Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de Armazenamento Definido por Software em uma Arquitetura Hiperconvergente de Containers

Rodrigo da Silva Leite Morais

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof. Dr.a Priscila América Solís Mendez Barreto

Brasília
2019

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

da Silva Leite Morais, Rodrigo
du Utilização de Armazenamento Definido por Software em uma Arquitetura Hiperconvergente de Containers / Rodrigo da Silva Leite Morais; orientador Priscila América Solís Mendez Barreto. -- Brasília, 2019.
p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2019.

1. Computação em Nuvem. 2. Hiperconvengência. 3. Sistemas de Arquivos Distribuidos. 4. Containers. I. América Solís Mendez Barreto, Priscila, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de Armazenamento Definido por Software em uma Arquitetura Hiperconvergente de Containers

Rodrigo da Silva Leite Morais

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr.a Priscila América Solís Mendez Barreto (Orientadora)
CIC/UnB

Prof. Dr. Eduardo Adilio Pelinson Alchieri
CIC/UnB

Prof. Dr. André Luiz da Costa Carvalho
IComp/UFAM

Prof. Dr.a Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 5 de Julho de 2019

Dedicatória

Dedico este trabalho à minha família, que sempre me guiou para o caminho correto, e em especial minha esposa Daniela, que com o seu carinho e compreensão, me torna cada dia um homem melhor.

Agradecimentos

A minha esposa Daniela, que esteve do meu lado em cada momento desta jornada. A minha família e amigos pelo apoio e carinho. A todos os meus professores, colegas de ensino e colegas de trabalho, em especial à Prof.ª Dr.ª Priscila América Solís Mendez Barreto, que me apoiou na realização de todas etapas deste projeto. E principalmente a Deus, que é a razão de tudo.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Esse trabalho propõe uma arquitetura hiperconvergente de *containers* que usa um sistema de armazenamento definido por software. A arquitetura promove a otimização dos recursos de computação e armazenamento, principalmente ao consolidar os recursos de armazenamento disponíveis em diversos servidores dispersos em um *datacenter*. Para avaliar a arquitetura proposta o sistema foi submetido a diferentes configurações e intensidades de carga de trabalho oriúndas dos *datacenters* da Microsoft e do Yahoo. Os resultados mostram que a arquitetura proposta apresenta potencial, visto que consegue obter melhor desempenho e eficiência do que uma arquitetura tradicional de armazenamento local.

Palavras-chave: Computação em Nuvem, Hiperconvergência, Armazenamento, Containers, Sistemas de Arquivos Distribuídos, GlusterFS, Docker, MongoDB, Cassandra

Abstract

This work proposes a hyperconverged architecture for textit containers that uses a software-defined storage. The proposed architecture promotes the optimization of compute and storage resources, especially by consolidating the storage resources available on multiple servers scattered in a datacenter. In order to evaluate the proposed architecture, the system was submitted to different configurations and workload intensities from Microsoft and Yahoo datacenters. The results show that the proposed architecture has potential because, achieving better performance and efficiency than a traditional local storage architecture.

Keywords: Cloud Computing, Hiperconvergence, Storage, Containers, Distributed File Systems, GlusterFS, Docker, MongoDB, Cassandra

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capitulo serão descritos o tema e a sua justificativa, assim como os objetivos e os benefícios a serem alcançados pela proposta apresentada neste trabalho.

1.1 Apresentação

Um cenário ainda comum no *datacenter* de muitas empresas são os recursos de computação e armazenamento serem tratados como duas entidades individuais e distintas. Tradicionalmente nesses *datacenters* existe uma infraestrutura de servidores que provê computação e outra infraestrutura separada que provê armazenamento. Essas duas são interligadas por uma terceira infraestrutura de rede de armazenamento, que é responsável pela comunicação entre os recursos de computação e os recursos de armazenamento. Esse tipo de arquitetura ainda é amplamente utilizada e vem sendo modernizada periodicamente pela indústria, existindo atualmente equipamentos monolíticos com grande capacidade e desempenho para o armazenamento de dados.

