UNIVERSIDADE DO VALE DO TAQUARI
CURSO DE ENGENHARIA DA COMPUTAÇÃO

COMPARATIVO DE DESEMPENHO DE UM *POOL* DE DISCOS CONECTADOS A UMA CONTROLADORA *RAID VERSUS* ZFS

Vinício Zanchettin

Lajeado, novembro de 2020

Vinício Zanchettin

COMPARATIVO DE DESEMPENHO DE UM *POOL* DE DISCOS CONECTADOS A UMA CONTROLADORA RAID *VERSUS* ZFS

Projeto de Monografia apresentado na disciplina de Trabalho de Conclusão de Curso II, do Curso de Engenharia da Computação, da Universidade do Vale do Taquari - Univates, como parte da exigência para a obtenção do título de BACHAREL EM ENGENHARIA DA COMPUTAÇÃO.

Orientador: Marcelo Euzébio Batista.

Lajeado, novembro de 2020

RESUMO

Este trabalho teve como objetivo verificar a viabilidade da utilização do sistema de arquivos ZFS (*Zettabyte File System*), para as demandas de armazenamento de dados. Atualmente, as aplicações estão crescendo quanto às suas demandas de espaço lógico. Focado neste tema, iniciou-se o estudo de uma solução alternativa para sua implementação em empresas de menor porte. A ideia principal do estudo foi testar a performance, a segurança e a praticidade entre as duas tecnologias: uma, totalmente baseada em *hardware* e já consagrada - com controladora de discos, bateria e memória *cache* - e outra tecnologia conhecida como ZFS, baseada em *software*. O sistema de arquivos ZFS foi escolhido para a execução do comparativo por ser muito simples e seguro para as implementações na grande maioria dos casos. Sendo assim, foram realizados alguns testes, cujos resultados foram considerados suficientes.

Palavras-chave: Arranjo de discos. Controladora RAID. Sistema de arquivos. ZFS.

ABSTRACT

This project aims to verify the feasibility of using the ZFS file system (Zettabyte File System), for data storage demands. Currently, applications are growing in terms of their logical space demands. Focused on that, the study of an alternative solution for the implementation in smaller companies begins. In the studies, the main idea will be to test the performance, safety and practicality between the two technologies. One totally based on hardware and already established, with disk controller, battery and cache memory and another technology known as ZFS and based on software. The ZFS file system was chosen to perform the comparison because it is very simple and secure for the implementations in the vast majority of cases. Therefore, some tests were started and the results were considered sufficient.

Keyword: Disk array, File System, RAID controller, ZFS.

LISTA DE FIGURAS

Figura 1 – Controladora RAID 13
Figura 2 – Exemplo de RAID 5 e *spare* 14
Figura 3 – Como o CoW trabalha em nível de blocos 16
Figura 4 – Exemplo de como o *snapshot* se reproduz 16
Figura 5 – Deduplicação de dados 17
Figura 6 – Sistema de gravação dinâmica 18
Figura 7 – Gráfico do sensor de temperaturas 28
Figura 8 – Apresentação dos discos com RAID hardware 30
Figura 9 – Apresentação dos discos com RAIDZ-1 do ZFS 30
Figura 10 – Pacotes básicos que foram instalados 31
Figura 11 – Comando de instalação do ZFS 31
Figura 12 – Comandos para a instalação do ZABBIX agente 32
Figura 13 – Comandos realizados para o MySQL 33
Figura 14 – Comandos utilizados para o Postgresql 33
Figura 15 – Exemplo de painel do *tmux* 34
Figura 16 – Lista dos principais comandos do *tmux* 35
Figura 17 – Comandos de importação e uso de disco 38
Figura 18 – Limpeza de *buffer* em memória 39

LISTA DE TABELAS

Tabela 1 – Alguns sistemas de arquivos mais utilizados 15
Tabela 2 – Trabalhos relacionados e recursos estudados 20
Tabela 3 – Descrição do servidor utilizado.....29
Tabela 4 – Custos aproximados de implementação de cada solução.....36
Tabela 5 – Resultados dos testes em bancos de dados 40
Tabela 6 – Consumo de CPU.....40
Tabela 7 – Consumo de memória 41
Tabela 8 – Latência de disco em ms.....42
Tabela 9 – Saída IOZone em IO/s em modo SYNC.....43
Tabela 10 – Saída IOZone em KB/s em modo SYNC.....43
Tabela 11 – Saída IOZone em IO/s, sem gravação assíncrona.....44
Tabela 12 – Saída IOZone em KB/s e sem gravação assíncrona.....44
Tabela 13 – Comando DD gravação e leitura com controladora.....45
Tabela 14 – Comando DD gravação e leitura com ZFS.....45

LISTA DE SÍMBOLOS E SIGLAS

CDDL	*Common Development and Distribution License*
CoW	*Copy on Write*
I/O	*input* e *output*
IP	*Internet protocol*
iSCSI	*Internet Small Computer System Interface*
KVM	*Kernel-based Virtual Machine*
LUN	*Logical Unit Numbers*
MBR	*Master Boot Record*
NAS	*Network Attached Storage*
PHP	*Personal Home Page*
RAID	*Redundant Array of Independent Disks*
RBD	*RADOS Block Devices*
SSD	*Solid State Disk*
VM	*Virtual Machine*
ZFS	*Zettabyte File System*

SUMÁRIO

1 INTRODUÇÃO 9
1.1 Problema de pesquisa 10
1.2 Objetivos 10
1.3 Estrutura 11

2 FUNDAMENTAÇÃO TEÓRICA 12
2.1 Controladora de discos 13
2.1.1 RAID 13
2.1.2 RAID nível 5 14
2.2 Sistemas de arquivos 15
2.2.1 CoW 15
***2.2.2 Snapshot* 16**
2.2.3 Deduplicação de dados 17
2.2.4 RAIDZ1 17

3 TRABALHOS RELACIONADOS 19

4 MATERIAIS E MÉTODOS 22
4.1 Procedimentos metodológicos 23
4.1.1 A Pesquisa quanto aos modelos científicos 23
4.1.2 A Pesquisa quanto ao modo de abordagem 24
4.1.3 A Pesquisa quanto aos fins 24
4.1.4 A Pesquisa quanto aos procedimentos técnicos 25
4.2 Tecnologias 25
4.2.1 ZFS 25
***4.2.2 Zabbix* 26**
4.2.3 Linux *Debian* 26
4.2.4 IOZONE 26
4.2.5 DD 27
4.3 Desenvolvimento 27
4.3.1 Informações do ambiente 27
4.3.2 Informação descritiva do servidor 29
4.3.3 Sistema operacional 29
4.3.4 Pacotes instalados no ambiente 31

4.3.5 Instalação e configuração do ZABBIX 32
4.3.6 Ajustes de banco de dados MySQL e PostgreSQL 32
4.3.7 *Dashboard* no ambiente SHELL com tmux 33
4.4 Estimativa de custos 35

5 TESTE E ANÁLISE DOS RESULTADOS 37
5.1 Testes executados 37
5.1.1 Importação de banco de dados 37
5.1.2 *Benchmark* IOZone 38
5.1.3 Geração de arquivos com o DD 39
5.2 Resultados 39
5.2.1 Importação dos bancos de dados 40
5.2.2 Execução do *Benchmark* IOZone 42
5.2.3 Gravação e leitura serial com DD 44

6 CONCLUSÃO 46

REFERÊNCIAS 48

ANEXO A – Tela do Linux Ubuntu 20.04 da versão experimental do ZFS 51
ANEXO B – Comparativos de utilização de espaço em disco com a deduplicação ativada no Bacula Backup 52

1 INTRODUÇÃO

Há poucos anos, as empresas falavam em *megabytes*, depois em *gigabytes*, agora em *terabytes* e assim por diante. A cada dia, a necessidade de armazenar e de buscar informações é maior e, na mesma medida, encontrar soluções flexíveis para resolver esse problema, que, em alguns casos, não está ao alcance financeiro de empresas de menor porte.

Soluções consideradas mais robustas são as baseadas em *hardware,* as quais, no caso, podem ser *storages*, quando um servidor único utiliza as controladoras RAID (*Redundant Array of Independent Disks*). Essas tecnologias contam com muitos recursos que garantem a performance e a integridade dos dados que ali estão armazenados.

Existem muitas soluções de *software* que prometem resolver esta questão e deixar o administrador de infraestrutura tranquilo. Neste momento, entra o ZFS (*Zettabyte file system*), que promete, segundo sua documentação, entregar vários recursos como soluções profissionais, entre elas, a redundância de disco, a checagem de integridade em tempo real, *cache* em níveis um e dois, *snapshot* de volumes inteiros, *spares,* entre muitos outros. A ideia do ZFS é que o planejamento e a manutenção dos sistemas que ali estão hospedados sejam simples.

Não se entende que o ZFS vá proteger os dados e eliminar um sistema de *backup* eficiente; ele apenas vai mantê-los tão seguros quanto a maioria das soluções de *storages* de mercado baseadas em *hardware* (WOJSŁAW, 2017).

O presente trabalho explorou as vantagens e as desvantagens de um sistema de arquivos estruturado em duas soluções distintas: uma baseada em *hardware* e outra, em *software*. Os critérios considerados no comparativo entre as duas soluções

foram funcionalidades, testes de performance, testes de resistência a falhas e facilidade de manuseio.

1.1 Problema de pesquisa

Este estudo partiu da ideia de que se pode reduzir o custo e manter a segurança e a performance de um sistema de arquivos, mesmo quando se faz uso de *software* para tal finalidade. Foram comparados os recursos de uma controladora RAID (PERC H330) de porte médio, com o sistema de arquivos ZFS (*Zettabyte file system*), bem como foram coletadas informações para a implementação de um *pool* de discos, baseada totalmente em *software*.

Sendo assim, a questão de pesquisa foi: “A procura de uma solução de armazenamento baseada em *software* e se ela pode ser utilizada para fins profissionais”. Com base neste problema, buscou-se identificar e descobrir em quais situações um sistema de arquivos realmente necessita ser gerenciado por uma controladora de discos e quando não se faz necessário, sendo possível reduzir drasticamente os investimentos em *hardware* e entregar uma solução completa do início ao fim.

1.2 Objetivos

O objetivo geral da pesquisa foi encontrar soluções de menor custo, com a mesma eficiência para o armazenamento de grandes volumetrias de dados, na tentativa de provar que o *hardware* nem sempre é a melhor opção, quando o cenário exige custo-benefício.

