I/O forwarding layer in HPC systems. In: **2017 3rd IEEE International Conference on Computer and Communications (ICCC)**. IEEE, 2017. v. 2018-Janua, p. 2424–2428. Available from Internet: <http://ieeexplore.ieee.org/document/8322970/>.

ZOLL, Q.; ZHU, Y.; FENG, D. A study of self-similarity in parallel I/O workloads. In: IEEE. **Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on**. [S.l.], 2010. p. 1–6.