Porém também é comum encontrar nesses *datacenters* servidores que possuem seus próprios recursos de armazenamento, ou seja, servidores que possuem unidades de armazenamento diretamente conectadas. Essas unidades de armazenamento são usualmente utilizadas para a instalação do sistema operacional ou armazenamento de dados não-críticos. Como os dados críticos são armazenados em equipamentos dedicados, habitualmente as unidades diretamente conectadas aos servidores tem baixa utilização e são configuradas com pouca ou nenhuma tolerância a falhas.

Esses recursos de armazenamento dispersos em vários servidores ao serem somados alcançam valores consideráveis de capacidade. Essa quantidade de recursos pode ser utilizada para o armazenamento de dados críticos, desde que o desempenho e os mecanismos de tolerância a falha se equiparem aos de um tradicional equipamento dedicado para armazenamento de dados.

1.2 Objetivo

Este trabalho tem como objetivo geral propor uma arquitetura hiperconvergente de *containers* que usa um sistema de armazenamento definido por software. A hiperconvergência é uma arquitetura que permite consolidar computação, armazenamento e virtualização de servidores em um único sistema. O armazenamento definido por software separa o plano de controle do plano de dados de um sistema de armazenamento, permitindo que um software controle diretamente os recursos de armazenamento. A arquitetura proposta pode ser implantada em um provedor de serviços em nuvem com o objetivo de consolidar os recursos de armazenamento disponíveis em diversos servidores dispersos em um *data-center*, de forma que os recursos se comportem com um único volume massivo e tolerante a falhas. Esse volume é então apresentado de volta para cada um dos servidores que cederam recursos, provendo-os com uma unidade de armazenamento de alta capacidade e tolerante a falhas.

Os objetivos específicos deste trabalho são:

- Construir um volume distribuído tolerante a falhas com os recursos locais de cada servidor (usando armazenamento definido por software).
- Apresentar o volume distribuído para cada servidor que cedeu recursos (usando hiperconvergência).
- Armazenar os dados de aplicações dentro do volume distribuído (usando containers).

Considerando a atual tendência de otimizar o uso de recursos em ambientes de computação em nuvem, foi definido o problema de propor uma arquitetura que possa ser implementada em uma empresa e que tem como principal objetivo atender às crescentes demandas de armazenamento de dados, ao mesmo tempo em que integra ideias e tecnologias do estado da arte. Dessa forma, a proposta tem como objetivo, além de prover armazenamento, favorecer uma evolução técnica sólida, robusta e de baixo custo para as futuras demandas.

Outros objetivos deste trabalho são:

- Fazer um levantamento do estado da arte das tecnologias que podem ser integradas na arquitetura.
- Propor novas ideias que venham dar um caráter inovador e evolutivo à arquitetura proposta.
- Avaliar o desempenho da arquitetura proposta com uma abordagem sistemática e indicar futuras possibilidades.

Um caso particular de estudo desse trabalho é o observado na CASSI (Caixa de Assistência dos Funcionários do Banco do Brasil), uma empresa operadora de planos de saúde, que possui em seu *datacenter* diversos servidores com dois discos diretamente conectados que são utilizados basicamente para instalação do hipervisor. O somatório desses recursos praticamente ociosos é estimado em 100TB, um valor considerado grande para a atual realidade e que rivaliza com a capacidade dos equipamentos dedicados instalados.

1.3 Justificativa

Um dos recentes pontos de interesse de pesquisa na área de computação em nuvem é o armazenamento de dados de aplicações executando em *containers* [1] [2] [3]. Um *container*, por definição, é uma entidade efêmera, ou seja, todos os dados armazenados no *container* durante sua execução são perdidos ao término da execução. Entretanto existem mecanismos para assegurar a gravação persistente de dados em um *container*. Analisando os trabalhos que abordam a questão do armazenamento persistente verifica-se que a maioria foca no armazenamento de dados localmente no *host*, sendo que essa configuração na maioria das vezes não provê tolerância a falhas de maneira adequada.