Para atingir os objetivos deste estudo, foram utilizados vários recursos literários, a maioria no idioma inglês; a aplicação de ferramentas que medem a performance dos cenários de testes; a utilização de *hardware* que mantém a disputa entre as tecnologias equivalentes; a comparação entre as informações obtidas.

1.3 Estrutura

Este estudo está dividido em cinco capítulos. No primeiro, apresenta-se o tema do projeto, o problema de pesquisa e os objetivos do trabalho. O segundo capítulo apresenta a fundamentação teórica com uma revisão sobre as tecnologias e embasamento teórico para o desenvolvimento do estudo.

Na sequência, no terceiro capítulo, são mencionados alguns trabalhos relacionados ao tema do estudo e como eles refletem nesta pesquisa. No quarto capítulo, são descritos os materiais e métodos utilizados, abordando os fins da pesquisa, tecnologias e o desenvolvimento. O quinto capítulo documenta os resultados dos testes e ambientes de reproduções utilizados para o estudo das tecnologias. No último capítulo, abordam-se a conclusão e orientações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordadas duas formas de montar um *pool* de discos. A primeira, utilizando uma controladora RAID (*Redundant Array of inexpensive ou Independent*) e a outra, o sistema de arquivos ZFS (*Zettabyte file system*). Ambas as soluções são montadas com um agrupamento de discos compatíveis com RAID nível cinco, que é mais voltado para quem precisa de espaço e maiores performances de leituras.

Segundo Siqueira (2009), o RAID via *hardware* é apresentado ao sistema operacional de forma transparente, sem exigir recursos como processamento e memória da placa-mãe. No caso de ser via *software,* como o ZFS, o sistema operacional toma conta das operações, fazendo-se necessário o uso dos recursos compartilhados.

As exponenciais melhoras dos processadores e a facilitação de acesso à grande quantidade de memória nos computadores atuais não vêm acontecendo na mesma velocidade para os discos rígidos, o que gera muita latência quando se necessita acessar algo que esteja no disco (TANENBAUM; WOODHULL, 2008). As duas formas de montar o *pool* de discos citadas acima possuem recursos e tecnologias como o *cache* em memória física. O *cache* é o armazenamento de dados recém-acessados em uma área de memória física, que oferece acesso mais rápido e evita releituras no disco rígido e latências.

2.1 Controladora de discos

A maioria dos servidores possui uma controladora de discos que tem as principais funcionalidades de armazenamento, entre elas, fazer *cache*, montar e gerenciar o agrupamento de discos, além de uma bateria para manter a consistência dos dados, caso aconteça um desligamento de forma brusca.

Figura 1 – Controladora RAID

Fonte: Adaptado pelo autor do Google Imagens (2020).

Na Figura 1, podemos observar uma controladora de disco de modelo PERC H330, que possui a memória *cache* e, no canto inferior direito, um engate rápido para a bateria.

2.1.1 RAID

É uma tecnologia que agrupa vários discos dentro de uma ou mais unidades virtuais. Foi proposta em 1988, por David A. Patterson, Garth A. Gibson e Randy H. Katz, na publicação "*A Case for Redundant Arrays of Inexpensive Disks* (RAID)", na conferência SIGMOD de 1988: p. 109–16. A ideia de fazer estes arranjos de discos é

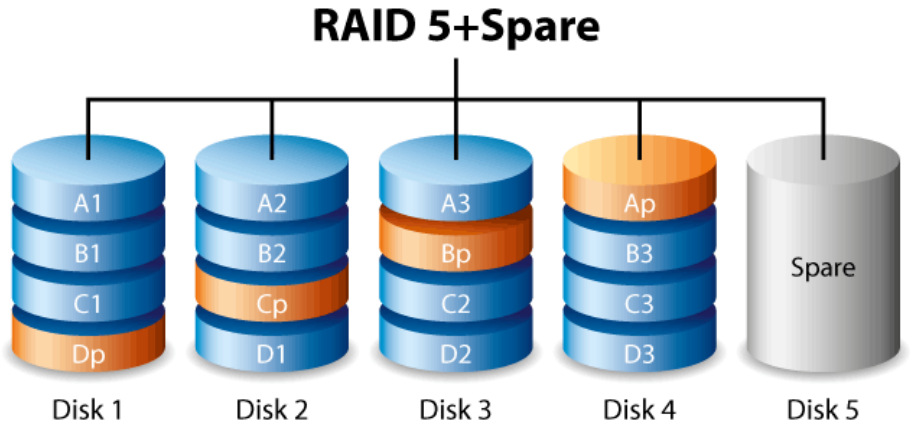
a busca de maior segurança, performance e escalabilidade, distribuindo os dados entre os discos. O sistema operacional recebe apenas um volume virtual que grava as informações. O gerenciador de sistema RAID, que pode ser *software* ou *hardware*, executa de forma transparente a gravação das informações, liberando o volume para executar mais processos de I/O (*input* e *output*).

Segundo Ferreira e Matayoshi (2016), a ideia inicial de Patterson na implementação do sistema RAID era que seria somente baseada em soluções de *hardware;* no entanto, em outra publicação, ele observou que a solução também poderia ser facilmente implementada com *software*.

2.1.2 RAID nível 5

No nível de RAID 5, utiliza-se a ideia de *striping* de dados, isto é, os dados são espalhados entre os discos, sendo uma parte a paridade. O aproveitamento de espaço neste nível de RAID é N-1, sendo N, número de discos. Ele também pode ser utilizado com o *global hot spare*, que garante um pouco mais o *pool* de discos, caso haja alguma falha. Na Figura 2, é possível visualizar o esquema dos discos.

Figura 2 – Exemplo de RAID 5 e *spare*



Fonte: Seagate (2020, texto digital).

O número mínimo de discos é três, podendo ser adicionados quantos a controladora comportar. Esse método é muito utilizado em sistemas de armazenamento, pois fornece garantia caso aconteça alguma falha de disco, além de aumentar significativamente a performance de leitura (MONTEIRO, 2005).

2.2 Sistemas de arquivos

Os sistemas de arquivos apresentam-se em constante evolução. Um disco pode conter até quatro partições primárias, sendo o setor zero reservado para a MBR (*Master Boot Record*), cuja principal função é manter os endereços de início e fim das partições (TANENBAUM; WOODHULL, 2008). Nas partições citadas acima, encontra-se o sistema de arquivos. Na Tabela 1, podemos observar alguns tipos de sistemas de arquivos.

Tabela 1 – Alguns sistemas de arquivos mais utilizados

Sistemas de arquivos	
Sigla	**Sistema operacional**
FAT32 (*File Allocation Table*)	Microsoft
NTFS (*New Technology File System*)	Microsoft
JFS (*Journaled File System*)	Linux
XFS	Linux
RaiserFS	Linux
BTRFS (*B-tree file system*)	Linux
EXT4 (*Fourth Extended Filesystem*)	Linux
EXT3 (*Third Extended Filesystem*)	Linux
EXT2 (*Second Extended File System*)	Linux

Fonte: Adaptado pelo autor de Wikepedia (2020).

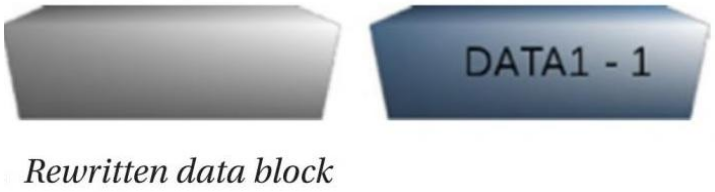
Com o passar dos anos, algumas tecnologias de sistemas de arquivos foram evoluindo, tornando-se mais eficientes e seguras. É neste ponto que entra o ZFS, com muitos recursos, com potencial de entregar muito mais do que apenas um lugar para o sistema operacional gravar as suas informações e dados de usuários.

2.2.1 CoW

O sistema de gravação que o ZFS possui é o CoW (*Copy on Write*), que oferece muitas das características do sistema de arquivos (WOJSŁAW, 2017). Na maioria dos

sistemas de arquivos, utiliza-se a técnica conhecida como *journaling*, uma pequena partição que funciona como um LOG. Qualquer arquivo que for alterado é, primeiramente, gravado numa área do disco e, somente após o sucesso da gravação, é atualizado no sistema de arquivos oficial, o que garante que arquivos não corrompam em caso de queda de luz ou de remoção inadequada do disco. Já no sistema cópia em gravação, é efetuada uma duplicação total do arquivo a ser modificado para outra área de disco, mantendo intacta a versão antiga. Caso algum problema ocorra durante a mudança ou a gravação do arquivo novo, o antigo ainda está preservado. Se a gravação ou a alteração for concluída com sucesso, a área antiga em que se encontra o arquivo na sua versão anterior é marcada como liberada para regravação. A Figura 3 demonstra um exemplo do CoW; o primeiro bloco é o que foi marcado para uma futura reescrita.

Figura 3 – Como o CoW trabalha em nível de blocos



Fonte: Wojsław (2017, p. 3).

2.2.2 Snapshot

O *snapshot* é uma forma de clonar ou fazer *backup* de uma partição inteira de um sistema de arquivos que está rodando a quente. Como o sistema ZFS trabalha com cópia em gravação, possibilita gerar uma imagem dos blocos atuais e armazená-los em outra unidade de disco, podendo assim restaurar ao ponto da montagem anterior ou até mesmo migrar para outra unidade de armazenamento externa, sem a perda de dados. Na Figura 4, pode-se observar como o *snapshot* em nível de bloco se reproduz.

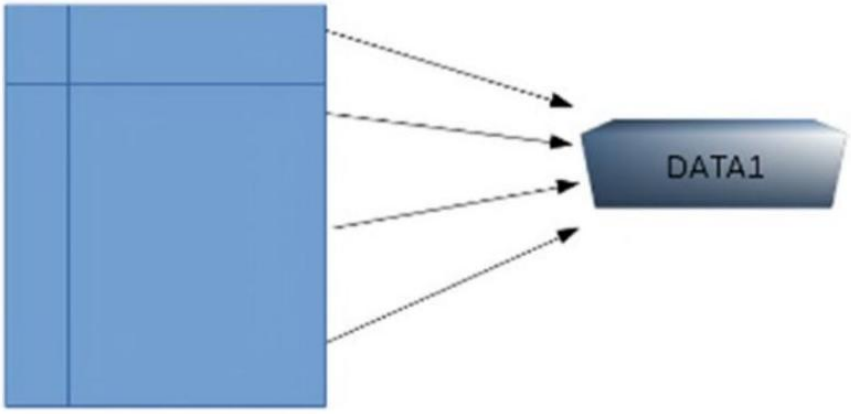
Figura 4 – Exemplo de como o *snapshot* se reproduz

Fonte: Wojsław (2017, p. 4).