Diante desse cenário a utilização de uma tecnologia de armazenamento definido por software aparenta ser uma arquitetura interessante para o armazenamento persistente de dados de *containers*. Diversos trabalhos recentes nessa área investigam o desempenho de diferentes softwares que implantam sistemas de arquivos distribuídos [4] [5] [6] ou analisam o desacoplamento entre o plano de controle e o plano de dados [7] [8] [9]. Um assunto recorrente nesses trabalhos são as características de escalabilidade, de tolerância a falhas e da possibilidade de *hardware* comum de prateleira. Essas características são motivações para investigar a aplicabilidade dessa tecnologia dentro da abordagem proposta neste trabalho.

Mesmo aliando a questão do armazenamento persistente de dados em *containers* com o uso de armazenamento definido por software, os recursos de computação e armazenamento ainda estão sendo tratados como duas entidades individuais e distintas. Uma infraestrutura hiperconvergente permite unificar os dois temas, ao agregar computação e armazenamento na mesma solução [10]. Estudos recentes sobre esse tópico indicam essa arquitetura como opção para o armazenamento e processamento de dados massivos gerados por dispositivos IoT (do inglês *Internet of Things*) [11] [12]. No contexto dessa proposta de pesquisa, a hiperconvergência permite que os recursos físicos de armazenamento de diversos nós que antes proviam somente computação sejam virtualizados, agregados e reapresentados a cada nó como uma unidade única onde toda abstração subjacente é implantada em software.

A contribuição esperada com este trabalho é o desenvolvimento de uma arquitetura de armazenamento de dados para computação em nuvem que seja escalável e promova alto desempenho das aplicações nela hospedadas. Essa arquitetura disponibilizará um ambiente hiperconvergente escalável para a hospedagem de containers, cuja abstração dos recursos físicos de armazenamento utilizarão uma metodologia inovadora.

1.4 Estrutura

Os próximos capítulos desse trabalho estão estruturados da seguinte maneira:

- O capítulo 2 apresenta a revisão bibliográfica onde são apresentados os conceitos utilizados no trabalho, os desafios do armazenamento de dados da computação em nuvem e os trabalhos relacionados aos temas que serão abordados na proposta de pesquisa.
- O capítulo 3 apresenta a proposta de pesquisa onde é apresentada a arquitetura da solução e as ferramentas utilizadas.
- O capítulo 4 apresenta a metodologia de avaliação de desempenho, os resultados experimentais, análise dos resultados e viabilidade da utilização da solução proposta.
- O capítulo 5 apresenta as conclusões e trabalhos futuros desta pesquisa.

Capítulo 2

Revisão Bibliográfica

Neste capítulo serão apresentados os conceitos teóricos da revisão da literatura associada ao tema, assim como os paradigmas e tecnologias que serão utilizados no Capítulo 3 para definir o foco desta dissertação.

2.1 Conceitos de Computação em Nuvem

Nesta seção serão apresentadas as definições de alguns conceitos de computação em nuvem e tecnologias relacionadas que serão utilizados neste trabalho.

2.1.1 Computação em Nuvem

Computação em nuvem é um modelo para permitir acesso de forma ubíqua, conveniente e sob demanda a um conjunto compartilhado de recursos de computação (redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços [13].

Segundo o NIST [13], o modelo de computação em nuvem é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação. As características essenciais foram definidas como:

- *Serviço self-service sob demanda:* um consumidor pode provisionar recursos computacionais unilateralmente (tais como processador, memória ou armazenamento), de forma automática e sem interação humana com o fornecedor do servidor.
- *Amplo acesso à rede:* Os recursos são disponibilizados em rede e acessados por meio de mecanismos padrão que promovem acessos a clientes heterogêneos (celulares, tablets ou estações de trabalho).

- *Compartilhamento de recursos:* Os recursos computacionais fornecidos são compartilhados entre os consumidores, com diferentes recursos físicos ou virtuais dinamicamente disponibilizados ou realocados para um consumidor de acordo com a demanda. O consumidor não tem controle ou conhecimento do local exato onde o recurso está localizado, sendo possível somente definir a localização do serviço em um plano maior de abstração (país, estado ou *datacenter*).

- *Rápida elasticidade:* Os recursos podem ser provisionados e desprovisionados, em alguns casos de forma automática, de forma a escalar crescendo ou diminuindo de acordo com a demanda. Do ponto de vista do consumidor os recursos disponíveis parecem ser ilimitados e podem ser utilizados em qualquer quantidade a qualquer hora.

- *Medição de serviço:* Serviços em nuvem controlam e otimizam os recursos usando medições no nível de abstração do serviço sendo fornecido, tais como armazenamento, processamento, usuários ativos, entre outros. Os recursos utilizados podem ser monitorados, controlados e com relatórios, visando prover transparência tanto para o provedor quanto para o consumidor.

A computação em nuvem pode ser provida em diferentes níveis de abstração e vantagens, a depender das necessidade e objetivos dos consumidores [14]. Nessa abordagem um provedor pode disponibilizar como serviço desde uma aplicação de escritório ou um ambiente para consumidores construírem seus próprios aplicativos, até todos os recursos tradicionais de computação (processamento, armazenamento, memória e rede) diretamente para o consumidor. Por esse motivo o NIST [13] definiu os seguintes modelos de serviços para computação em nuvem:

- *Software as a Service (SaaS):* Capacidade de prover ao consumidor uma aplicação executando em uma infraestrutura em nuvem. A aplicação é acessada a partir de vários dispositivos clientes, por meio de interface WEB ou aplicativo instalado. O consumidor não gerencia nem controla a infraestrutura de nuvem por baixo do serviço prestado, exceto algumas configurações da aplicação.

- *Plataform as a Service (PaaS):* Capacidade de prover ao consumidor uma plataforma que permita criação de suas próprias aplicações, usando linguagens de programação, bibliotecas, serviços e ferramentas disponibilizadas pelo provedor. O consumidor não gerencia nem controla a infraestrutura de nuvem por baixo do serviço prestado (servidores, sistemas operacionais, armazenamento, rede, etc.), exceto algumas configurações referentes ao ambiente de hospedagem da aplicação.

- *Infrastructure as a Service (IaaS):* Capacidade de prover ao consumidor meios de provisionar processamento, armazenamento, rede e outros recursos computacionais fundamentais. Nesse modelo o consumidor é capaz de rodar qualquer software, ou seja, tem liberdade de instalar qualquer sistema operacional e aplicação. O consumidor não gerencia nem controla a infraestrutura de nuvem por baixo do serviço prestado, mas tem controle sobre o sistema operacional, armazenamento, aplicações instaladas e limitado controle sobre os componentes de rede.

Um sistema de computação em nuvem pode ser implantado de forma privada ou hospedado nas instalações de um consumidor, pode ser compartilhado entre um número determinado de parceiros, pode ser hospedado por terceiros ou pode ser um serviço publicamente acessível. Dependendo do tipo de implantação, a nuvem pode ter recursos limitados de computação privada ou pode ter acesso a grandes quantidades de recursos acessados remotamente. Os diferentes modelos de implantação apresentam vantagens e desvantagens em como os consumidores podem controlar seus recursos, escala, custo e disponibilidade [14]. Diante disso, o NIST [13] definiu os seguintes modelos de implantação para computação em nuvem:

- *Nuvem privada:* A infraestrutura da nuvem é provisionada para uso exclusivo de uma única organização, abrangendo múltiplos consumidores. Pode ser própria, gerenciada e operada pela organização, por um terceiro ou por uma combinação entre os dois. Além disso pode estar instalada dentro ou fora das dependências da organização.

- *Nuvem comunitária:* A infraestrutura da nuvem é provisionada para uso exclusivo de uma comunidade de consumidores de organizações que tem interesses compartilhados (exemplos: órgãos da justiça ou de ensino superior). Pode ser própria, gerenciada e operada por uma ou mais organizações da comunidade, por um terceiro ou por uma combinação entre os dois. Além disso pode estar instalada dentro ou fora das dependências da organização.