2.2.3 Deduplicação de dados

A deduplicação de dados é um recurso muito utilizado para eliminar o uso desnecessário de blocos no sistema de arquivos, fazendo com que dados duplicados não ocupem duas vezes o espaço no disco.

Figura 5 – Deduplicação de dados



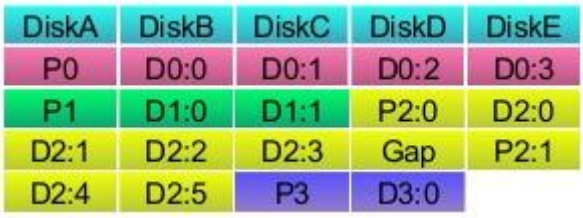
Fonte: Wojsław (2017, p. 4).

Como se pode observar na Figura 5, o sistema de arquivos ZFS faz comparações binárias entre os blocos de arquivos e, quando encontra duplicações, gera um novo, cadastra-o na tabela de deduplicação, faz referência apenas ao endereço dos outros blocos e marca os duplicados como regraváveis (WOJSŁAW, 2017).

2.2.4 RAIDZ1

A ideia do RAID-Z1 no sistema de arquivos ZFS é semelhante ao do RAID 5 (ORACLE, 2013), quando utilizado com uma controladora *hardware* ou com o *software* MDAdm. Uma das poucas diferenças é que, nesse sistema de redundância, não é necessário os discos serem do mesmo tamanho, o que possibilita a gravação dinâmica dos blocos no *pool* de discos.

Figura 6 – Sistema de gravação dinâmica



Fonte: Adaptado pelo autor do Google Imagens (2020).

Na Figura 6, podemos observar que o ZFS, diferente do RAID 5 convencional, possui um sistema de gravação dinâmico e não estático no disco.

3 TRABALHOS RELACIONADOS

O trabalho de Gurjar e Kumbhar (2019) faz um comparativo entre dois sistemas de arquivos, ZFS e BTRFS (*B-tree file system*). Ambos são manutenidos pela ORACLE e prometem várias possibilidades ao serem implementados. A pesquisa evidenciou que os dois sistemas de arquivos são de fácil manuseio e que o BTRFS atingiu uma taxa mais alta de transferência.

O estudo realizado por Eko, Agung e Ahmad (2016) apresenta como ideia base a implementação de um agrupamento de discos, ou seja, um *storage* sem que seja necessária a utilização de uma controladora RAID. São efetuados vários testes com o sistema de arquivos, para tentar garantir a performance e a segurança dos dados armazenados. Um ponto interessante é que foi detectada uma utilização maior de processador em caso de habilitação da compactação GZIP, tornando o sistema de arquivos mais lento.

No estudo de Zhang *et al.* (2013), testaram-se algoritmos de checagem e de correção de erros do sistema ZFS. No artigo, são pontuadas as evoluções e a grande preocupação das comunidades científicas em relação à integridade dos dados gravados em sistemas de arquivos da proporção *zettabyte*. A conclusão foi que o sistema de arquivos utiliza algoritmos diferentes para a checagem e de forma randômica.

Nos testes efetuados em "*Ext4, XFS, BtrFS and ZFS Linux File Systems on RADOS Block Devices* (RBD)", os autores buscaram uma forma de distribuir um sistema de arquivos em vários *storages*, com o objetivo de superar a limitação física de disco que cada computador consegue suportar. A ideia foi distribuir os dados entre os servidores, utilizando o RADOS (*Ceph Project*), em nível de bloco, buscando uma

melhor performance. Efetuaram testes parecidos com os que foram pensados para este trabalho, como também realizaram o uso do *benchmark* IOZONE.

Na pesquisa de Phronchana, Nupairoj e Pirornsopa (2011), são efetuados testes de comparativos entre o sistema de arquivos ZFS versus LVM (*with ext4*). O principal objetivo foi avaliar qual dos sistemas de arquivos teria a melhor performance, rodando sobre protocolo iSCSI, que é baseado em pacotes IP. Os autores observaram que o ZFS aumenta significativamente a sua performance, quando o *pool* aumenta a quantidade de disco físicos. Concluíram que isso acontece, porque o ZFS tem uma característica de gravação randômica, podendo gravar dados em qualquer disco, a qualquer momento.

Tabela 2 – Trabalhos relacionados e recursos estudados

Pesquisa	**RAID *hardware versus* RAIDZ1**	**ZFS**	**Performance**
A Review on Performance Analysis of ZFS & BTRFS	Não	Sim	Sim
On The Implementation of ZFS (Zettabyte File System) Storage System	Não	Sim	Sim
Zettabyte Reliability with Flexible End-to-end Data Integrity	Não	Sim	Sim
Ext4, XFS, BtrFS and ZFS Linux File Systems on RADOS Block Devices (RBD)	Não	Sim	Sim
Performance Evaluation of ZFS and L VM (with ext4) for Scalable Storage System	Não	Sim	Sim
Este trabalho	Sim	Sim	Sim

Fonte: Do autor (2020).

O presente estudo tem como objetivo a comparação de um *pool* de discos montados no esquema compatível com RAID 5, nas duas tecnologias. Nos dias atuais, quando se procura segurança e velocidade, a orientação é a aquisição de uma controladora RAID baseada em *hardware*, mas, como o sistema de arquivos ZFS promete em sua documentação entregar alta performance aliada à segurança, acredita-se que pode ser uma ótima opção de custo-benefício para os agrupamentos

de discos em ambientes corporativos, podendo assim competir com soluções baseadas em *hardware*.

4 MATERIAIS E MÉTODOS

Com o passar dos anos, observa-se que as soluções baseadas em *software* estão substituindo as fundamentadas em *hardware*. Um bom exemplo disso é o sistema *softraid* MDAdm (Linux), em cuja implementação, há poucos anos, somente alguns entusiastas sentiam confiança. Hoje em dia, é a solução para muitas instalações de menor custo, mantendo a segurança dos sistemas de arquivos.

Na busca de um sistema de arquivos para gerenciar grande armazenamento de dados, podemos encontrar o sistema de arquivos ZFS, ao qual se dispensa um olhar diferenciado quando se fala em performance, segurança e grande quantidade de dados. O ZFS permite montar um *pool* de discos, equivalente ao de uma controladora RAID baseada em *hardware*. Sendo assim, o ZFS apresenta várias características que foram estudadas e comparadas às de uma controladora *hardware*.

Este capítulo apresenta a forma escolhida pelo autor para a investigação e o desenvolvimento do estudo. Nesta seção, estão definidos os métodos científicos, fins da pesquisa e procedimentos técnicos. Além deles, são apresentadas as tecnologias utilizadas para a implantação dos sistemas de arquivos propostos e a realização dos testes. O capítulo também expõe uma estimativa de custos para a implementação de cada sistema, mostrando um comparativo entre ambos.

4.1 Procedimentos metodológicos

Segundo Gil (2008), o experimento é a forma que melhor descreve uma pesquisa científica. O autor também recomenda que se deve moldar processos de controle, para não haver alterações nos resultados finais da pesquisa.

4.1.1 A Pesquisa quanto aos modelos científicos

O método aplicado nesta pesquisa foi o dedutivo e o dialético, visto que a pesquisa se baseia no estudo de *softwares* e de *hardwares* já existentes, sendo as premissas comprovadas como reais. Segundo Gil (2008), o método dedutivo começa de um conhecimento já comprovado ou pré-estabelecido que vai na direção de uma opinião própria. A essência do método dedutivo é a sua aplicação nas ciências, nos estudos científicos (GIL, 2008). Nesse sentido, no caso deste estudo, busca-se comparar uma tecnologia de armazenamento baseada em *hardware versus software*. Sendo assim, a pesquisa é diretamente ligada a um projeto científico. Neste trabalho, os resultados reais foram analisados, comparados e apresentados em forma de planilha, em busca de um novo ponto de vista ou de comprovação de que eles eram reais. A partir das premissas e testes já efetuados, tentamos capturar uma nova ideia decorrente de outras análises já testadas em outros eventos (PRODANOV; FREITAS, 2013).

O método dialético consiste em analisar estudos anteriores como uma forma de leitura e de comparação de pontos de vista. Segundo Prodanov e Freitas (2013), o interesse do método é a interpretação da realidade, partindo da ideia de que todos os acontecimentos apresentam características organicamente contraditórias e indissolúveis.

No mundo científico, não é interessante começar todas as pesquisas do zero. Quando o assunto já está comprovado, cabe aos próximos tentar corroborar novas formas ou probabilidades. Também se pode reavaliá-las, pois a evolução da tecnologia ocorre de forma exponencial; por isso, em pouco tempo, tudo pode mudar. Assim, reavaliando os estudos, conseguimos chegar a novas conclusões.

4.1.2 A Pesquisa quanto ao modo de abordagem

O presente trabalho, cuja ideia é buscar a maior exatidão possível, tem como abordagem o método qualitativo, que permite a observação por vários ângulos. Os pesquisadores que utilizam a abordagem qualitativa opõem-se ao pressuposto que descreve um modelo único de pesquisa para todas as ciências (GERHARDT; SILVEIRA, 2009). Nas pesquisas e estudos qualitativos, a comprovação da ideia, segundo o entendimento de alguns autores, não é obrigatória. Contudo, uma ideia de pesquisa pode orientar a estrutura do projeto de pesquisa (PRODANOV; FREITAS, 2013).

Como o propósito da pesquisa é obter resultados reais, e não comparações matemáticas, optou-se pela abordagem qualitativa. A amostragem randômica simples foi o procedimento base da coleta de dados científicos. Pode-se afirmar, com base em Prodanov e Freitas (2013), que todos os outros procedimentos utilizados para compor dados e amostras são variações do método qualitativo.

4.1.3 A Pesquisa quanto aos fins

Quanto aos fins da pesquisa, implementou-se o modelo *estudo de caso*, visando a executar sobrecargas de leitura e gravação, desligamentos bruscos e remoção de discos das unidades de armazenamento, para coletar dados por amostragens e, em seguida, analisar o desempenho de um sistema de arquivos montado sobre um *pool* de discos de uma controladora RAID e, posteriormente, sobre um *pool* de discos ZFS. Na pesquisa descritiva, o pesquisador apenas documenta e descreve as informações capturadas, sem interferir sobre elas (PRODANOV; FREITAS, 2013). Alguns estudos descritivos podem aproximar-se de uma pesquisa explicativa, pois tentam interpretar a natureza ou o resultado de alguma pesquisa (GIL, 2002). Também podemos compreender que uma pesquisa explicativa pode ser a continuação de uma pesquisa descritiva. Nesse caso, os dados são extraídos e mantidos em seu estado original, sem a necessidade de manipulação ou de mudança por parte do pesquisador.

4.1.4 A Pesquisa quanto aos procedimentos técnicos

Nesta pesquisa, utilizamos o *estudo de caso* como forma de investigação, pois efetuou-se a coleta de dados por amostragem, que foram analisados posteriormente. Segundo Gil (2008), o estudo de caso exaustivo e aprofundado pode ser definido pela escolha de um ou de poucos elementos. Quando utilizamos a técnica de estudo de caso, não há como garantir que exista apenas uma maneira de obter as informações, mas, sim, várias formas diversas e abrangentes (PRODANOV; FREITAS, 2013).

4.2 Tecnologias

Nesta seção, são apresentadas as principais tecnologias para a execução dos testes de comparação das duas tecnologias de armazenamento de dados.

4.2.1 ZFS

O ZFS é um sistema de arquivos de código aberto, lançado junto ao sistema operacional *Solaris*, em novembro de 2005, pela *Sun Microsystems*, atualmente mantido e de propriedade da ORACLE, sob o licenciamento "*Common Development and Distribution License* (CDDL)". Se comparado a outros sistemas de arquivos, o ZFS se destaca, pois ele integra muitas funcionalidades sem a necessidade de *plugins* de terceiros. Entre elas, podemos citar: a RAID, *cache* de escrita e de leitura em nível dois (com discos SSDs e ARC2), *cache* em nível um (na memória física do servidor), automontagem e subdivisão de partições, CoW (*Copy on Write*), *snapshots*, compressão e deduplicação de dados em tempo real, entre muitos outros recursos.

O ZFS pode ser administrado pela linha de comandos do Linux, facilmente implementado. Já há distribuições Linux que entregam suporte nativo ao ZFS. Há uma grande discussão em relação à padronização ou não do ZFS nas distribuições Linux, pois, em sua versão original, está sob a guarda da ORACLE, o que deixa lacunas em relação a se algum dia será cobrado de forma comercial.

4.2.2 Zabbix

A *Zabbix* é uma ferramenta *open source*, fundada por Alexei Vladishev, cuja essência é o monitoramento de equipamentos de TI. Com o *Zabbix*, podemos monitorar em tempo real qualquer equipamento e gerar gráficos para uma consulta posterior. Ele roda no sistema operacional Linux e é compatível com vários sistemas de banco de dados. A visualização e as configurações de ambiente se dão numa interface WEB e a programação em PHP (*Personal Home Page*). O código fonte das bibliotecas é escrito na linguagem de programação C.

4.2.3 Linux *Debian*

Debian é um sistema operacional baseado no *kernel* Linux, cuja primeira versão foi lançada em 1993, por Ian Murdock. Seu crescimento foi muito vagaroso até meados de 1999, quando seu crescimento foi acelerado. Mantida pelo projeto *Debian*, atualmente está na versão 10, com o nome de versão *Debian Burst*. Ele é utilizado pelo sistema de virtualização *Proxmox* como sistema operacional base, tendo uma instalação mínima que consegue carregar utilizando poucos recursos de *hardware*.

4.2.4 IOZONE

O IOZone é um *software* compilado em linguagem C, muito utilizado para fazer testes de performance em sistemas de arquivos, podendo ser portado para vários sistemas operacionais (AIX, BSDI, HP-UX, IRIX, FreeBSD, Linux, OpenBSD, NetBSD, OSFV3, OSFV4, OSFV5, SCO OpenServer, Solaris, MAC OS X, Windows (95/98/Me/NT/2K/XP)), quando baixada a sua versão binária. Com o IOZone, podemos adquirir as seguintes métricas de um sistema de arquivos: *Read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random read, pread ,mmap, aio_read, aio_write*.

As características do IOZone são várias, entre elas, pode-se citar: *ANSII C source, POSIX async I/O, mmap() file I/O, normal file I/O, single stream measurement,*

multiple stream measurement, distributed file server measurements (Cluster), POSIX pthreads, multi-process measurement, excel importable output for graph generation, latency plots 64 bit compatible source, large file compatible, stonewalling in throughput tests to eliminate straggler effects, processor cache size configurable, selectable measurements with fsync, O_SYNC.

Quando executado em um ambiente *shell*, o IOZone possui vários parâmetros que definem se o teste a ser efetuado é de uma ou mais métricas e de blocos de tamanhos variados. Também há a possibilidade de jogar a saída do *benchmark* para uma planilha de *excel* e, posteriormente, passar à geração de gráficos.

4.2.5 DD

O *software* DD é um utilitário muito antigo, da linha de comando do *Linux,* que é frequentemente utilizado para clonar partições ou volumes inteiros, gerar imagens de discos ou cds, além de gravar zeros ou dados randômicos no disco, destruindo qualquer vestígio de informações que ali pode conter. Neste caso, ele foi utilizado para efetuar testes de performance num sistema de arquivos.

Para o teste de gravação, foi utilizado o seguinte comando "*dd if=/dev/zero bs=1024 count=10000000 of=10GB_file_to_write*" e a leitura "*dd if=10GB_file_to_write of=/dev/null bs=1024*".

4.3 Desenvolvimento

Este capítulo do desenvolvimento traz as informações dos *hardwares* utilizados, a montagem e o monitoramento do ambiente.

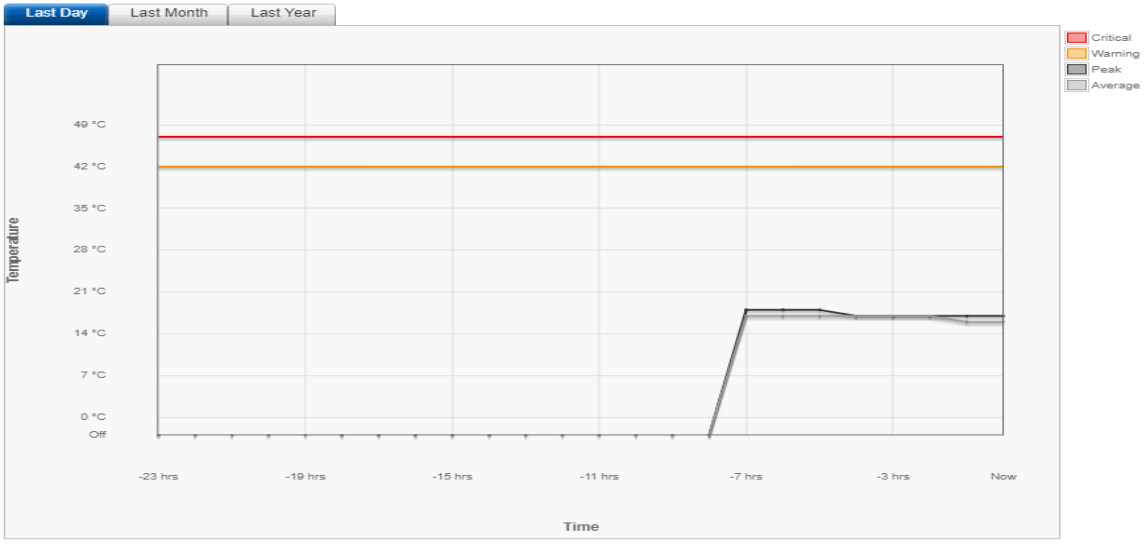
4.3.1 Informações do ambiente

Segundo Faccioni Filho (2016), uma sala de servidores ou ambiente de *datacenter* é formado por vários subsistemas, para garantir o melhor funcionamento

dos equipamentos que ali estão instalados. No nosso caso, conseguimos manter controlados os seguintes critérios: refrigeração, energia elétrica, conectividade de rede e controle de acesso à sala de TI.

O local onde os testes foram efetuados é uma sala de 3 metros de largura por 6 metros de comprimento. A climatização foi mantida nos 23 ºC, com um sistema de ar-condicionado de 12.000 BTUs. Acredita-se que a temperatura ficou estável, pois os sensores de temperatura do servidor utilizado nos testes não passaram de 22 ºC.

Figura 7 – Gráfico do sensor de temperaturas



Fonte: Adaptado pelo autor de iDrac da Dell (2020).

No gráfico da Figura 7, podemos observar que a temperatura sempre se manteve num valor aceitável, baseada na Norma NBR 14565 (FACCIONI FILHO, 2016), que é entre 17ºC e 27ºC.

O *nobreak* utilizado para manter a rede estabilizada possui um banco de baterias com capacidade de manter todos os equipamentos ligados por 1 hora, caso ocorra alguma falha na rede elétrica externa. A alimentação externa de energia é entregue pela concessionária e passa por um gerador de energia que leva entre 30 a 45 segundos até estabilizar-se e assumir a entrega total de energia elétrica da sala de TI.

A conectividade da rede é equipada com *switches* de camada dois e *gigabit.* A parte de cabeamento com estrutura nos padrões é certificada, para garantir que a rede de dados não interferisse nos testes.

Já a sala de TI possui como restrição de acesso uma porta chaveada, a cujas cópias das chaves apenas pessoas autorizadas têm acesso. Também conta com um sistema de CFTV e sensores de alarme, para garantir que apenas pessoas autorizadas acessem a sala.

4.3.2 Informação descritiva do servidor

O servidor utilizado foi um equipamento da marca DELL, conforme as informações na Tabela 3.

Tabela 3 – Descrição do servidor utilizado

Configuração do servidor utilizado		
Peça	**Detalhes**	**Firmware**
Processador	Intel(R) Xeon(R) CPU E5-2420 v2 @ 2.20GHz	
Memória	16 GB DDR 1600MHz	
Controladora RAID OFFBOARD	PERC H710 Adapter	21.3.4-0001
Controladora de discos ONBOARD		
HD	4 HD 500 GB ST500DM002 1BD1	KC45

Fonte: Do autor (2020).