- *Nuvem pública:* A infraestrutura da nuvem é provisionada para uso do público em geral. Pode ser própria, gerenciada e operada por alguma empresa ou governo ou combinação entre os dois. Fica instalada nas dependências do provedor do serviço.

- *Nuvem híbrida:* A infraestrutura da nuvem é composta de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública) que se comportam como uma entidade única, sendo integradas por tecnologias padronizadas ou proprietárias que permitem portabilidade de dados e aplicações.

Hoje em dia a computação em nuvem é basicamente um modelo baseado na Internet e no compartilhamento de softwares e hardwares para suprir as necessidades computacionais (processamento, memória, armazenamento e rede) de usuários e dispositivos. Sua utilização é interessante porque os usuários não precisam considerar os detalhes da infraestrutura na nuvem e nem ter conhecimento profissional ou controle direto sobre a mesma [15].

2.1.2 Virtualização

O paradigma da computação em nuvem oferece vários benefícios por meio da utilização das tecnologias de virtualização [16]. A virtualização permite a abstração e o isolamento de funcionalidades de baixo nível e hardware subjacente. Isso permite a portabilidade das funcionalidades de alto nível e compartilhamento e/ou agregação de recursos físicos [17]. É extremamente adequada para ambientes dinâmicos de computação em nuvem pois provê importantes vantagens de compartilhamento, gerenciamento e isolamento [18]. Dentre as tecnologias de virtualização existentes pode-se citar a virtualização de hardware, virtualização de software, virtualização de memória e virtualização de armazenamento [15].

2.1.3 Virtualização de Servidores

Dentre as tecnologias de virtualização de hardware a mais popular é a virtualização de servidores. Essa tecnologia abstrai o acoplamento entre hardware e sistema operacional, abstraindo os recursos lógicos dos recursos físicos subjacentes visando prover agilidade, flexibilidade e redução de custos. Em um ambiente de servidores virtualizados, recursos computacionais podem ser dinamicamente criados, expandidos, diminuídos e movidos [18]. Em uma interpretação comum, a virtualização de servidores faz o mapeamento dos recursos de um servidor físico em múltiplas representações lógicas, conforme ilustrado na Figura 2.1. Como pode ser observado na ilustração, a arquitetura convencional permite a existência de apenas um sistema operacional por servidor, e na maioria das implantações também apenas uma aplicação. Na arquitetura virtualizada o sistema operacional é substituído por uma camada de virtualização que permite que vários sistemas operacionais diferentes compartilhem um mesmo servidor.

O uso de virtualização é sempre uma relação de compromisso entre a proximidade do controle e o acesso ao hardware real subjacente, o desempenho oferecido ao software hospedado e a flexibilidade do desenvolvimento e implantação [16].

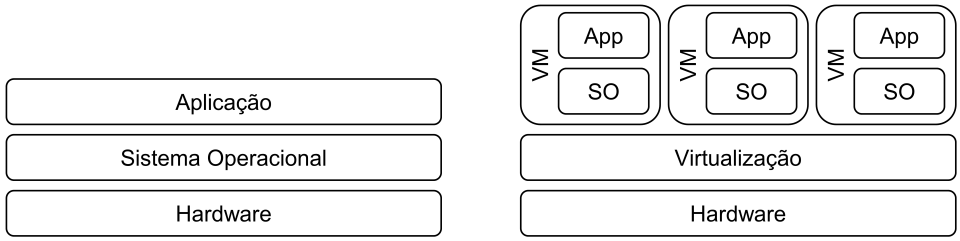

Figura 2.1: Arquitetura Convencional e Arquitetura Virtualizada.

As tecnologias de virtualização oferecem aos aplicativos uma visão abstrata dos recursos de hardware subjacentes, possibilitando os seguintes benefícios para a computação em nuvem [16]:

- *Execução isolada:* O hipervisor realiza a proteção entre máquinas virtuais e, portanto, entre aplicativos executando em diferentes máquinas virtuais. Os usuários podem ter privilégios em sua máquina virtual sem comprometer o isolamento ou a integridade dos recursos físicos.
- *Provisionamento especializado:* A virtualização permite o provisionamento de ambientes altamente especializados e personalizados com finalidades específicas que podem conter sistemas operacionais, bibliotecas ou ambientes de execução. Como a virtualização oferece isolamento funcional, possibilita várias visualizações sobre os mesmos recursos físicos.
- *Gerenciamento facilitado:* Ambientes de execução customizados podem ser iniciados, migrados ou desligados de forma flexível dependendo das necessidades de quem fornece os recursos físicos subjacentes.
- *Legado:* Permite a coexistência de aplicativos legados com novos aplicativos. As máquinas virtuais ajudam a preservar a compatibilidade de binários nos ambientes de execução das aplicações legadas.
- *Teste e depuração:* O teste e depuração de aplicativos paralelos se beneficiam dos ambientes virtualizados, pois um sistema completo pode ser emulado em um único servidor físico.
- *Hipervisores:* Os hipervisores e seus recursos de migração em tempo real permitem aumentar a confiabilidade dos aplicativos virtualizados hospedados, tornando-os independentes dos recursos físicos subjacentes, de maneira transparente aos aplicativos.

A virtualização de servidores também oferece benefícios não relacionados a computação em nuvem, tais como [19]:

- *Redução de custos:* A implantação da virtualização de servidores nas organizações resulta em redução do custo total de propriedade (TCO) e maior retorno sobre o investimento (ROI). Em outras palavras, as organizações conseguem mais recursos computacionais com menos recursos financeiros. Nesse sentido, a virtualização de servidores pode ser uma tecnologia adequada para organizações que buscam soluções para reduzir custos financeiros. A redução pode inclusive abranger os custos da instalação física do *datacenter*, custos com eletricidade, custos com resfriamento, entre outros.

- *Sustentabilidade ambiental:* A virtualização de servidores ajuda a reduzir as demandas de energia e resfriamento decorrentes da consolidação de servidores de *datacenters*, reduzindo indiretamente emissões de carbono no meio ambiente.

- *Alta disponibilidade:* A alta disponibilidade é crucial para as aplicações de missão crítica das organizações. O uso da virtualização de servidores nas organizações permite soluções de rápida recuperação no caso de grandes falhas.

- *Recuperação de desastre:* Planos de recuperação de desastres são importantes para as organizações visando não ocorrer perdas financeiras em casos de interrupções dos serviços de tecnologia. Com a virtualização de servidores é possível que as organizações projetem servidores virtuais em caso de desastre usando soluções de recuperação. Podem ser configuradas para serem executadas no modo de replicação ou *failover*. Quando uma falha grave ocorre, a solução alterna do modo de replicação para o modo de *failover*.

- *Ocupação de espaço físico:* O espaço físico é outra questão importante que as organizações enfrentam devido ao aumento de infraestruturas de tecnologia da informação, tais como servidores. Como a virtualização de servidores ajuda as organizações a consolidar seus servidores físicos em poucas unidades, é possível liberar espaço físico no *datacenter* para outras finalidades.

- *Agilidade:* A virtualização de servidores aumenta a agilidade dos negócios, pois as máquinas virtuais podem ser movidas com rapidez e facilidade em ambientes virtuais. A agilidade é um dos requisitos fortes para cumprir os acordos de nível de serviço nas operações de negócios.