Pelo iDRAC (DELL, 2020), que fornece informações sobre a saúde do servidor, conseguimos constatar que o *hardware* apresentava-se em plenas condições de uso para os testes. Nele, capturamos informações de CPU, memória, temperaturas, controladora RAID e discos.

4.3.3 Sistema operacional

Para os testes, foi utilizado o sistema operacional Linux Debian 10. A instalação foi executada em um *pendrive* de 16GB e executada diretamente por esta mídia. A

ideia era não haver interferências do sistema operacional ou de outros processos, como o sistema de banco de dados, que poderia interferir nos testes.

Figura 8 – Apresentação dos discos com RAID hardware

```
root@linux0196:~# lsblk -f
NAME         FSTYPE      LABEL UUID                                   FSAVAIL FSUSE% MOUNTPOINT
sda
sdb
├─sdb1       ext4              9eaae22d-e7bf-43a2-836c-f0d528890620    167,2M    21% /boot
└─sdb2       LVM2_member       kc25UG-ScqZ-3Zel-Pu9n-N18g-ykO3-vdl0oP
  ├─vg0-swap swap              fcda94f0-ee95-4db3-9def-2cade50402c8                [SWAP]
  └─vg0-root ext4              75b01bd6-6d46-41db-ad47-092df6fd5d96      4,7G    17% /
sr0
```

Fonte: Do autor com base no sistema deste estudo (2020).

Na Figura 8, podemos observar que o volume “sda”, onde foram efetuados os testes, está separado do local onde se encontra o sistema operacional. Neste caso, a unidade “sda” já é uma unidade virtual montada com a controladora de discos RAID, baseada em *hardware*. Sendo assim, pode-se verificar que os quatro discos aparecem como sendo apenas um. Isto ocorre, porque quem faz todo o trabalho de agrupar os discos é a controladora RAID.

O mesmo não acontece quando os discos estão conectados na controladora *onboard,* que são gerenciados pelo sistema de arquivos ZFS.

Figura 9 – Apresentação dos discos com RAIDZ-1 do ZFS

```
root@linux0196:~# lsblk -f
NAME         FSTYPE      LABEL UUID                                   FSAVAIL FSUSE% MOUNTPOINT
sda
├─sda1
└─sda9
sdb
├─sdb1
└─sdb9
sdc
├─sdc1
└─sdc9
sdd
├─sdd1
└─sdd9
sde
├─sde1       ext4              9eaae22d-e7bf-43a2-836c-f0d528890620    167,2M    21% /boot
└─sde2       LVM2_member       kc25UG-ScqZ-3Zel-Pu9n-N18g-ykO3-vdl0oP
  ├─vg0-swap swap              fcda94f0-ee95-4db3-9def-2cade50402c8                [SWAP]
  └─vg0-root ext4              75b01bd6-6d46-41db-ad47-092df6fd5d96      3,6G    35% /
sr0
root@linux0196:~#
```

Fonte: Do autor com base no sistema deste estudo (2020).

Conforme se observa na Figura 9, quando o *pool* é montado utilizando o *software,* o sistema operacional consegue visualizar todos os discos e quem faz toda a gerência da redundância e dos recursos de IO é o sistema de arquivos ZFS.

4.3.4 Pacotes instalados no ambiente

Os pacotes necessários para os testes foram instalados com o gerenciador de pacotes padrão do *Debian*.

Figura 10 – Pacotes básicos que foram instalados

```
apt-get install make gcc g++ net-tools nmap tcpdump iotop htop mariadb-server postgresql tmux iotop -y
```

Fonte: Do autor com base no sistema deste estudo (2020).

Os pacotes acima instalados, Figura 10, foram utilizados para as compilações, monitoramentos, além dos dois mecanismos de banco de dados que foram utilizados nos testes.

O próximo passo foi instalar os pacotes e módulos do sistema de arquivos ZFS, para que ele funcione corretamente. Faz-se necessária a importação e a recompilação do *kernel*, mas, com o Linux Debian 10, o gerenciador de pacotes APT já executa praticamente a maioria das tarefas.

Figura 11 – Comando de instalação do ZFS

```
#Configuração backports
echo "deb http://http.debian.net/debian buster-backports main contrib" >> /etc/apt/sources.list
apt update
apt install linux-headers-`uname -r`
apt install -t buster-backports dkms spl-dkms
apt install -t buster-backports zfs-dkms zfsutils-linux
```

Fonte: Do autor com base no sistema deste estudo (2020).

Na Figura 11, observam-se os comandos utilizados para a instalação do ZFS. Utiliza-se o repositório *backports* do Debian, pois o ZFS ainda não é um pacote oficial do Kernel Linux.

4.3.5 Instalação e configuração do ZABBIX

Foi instalado o agente do ZABBIX para monitorar o servidor em tempo real, durante todos os testes executados. Para a configuração, foi executado o procedimento que consta no *site* da ferramenta.

Figura 12 – Comandos para a instalação do ZABBIX agente

```
# Download de repositórios
wget https://repo.zabbix.com/zabbix/5.0/debian/pool/main/z/zabbix-release/zabbix-release_5.0-1+buster_all.deb

# Instalação dos repositórios
dpkg -i zabbix-release_5.0-1+buster_all.deb

# Atualização de repositórios e instalação do agente
apt update
apt install zabbix-agent

# Ativação dos serviços
systemctl restart zabbix-agent
systemctl enable zabbix-agent
```

Fonte: Do autor com base no sistema deste estudo (2020).

Após a execução dos comandos citados na Figura 12, o agente do ZABBIX já estava instalado e operacional, bastando apenas adicionar o servidor ao monitoramento com os *templates* necessários para monitoramento.

4.3.6 Ajustes de banco de dados MySQL e PostgreSQL

Para que os dois sistemas de banco de dados utilizassem as unidades montadas no *pool* de disco correto, foi necessário fazer alguns ajustes, pois a instalação padrão deixa os dados na montagem do “/var”. As Figuras 13 e 14 demonstram exatamente o que foi reconfigurado nas pastas de dados dos bancos de dados.

Figura 13 – Comandos realizados para o MySQL

```
# Parando o servidor do MySQL
/etc/init.d/mysql stop

# Copiando os arquivos base e mantendo as permissões
cp -R -p /var/lib/mysql /mnt/testes/

# Ajustes nas configurações do serviço
pico /etc/mysql/mariadb.conf.d/50-server.cnf  "editar datadir"

# Inicializando o serviço
/etc/init.d/mysql start
```

Fonte: Do autor com base no sistema deste estudo (2020).

Figura 14 – Comandos utilizados para o Postgresql

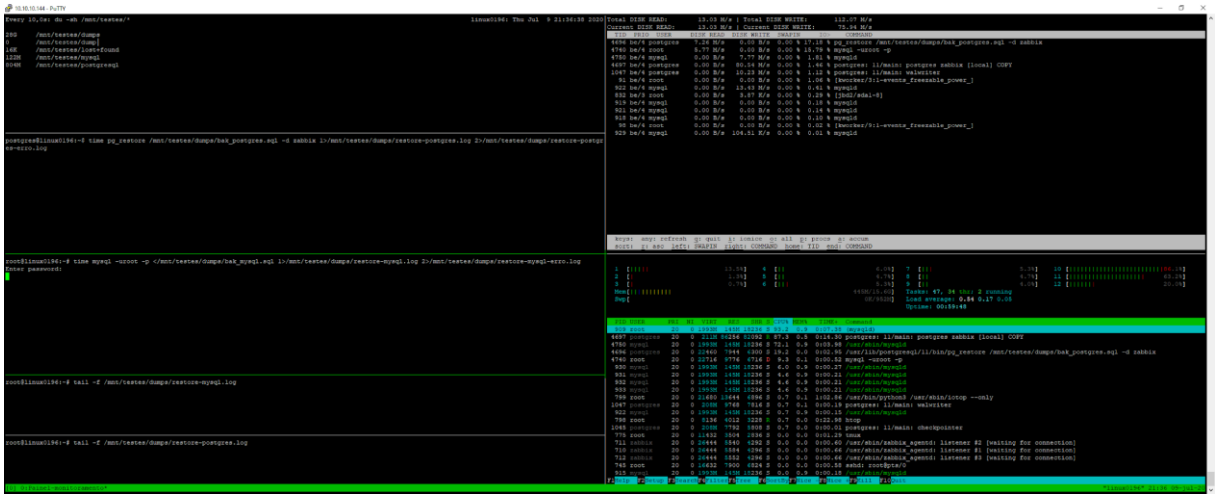
```
# Parando o serviço do Postgresql
/etc/init.d/postgresql stop

# Copiando os arquivos base e mantendo as permissões
cp -R -p /var/lib/postgresql/11/main /mnt/testes/postgresql/

# Ajustes nas configurações do serviço
pico postgres.conf "editar datadir"

# Inicializando o serviço
/etc/init.d/postgresql start
```

Fonte: Do autor com base no sistema deste estudo (2020).

4.3.7 *Dashboard* no ambiente SHELL com tmux

O aplicativo que roda no *shell tmux* foi utilizado para montar o *dashboard* com o *iotop*, o *htop* e os processos que estavam executando no momento dos testes. Este aplicativo foi escolhido, porque ele trabalha no formato de seção. Caso a conexão com o servidor seja fechada por algum motivo qualquer, basta reconectá-la ao servidor e anexar a seção atual do *tmux*. Na Figura 15, podemos ver um exemplo de uma *dashboard*.

Figura 15 – Exemplo de painel do *tmux*



Fonte: Do autor com base no sistema deste estudo (2020).

O *tmux* utiliza muitos comandos especiais para a montagem de um painel. Sendo assim, os utilizados por este estudo foram apenas os necessários para a montagem do *dashboard*.

Figura 16 – Lista dos principais comandos do *tmux*

```
# Iniciar uma sessão do tmux
tmux

# Abrir uma nova aba
control + b depois c

# Navegação entre as abas criadas
control + b depois p
control + b depois n

# Dividir a tela horizontalmente
control + b depois %

# Dividir a tela verticalmente
control + b depois “

# Navegar nas divisões da tela
control + b depois setas direitra esqueda biaxo cima

# Desconectar da sessão
control + b depois d

# Listar sessões abertas
tmux ls

# Reconectar a sessão
tmux attach -t 0 (0 é o número da sessão)
```

Fonte: Do autor com base no sistema deste estudo (2020).

A Figura 16 apresenta uma relação das principais combinações de teclas do *software tmux*. Verifica-se que podemos dividir as telas para ambas as direções e, caso haja alguma instabilidade da conexão SSH com o servidor, basta reconectar a seção que permanecerá ativa.

4.4 Estimativa de custos

A etapa da estimativa de custos é o momento em que é possível ganhar maior adesão do cliente na hora de escolher a tecnologia a ser adquirida. Isso acontece na

maioria dos projetos, pois é o momento de o setor financeiro visualizar não apenas a melhor solução, mas também, o custo da execução.

A Tabela 4 dá uma ideia aproximada dos custos envolvidos em cada umas das duas soluções. A tabela contempla apenas os custos com o controlador de armazenamento, sem contabilizar os discos. Uma característica importante a ser colocada é que o ZFS pode trabalhar de forma híbrida com o disco SSD para *cache* nível dois. Já a controladora RAID precisa de um *pool* totalmente sólido ou mecânico, o que pode aumentar mais os valores ao utilizar a controladora RAID.

Tabela 4 – Custos aproximados de implementação de cada solução.

Custos aproximados de implementação				
Controlador de discos:	**Custo de *hardware***	**Custo de implementação**	**Estimativa de tempo para implementação**	**Totais**
RAID PERC H740P com cache NV de 8 GB	R$4.550,00	R$100,00/hora	5horas	R$5500,00
ZFS	R$0,00	R$100,00/horas	5horas	R$500,00

Fonte: Dados estimados pelo autor (2020).

5 TESTE E ANÁLISE DOS RESULTADOS

Neste capítulo 5, são apresentados os testes realizados, resultados e explicações.

5.1 Testes executados

Foram efetuadas três baterias de testes, pois a ideia era buscar o melhor de cada *pool* de discos em execuções diferentes, para analisar qual *pool* de discos se comporta melhor com I/O, uso de memória para *cache* e CPU.

5.1.1 Importação de banco de dados

O primeiro teste efetuado foi a importação de dois bancos de dados simultâneos (MySQL e *Postgresql*). O MySQL utiliza arquivos de *dump* no formato texto puro e o *Postgresql* binário. A ideia deste primeiro teste foi colocar dois *backups* para restaurar de forma síncrona e medir o tempo total para a execução das tarefas.

Figura 17 – Comandos de importação e uso de disco

```
# Comando "du -sh" nas pastas com os dumps e as bases de dados.
# O dump MySQL com 21G e o do PostgreSQL 7G, totalizando 28G.
28G     /mnt/testes/dumps
46G     /mnt/testes/mysql
92G     /mnt/testes/postgresql

# Comando de importação do Postgres.
“time pg_restore /mnt/testes/dumps/bak_postgres.sql -d dbteste \
1>/mnt/testes/dumps/restore-postgres.log \
2>/mnt/testes/dumps/restore-postgres-erro.log”

# Comando de importação do MySQL.
“time mysql -uroot -p </mnt/testes/dumps/bak_mysql.sql \
1>/mnt/testes/dumps/restore-mysql.log \
2>/mnt/testes/dumps/restore-mysql-erro.log”
```

Fonte: Do autor com base no sistema deste estudo (2020).

Na Figura 17, podemos observar o tamanho dos *dumps* que foram importados, os dois bancos de dados após a importação e também os comandos executados para a importação. A rotina “*time*” foi adicionada na frente de cada comando de importação, para medir o tempo necessário para cada tarefa.

5.1.2 *Benchmark* IOZone

O IOZone foi executada com parâmetros personalizados, para que a coleta de recursos ficasse mais precisa possível. Na primeira execução, utilizou-se “*time ./iozone -a -r 8K -r 16K -r 32K -r 64K -r 128K -s 2G -o -O -i0 -i1 -i2 -p –f /mnt/testes/iozone3_489/src/current/ >saida-iozone-hardware-iops.log*”.

O parâmetro “-a” especifica que a operação é automática. O “-r”, para definir o tamanho do bloco de gravação; o “-s”, o tamanho máximo do arquivo de teste. Já o “-O” indica a saída do comando em IOPS (entradas e saídas por segundo) e o “-o”, minúsculo, a execução do teste na forma *O_SYNC,* que só libera o I/O quando tudo está gravado em disco. Nos nossos testes, a execução foi feita de forma síncrona, assíncrona e a saída foi em IOPS e KBPS. As *flags* “-i0, -i1 e -i2” definem que o teste é do tipo leitura, escrita, releitura, reescrita e leitura e escrita randômica.

5.1.3 Geração de arquivos com o DD

Com o *software* de clonagem *bit-a-bit DD,* criamos uma imagem no disco para testar a gravação de dados em ambos os *pools.* A imagem, com 10GB, também foi utilizada para a execução da leitura dos dados de forma serial. A leitura serial é diferente da randômica, em que os dados ou *bits* estão em ordem, sem fragmentação. Normalmente, este tipo de procedimento consegue reproduzir velocidades maiores, pois a cabeça de leitura dos discos não precisa mover-se a grandes distâncias.

5.2 Resultados

Nesta seção, apresentamos os resultados obtidos em nosso laboratório de testes. Com a intenção de obter maior precisão das informações e evitar que haja algum dado já em memória, entrega-se alguma vantagem na hora da execução dos testes. Foram executados os comandos "*sync*" e "*echo* 3 > */proc/sys/vm/drop_caches*" para a liberação da memória em *cache* e em qualquer outro tipo de *buffer*, em tempo de execução do *Kernel*. Na Figura 18, temos uma sequência dos comandos citados acima e como ele libera as informações contidas no *cache*.

Figura 18 – Limpeza de *buffer* em memória

```
[root@cpanel ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:          15726        1633        1805          82       12286       13677
Swap:          4055         377        3678
[root@cpanel ~]#
[root@cpanel ~]# sync; echo 3 > /proc/sys/vm/drop_caches;
[root@cpanel ~]#
[root@cpanel ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:          15726        1792       13715          82         218       13614
Swap:          4055         377        3678
[root@cpanel ~]#
```

Fonte: Do autor com base no sistema deste estudo (2020).

5.2.1 Importação dos bancos de dados

Na importação dos bancos de dados, obtivemos resultados interessantes. O MySQL se comportou.

Tabela 5 – Resultados dos testes em bancos de dados

Processamento dos *backups* de banco de dados		
Banco de dados	***Pool* de discos**	**Tempo**
PostgreSQL	*Hardware*	764 minutos
PostgreSQL	ZFS	370 minutos
MySQL	*Hardware*	888 minutos
MySQL	ZFS	2451 minutos

Fonte: Dados coletados pelo autor (2020).

Observa-se na Tabela 5 que obtivemos valores cruzados. Cada banco de dados se comportou melhor em sistemas de arquivos opostos. A importação do banco de dados binário do *PostgreSQL* obteve melhor performance no *pool* de disco ZFS. Já no arquivo de *dump* do MySQL, em formato texto puro, onde as inserções de dados acontecem linha após linha, conseguiu-se atingir um tempo muito melhor no *pool* de discos *hardware*.

Neste procedimento, foi utilizado o ZABBIX para monitorar o comportamento do servidor. A ideia foi analisar o impacto que as duas tecnologias de armazenamento geram nos recursos de CPU, de memória e de disco.

Tabela 6 – Consumo de CPU

CPU *Load* 15 minutos		
Tipo de pool	**Média**	**Máxima**
Hardware	2,64	4,27
ZFS	3,30	6,41

Fonte: Dados coletados pelo autor (2020).

Na Tabela 6, temos o monitoramento de CPU, que é a carga de processos enfileirados para serem processados. Observa-se que, utilizando o *pool* de discos

baseado em *software*, a elevação da carga aumentou em um ponto, o que, neste caso, não significaria muito, levando em conta as unidades de processamento atuais.

Tabela 7 – Consumo de memória

Memória do ambiente		
Tipo de *pool*	**Média**	**Máxima**
Hardware	1,01 GB	1,04 GB
ZFS	9,21 GB	9,56 GB

Fonte: Dados coletados pelo autor (2020).

Na Tabela 6, que apresenta o consumo de memória das duas formas de armazenamento, temos valores que precisamos levar em consideração, ao escolher um sistema de armazenamento baseado em *hardware* ou *software*. A controladora não teve um consumo alto de memória, o que pode ser bom em alguns casos, mas um problema, em outros.

Isto acontece, porque ela possui 512MB de memória interna, dedicada ao *cache;* por isso, não utiliza os recursos do ambiente. Neste caso, se, em algum momento, a controladora necessitar de mais memória, ela deve ser adquirida nova, mas, geralmente, o valor de mercado é alto. Caso seja implementado com o ZFS, ele vem com o ARC instalado como padrão, que é o responsável pelo gerenciamento do *cache*. O valor padrão de memória que ele separa do ambiente é a metade de toda a memória. No caso do estudo acima, foi quase 8GB de memória.

Os valores de memória do ARC podem ser alterados a qualquer momento, mas, normalmente, quanto maior, melhor será a performance. Neste momento, precisamos levar em consideração a escolha do sistema de armazenamento, pois o ZFS necessita de uma grande fatia de memória do ambiente, para manter-se rápido e constante.

Tabela 8 – Latência de disco em ms

Latência de disco			
Tipo de *pool*	**Tipo**	**Média**	**Máxima**
Hardware	Escrita	3,25 ms	64,43 ms
ZFS		5,30 ms	26,42 ms
Hardware	Leitura	545,74 ms	1626,89 ms
ZFS		25,48 ms	81,50 ms

Fonte: Dados coletados pelo autor (2020).

O comportamento dos armazenamentos durante a leitura e a escrita pode ser analisado na Tabela 8. O sistema de armazenamento baseado em *software* mostrou-se com bons valores em todos os itens. Enquanto eles estavam gravando as informações no *pool*, ambos mantiveram uma média parecida, mas, durante as leituras, o ZFS manteve-se melhor.

Este comportamento pode ser explicado, considerando que o ZFS reservou 8GB de memória física do sistema para as transações em disco. Sendo assim, ele tinha uma capacidade maior de *cache*. Este comportamento de dedicar memória do ambiente para operações de disco é ideal para implementações em que não se fará tanto uso de memória para outras aplicações no mesmo *hardware*.

5.2.2 Execução do *Benchmark* IOZone

Os resultados do IOZone foram importantes para esta pesquisa, pois nele coletamos dados com saídas em KB/s e IO/s. Trata-se de informações importantes que explicam o porquê de a importação do banco de dados binário mostrar-se melhor com o ZFS.