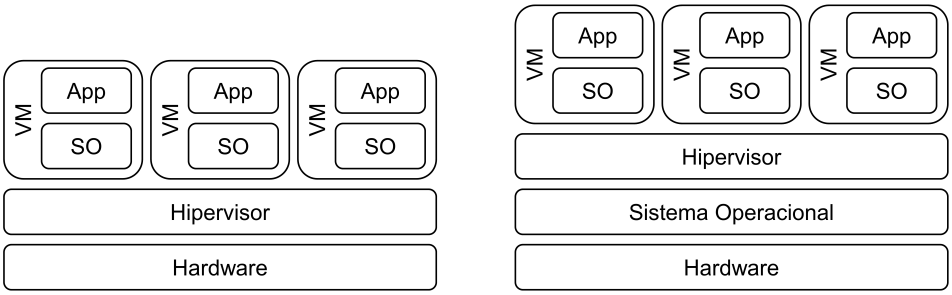

Figura 2.2: Hipervisores Tipo 1 e Hipervisores Tipo 2.

2.1.4 Hipervisores

Os monitores de máquinas virtuais (*Virtual Monitor Machine* – VMM), também conhecidos como hipervisores (*hypervisors*), são implementados como uma camada de software entre o hardware e o sistema operacional, oferecendo uma máquina virtual para outro sistema operacional [20]. É uma tecnologia popular para virtualização de sistemas, visto que é possível obter alta flexibilidade na forma como os recursos virtuais são definidos e gerenciados [16].

Os hipervisores são a plataforma básica para servidores virtuais. Suas principais funções consistem no escalonamento de tarefas, gerência da memória e manutenção do estado do servidor virtual. O desempenho e a escalabilidade do hipervisor define a qualidade da virtualização. Pode-se citar como características necessárias a um hipervisor a segurança sobre os recursos virtualizados e agilidade na reconfiguração de recursos computacionais, sem interromper as operações das máquinas virtuais [20].

Existem duas técnicas usadas nos hipervisores: virtualização total e paravirtualização. A diferença essencial é a necessidade do sistema operacional convidado ser modificado (paravirtualização) ou não (virtualização total) para executar sob o hipervisor.

A virtualização completa realiza toda a abstração do sistema físico sobre o qual o sistema operacional convidado é executado. Não é necessário fazer qualquer modificação no sistema operacional convidado ou em suas aplicações. Esse tipo de virtualização facilita a migração de máquinas virtuais entre servidores físicos, porque há total independência dos recursos físicos do servidor. Por esse motivo a segurança é facilitada devido ao isolamento entre as máquinas virtuais, pois cada instância da máquina virtual é um processo diferente executando no hipervisor. [20]. Uma desvantagem da virtualização completa é o desempenho, pois o hipervisor verifica a execução de todas as instruções feitas pelo sistema operacional convidado e as substitui por ações equivalentes no servidor físico.

A paravirtualização é uma alternativa para contornar os problemas de desempenho e subutilização de recursos da virtualização total. Nesse caso o sistema operacional con-

vidado é alterado para chamar o hipervisor somente quando for executar uma instrução privilegiadas. As instruções não privilegiadas são aquelas realizadas pelos processos de usuários e que podem ser executadas diretamente sobre o processador nativo. Hipervisores que empregam paravirtualização permitem que as máquinas virtuais usem os drivers de dispositivos físicos reais sob o controle do hipervisor, o que otimiza o desempenho. A principal desvantagem da paravirtualização é a necessidade de modificação do sistema operacional convidado [20].

Os hipervisores também podem ser divididos em relação a maneira que acessam os recursos físicos do servidor[16] (Figura 2.2):

- *Tipo 1 (bare metal):* É executado diretamente no hardware físico, virtualizando os dispositivos e oferecendo partições isoladas independentes. Também fornece serviços básicos para controle e comunicação entre partições. Os hipervisores do tipo 1 são adequados para sistemas de tempo real, uma vez que os ambientes virtuais estão próximos do hardware e podem inclusive usar os recursos de hardware diretamente.
- *Tipo 2 (hosted):* É executado em cima de um sistema operacional. São considerados hipervisores hospedados, pois são executados em um ambiente de sistema operacional convencional. A camada de hipervisor normalmente é um nível de software diferenciado no topo do sistema operacional (que é executado diretamente acima do hardware) e o sistema operacional convidado é executado em um nível diferente.