Na Tabela 9, temos a representação dos dados obtida pelo *benchmark* no formato de saída em IO/s, com gravação síncrona no disco. A gravação síncrona trabalha de uma forma diferente, sendo possível a medição apenas quando os dados realmente já estiverem gravados nos discos.

Tabela 9 – Saída IOZone em IO/s em modo SYNC

IOZone - Saída síncrona em IO/s			
Tipo de *pool*	**Bloco**	**Escrita**	**Leitura**
Hardware	64 KB	1217 iops	40269 iops
ZFS	64 KB	32 iops	33 iops
Hardware	128 KB	691 iops	37254 iops
ZFS	128 KB	33 iops	33 iops

Fonte: Dados coletados pelo autor (2020).

Como podemos observar, para tipos de gravações síncronas em disco, o ZFS tem baixa capacidade de gravação e leitura se comparado a uma controladora baseada em *hardware*. Sendo assim, um agrupamento de discos com ZFS tem a performance bastante degradada quando os dados não podem ficar em *cache*.

A Tabela 10 é o mesmo processo síncrono, mas com a saída em KB/s, apenas para termos um parâmetro de velocidade.

Tabela 10 – Saída IOZone em KB/s em modo SYNC

IOZone - Saída síncrona em KB/s			
Tipo de *pool*	**Bloco**	**Escrita**	**Leitura**
Hardware	64 KB	78746 KB/s	3343023 KB/s
ZFS	64 KB	1812 KB/s	2004444 KB/s
Hardware	128 KB	87602 KB/s	3544288 KB/s
ZFS	128 KB	3586 KB/s	2026791 KB/s

Fonte: Dados coletados pelo autor (2020).

Na sequência, os resultados dos testes com o IOZone, sem a opção de gravação de dados sincronizados.

Tabela 11 – Saída IOZone em IO/s, sem gravação assíncrona

IOZone - Saída assíncrona em IO/s			
Tipo de *pool*	**Bloco**	**Escrita**	**Leitura**
Hardware	64 KB	21223 iops	50353 iops
ZFS	64 KB	5196 iops	30957 iops
Hardware	128 KB	10922 iops	24565 iops
ZFS	128 KB	3094 iops	17767 iops

Fonte: Dados coletados pelo autor (2020).

Na Tabela 11, verifica-se o quanto o ZFS melhora o seu desempenho quando a gravação deixa de ser síncrona. Uma gravação não sincronizada é aquela em que os dados ficam em *cache* enquanto os discos estão ocupados, permitindo a emissão de mais entradas e saídas.

Agora, na Tabela 12, o mesmo teste, mas com as saídas em KB/s.

Tabela 12 – Saída IOZone em KB/s e sem gravação assíncrona

IOZone - Saída assíncrona em IO/s			
Tipo de pool	**Bloco**	**Escrita**	**Leitura**
Hardware	64 KB	1421327 KB/s	3401816 KB/s
ZFS	64 KB	334175 KB/s	2473260 KB/s
Hardware	128 KB	1373134 KB/s	5117998 KB/s
ZFS	128 KB	362589 KB/s	2585440 KB/s

Fonte: Dados coletados pelo autor (2020).

5.2.3 Gravação e leitura serial com DD

Nesta etapa dos testes, o objetivo é a demonstração de como se comportam as duas tecnologias em gravações seriais. Uma gravação serial é quando um dado é posto atrás do outro, facilitando a escrita e a posterior leitura.

Tabela 13 – Comando DD gravação e leitura com controladora

DD - Gravação e leitura com controladora			
Operação	**Tamanho**	**Tempo**	**Velocidade**
Gravação	10GB	34,53 segundos	297MB/s
Leitura	10GB	41,47 segundos	247MB/s

Fonte: Do autor com base no sistema deste estudo (2020).

Na Tabela 13, apresentam-se informações de performance do DD, rodando sobre o *pool* de discos, baseado em *hardware*. Podemos observar que na leitura e na escrita serial, as velocidades quase se equiparam.

Agora, na Tabela 14, as informações de leitura e escrita do DD em um *pool* de discos baseado em *software*.

Tabela 14 – Comando DD gravação e leitura com ZFS

DD - Gravação e leitura com ZFS			
Operação	**Tamanho**	**Tempo**	**Velocidade**
Gravação	10GB	75,94 segundos	135MB/s
Leitura	10GB	59,52 segundos	172MB/s

Fonte: Do autor com base no sistema deste estudo (2020).

Nos testes de leitura e escrita em disco de forma serial, o agrupamento de disco baseado em software obteve resultados inferiores nos dois sentidos, o que não quer dizer que o ZFS é pior, pois, como verificamos na importação do banco de dados *PostgreSQL*, ele é muito bom com arquivos binários.

6 CONCLUSÃO

O presente trabalho buscou fazer uma comparação entre uma tecnologia focada em *hardware,* já totalmente consagrada, e outra, em *software*, que vem crescendo nas aplicações convencionais. Com a execução de testes de performance e de estresse em ambas as estruturas, pode-se observar e armazenar informações de como cada uma se comporta em ambientes críticos. Também coletamos métricas para comparar o desempenho das duas soluções, com o objetivo de descobrir em quais situações pode-se trocar tecnologias baseadas em *hardware* por *software*. As baseadas em *software* têm a vantagem de terem menor custo e a possibilidade de aplicá-las para clientes de menor porte, garantindo benefícios compatíveis com suas necessidades.

Durante os estudos do sistema de arquivos ZFS, percebeu-se, por meio de notícias oficiais, que algumas distribuições *Linux* já iniciaram a implementação do ZFS de forma experimental, em suas instalações. Entre elas pode-se citar a Canonical que mantém o Linux Ubuntu (UBUNTU, 2020), o virtualizador XCP-ng (XCP-NG, 2018) e o sistema de *backup* Bacula, que utiliza os artifícios para a deduplicação de dados (BACULA LAT, 2018). Com essas informações, pode-se reforçar a nossa hipótese de que o ZFS é um grande substituto para implementações de menor custo.

Conforme os resultados obtidos, o ZFS necessita de muita memória do sistema operacional para fazer o *cache*. Sem o *cache*, o *pool* de disco torna-se muito lento e ineficaz. Já a capacidade de fazer entradas e saídas em disco do ZFS é inferior ao de uma controladora *hardware*, quando os dados precisam ser gravados de maneira síncrona nos disco físicos. Foram obtidos melhores resultados com a controladora

hardware, manipulando arquivos de formas binárias, como no caso da importação do banco de dados do *PostgreSQL*.

Um outro fator que chamou atenção foi a baixa latência que o ZFS demonstrou em relação às leituras no *pool* de discos, o que pode ser explicado pelo fato de ele armazenar muita informação em memória, conseguindo assim um tempo de latência menor; porém, o tempo de escrita em disco não teve relevância, ficando as duas tecnologias muito próximas.

A controladora *hardware* mostrou-se superior nas operações de processamento de arquivos pequenos, que exigem mais entradas e saídas no *pool* de discos. Ela faz praticamente todo o processamento em seu próprio *chip*, evitando gerar impactos de performance nos outros recursos do servidor.

Sendo assim, deve-se estudar e projetar muito bem o cenário antes de implementar uma solução de armazenamento baseada em *software*, mas, em muitas situações, pode funcionar de forma adequada. Um fator a ser observado é a quantidade de memória que o servidor terá disponível para a execução de *cache* nível 1.

Nos testes, não efetuamos simulações com *cache* nível 2, que é executado sobre SSDs, nem a parte de replicação do ZFS. Então, pode-se citar como motivação para futuros trabalhos a aplicação da ferramenta com essas duas características que são muito importantes para a redundância e a velocidade do sistema de arquivos ZFS.

REFERÊNCIAS

BACULA LAT. **Tutorial Deduplicação Sistema de Arquivos em Nível de Blocos com Volumes Alinhados (Bacula 9.0.8 e superior)**. 2018. Disponível em: http://www.bacula.lat/tutorial-deduplicacao-em-nivel-de-blocos-com-volumes-alinhados-bacula-7-9-9-0-e-superior/. Acesso em: 15 out. 2020.

DELL. **iDrac**. 2020. [Sistema interno da DELL].

EKO, D. Widianto; AGUNG, B. Prasetijo; AHMAD, Ghufroni. On the implementation of zfs (zettabyte file system) storage system. *In:* INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY, COMPUTER, AND ELECTRICAL ENGINEERING (ICITACEE), 3., 2016. **Anais** [...]. [S.l.]: Semarang, 2016. p. 408-413. Disponível em: https://ieeexplore.ieee.org/document/7892481. Acesso em: 14 fev. 2020.

FACCIONI FILHO, Mauro. **Gestão da infraestrutura do datacenter.** Palhoça: UnisulVirtual, 2016. Disponível em: https://www.researchgate.net/publication/319913896_Gestao_da_Infraestrutura_do_Datacenter. Acesso em: 11 ago. 2020.

FERREIRA, Diogo Assis; MATAYOSHI, Getúlio Yassuyuki. **Aplicação de RAID em sistema de arquivos distribuídos**. Brasília: UnB, 2016. Disponível em: https://bdm.unb.br/bitstream/10483/16024/1/2016_DiogoAssisFerreira_GetulioYassuykiMatayoshi_tcc.pdf. Acesso em: 31 mai. 2020.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa.** 1. ed. Porto Alegre: UFRGS, 2009. Disponível em: http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf. Acesso em: 13 jun. 2020.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002. Disponível em: http://home.ufam.edu.br/salomao/Tecnicas%20de%20Pesquisa%20em%20Economia/Textos%20de%20apoio/GIL,%20Antonio%20Carlos%20-%20Como%20elaborar%20projetos%20de%20pesquisa.pdf. Acesso em: 31 mai. 2020.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008. Disponível em: https://ayanrafael.files.wordpress.com/2011/08/gil-a-c-mc3a9todos-e-tc3a9cnicas-de-pesquisa-social.pdf. Acesso em: 20 out. 2020.

GOOGLE IMAGENS. 2020. Disponível em: https://www.google.com/imghp?hl=pt-br. Acesso em: 20 ago. 2020.

GURJAR, Devyani; KUMBHAR, Satish S. A review on performance analysis of ZFS e BTRFS. *In:* INTERNATIONAL CONFERENCE ON COMMUNICATION AND SIGNAL PROCESSING (ICCSP), 2019. **Anais** [...]. Chennai, Índia: INSPEC, 2019. Disponível em: https://ieeexplore.ieee.org/document/8698103. Acesso em: 07 de setembro de 2020.

MONTEIRO, Mariana. **Armazenamento de dados: storage area network.** 2005. 33f. Monografia (Curso de Sistemas de Informação) – Faculdade de Tecnologia Padre Anchieta, Jundiaí, 2005.

ORACLE. **Guia de administração do oracle solaris ZFS**. 2013. Disponível em: https://docs.oracle.com/cd/E38904_01/pdf/E38890.pdf. Acesso em: 6 set. 2020.

PHRONCHANA, Veerapat; NUPAIROJ,Natawut; PIRORNSOPA, Krerk. Performance evaluation of ZFS and LVM (with ext4) for scalable storage system. *In:* EIGHTH INTERNATIONAL JOINT CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING (JCSSE), 2011. **Anais** [...].Nakhon Pathom, Tailândia: IEEE, 2011. Disponível em: https://ieeexplore.ieee.org/document/5930130. Acesso em: 07 set. 2020.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho acadêmico. São Leopoldo: FEVALE, 2013. Disponível em: http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf. Acesso em: 31 mai. 2020.

SEAGATE. **Modos de RAID**. 2020. Disponível em: https://www.seagate.com/br/pt/manuals/network-storage/business-storage-nas-os/raid-modes/. Acesso em: 10 mar. 2020.

SIQUEIRA, Luciano Antônio. **Certificação LPI-2.** 2. ed. [S.l.]: The Linux Professional Institute, 2009.

TANENBAUM, Andrew S; WOODHULL, Albert S. **Sistema operacionais:** projeto e implementação. 3. ed. Londres: Pearson University, 2008.

UBUNTU. **ZFS focus on Ubuntu 20.04 LTS.** 2020 Disponível em: https-://ubuntu.com/blog/zfs-focus-on-ubuntu-20-04-lts-whats-new. Acesso em: 18 out. 2020.

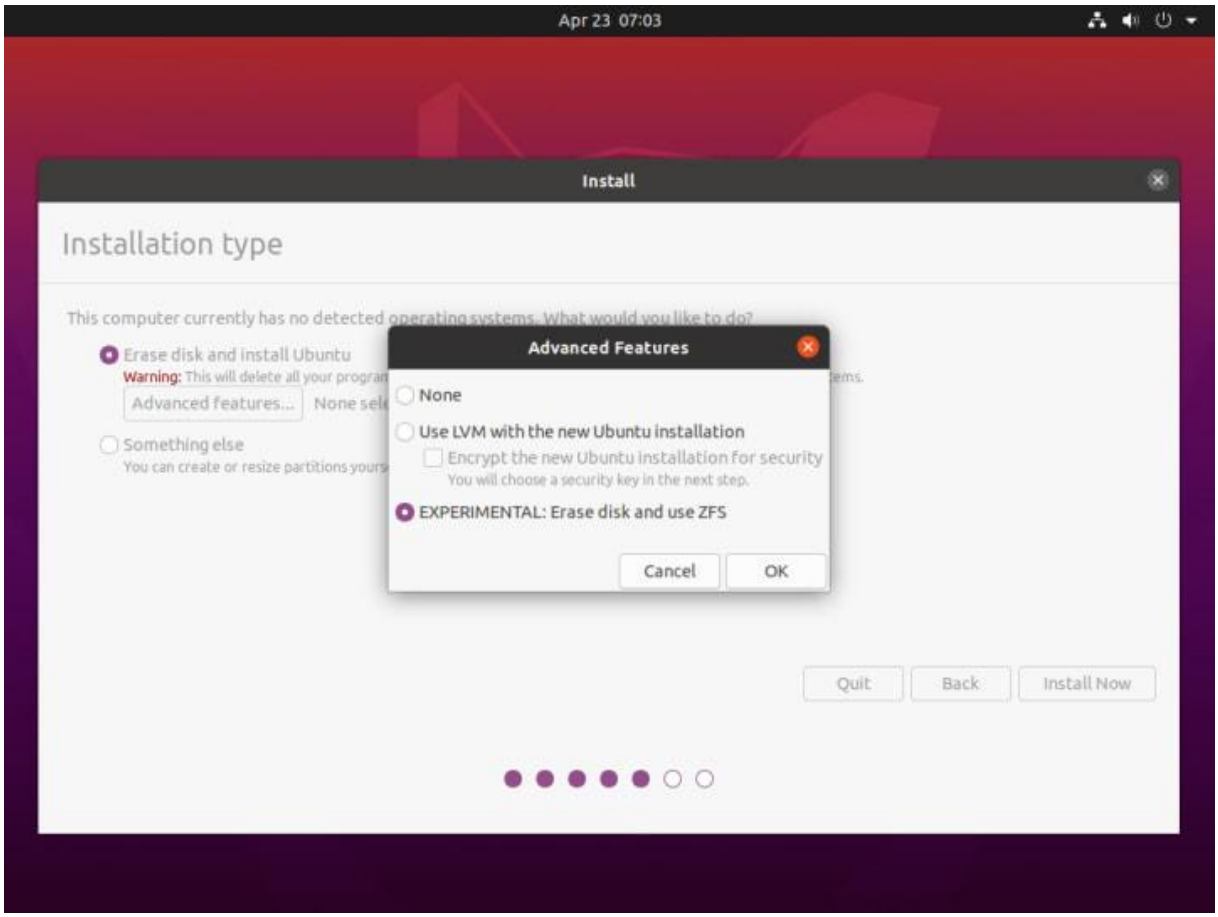
WIKEPEDIA. **Sistema de ficheiros**. 2020. Disponível em: https://pt.wikipedia.org/wiki/Sistema_de_ficheiros. Acesso em: 10 ago. 2020.

WOJSŁAW, Damian. **Introducing zfs on linux. Understand the basics of storage with ZFS**. Poland: Apress, 2017. E-Book. Disponível em: https://books.google.com.br/books?id=-UNADwAAQBAJ&printsec=frontcover&dq=zfs+on+linux&hl=pt-BR&sa=X&ved=0ahUKEwjhgJbo_7XoAhXlHrkGHZ4PC2IQ6AEIKjAA#v=onepage&q&f=true. Acesso em: 07 set. 2020.

XCP-NG. **XCP-ng with ZFS**. 2018. Disponível em: https://xcp-ng.org/blog/2018/08/03/xcp-ng-with-zfs/. Acesso em: 18 out. 2020.

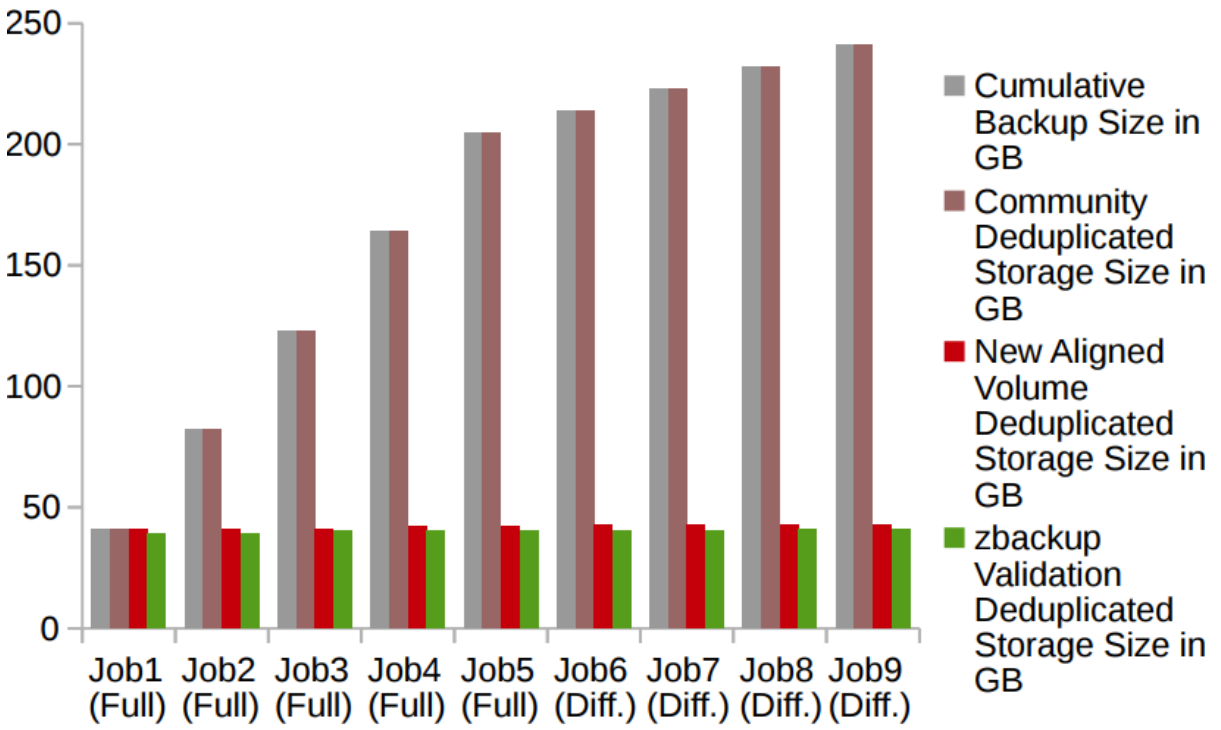
ZHANG, Yupu *et al.* Zettabyte reliability with flexible end-to-end data integrity. *In:* SYMPOSIUM ON MASS STORAGE SYSTEMS AND TECHNOLOGIES (MSST), 9., 2013. **Anais** [...]. long Beach, Califórnia: IEEE, 2013. Disponível em: https://ieeexplore.ieee.org/document/6558423. Acesso em: 14 fev. 2020.

ANEXO A – Tela do Linux Ubuntu 20.04 da versão experimental do ZFS



Fonte: https://didrocks.fr/2019/10/11/ubuntu-zfs-support-in-19.10-zfs-on-root/

ANEXO B – Comparativos de utilização de espaço em disco com a deduplicação ativada no Bacula Backup



Fonte: http://www.bacula.lat/tutorial-deduplicacao-em-nivel-de-blocos-com-volumes-alinhados-bacula-7-9-9-0-e-superior/

UNIVATES

R. Avelino Tallini, 171 | Bairro Universitário | Lajeado | RS | Brasil
CEP 95900.000 | Cx. Postal 155 | Fone: (51) 3714.7000
www.univates.br | 0800 7 07 08 09