2.1.5 Containers

Os *containers* são uma tecnologia que permite empacotar e isolar aplicações dentro de um sistema operacional, ou seja, possibilita rodar múltiplos espaços de usuários isolados em um mesmo *kernel* [21]. Por esse motivo também é conhecido como "virtualização do sistema operacional". Permite que vários aplicativos sejam executados isoladamente em um sistema operacional, o que o torna semelhante à virtualização de servidores. [22]

Seu foco é isolar processos e recursos em vez de emular todo um servidor físico. Do ponto de vista de uma aplicação, cada *container* é um sistema operacional completo. *Containers* rodam diretamente no sistema operacional compartilhando seus recursos de forma otimizada. Por serem mais otimizados em relação a servidores virtuais, são leves e iniciam rápido [21].

Containers podem ser vistos como ferramentas mais flexíveis para empacotamento, entrega e orquestração de serviços e aplicativos de software. Por serem uma tecnologia recente, permitem melhor portabilidade e interoperabilidade [23], o que vêm contribuindo para uma mudança substancial nos métodos de desenvolvimento e publicação de aplicativos [22].

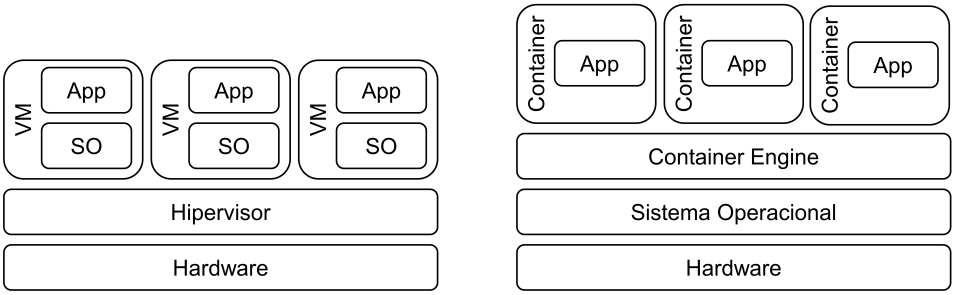

Figura 2.3: Virtualização de Servidores e Virtualização de Sistemas Operacionais.

Um *container* encapsula um aplicativo em um sistema de arquivos completo que contém tudo o que precisa para ser executado: códigos, *runtimes*, ferramentas, bibliotecas. Ou seja, qualquer item que precise ser instalado em um servidor. Ao encapsular e isolar tudo em um *container*, garante-se que o *container* sempre seja o mesmo, independente do ambiente em que está sendo executado [24]. O isolamento da execução do *container* é garantido pela tecnologia *namespaces* e o controle dos recursos utilizados por cada *container* é controlado pela tecnologia *cgroups*, sendo que ambas tecnologias são providas pelo sistema operacional Linux [24]. Pelo fato dos *containers* se basearem em tecnologias exclusivas dos sistemas operacionais do tipo Unix, as implantações são limitadas à sistemas operacionais desse tipo. Quando *containers* são utilizados sobre o sistema operacional Windows, se faz necessário utilizar servidores virtuais com Linux [25].

Containers rodam em qualquer lugar onde o "*Container Engine*" esteja instalado: diretamente no servidor físico (*bare metal*), em servidores virtuais ou diretamente na nuvem. Essa flexibilidade permite a portabilidade entre ambientes diferentes sem precisar reconfigurar a aplicação.

Um comparativo entre *containers* e virtualização de servidores pode ser observado na Figura 2.3. Enquanto na virtualização de servidores cada sistema operacional funciona de forma separada e independente, nos *containers* apenas os aplicativos são isolados, o que os tornam leves e otimizados, cabendo ressaltar que *containers* não são versões leves de servidores virtuais.

Virtualização de servidores e *containers* são arquiteturas completamente diferentes, entretanto, não são mutualmente exclusivas. *Containers* podem ser executados dentro de servidores virtuais sem dificuldade, a depender da solução adotada [24]. Ao utilizar *containers* em uma infraestrutura de virtualização existente, aumenta-se a densidade das cargas de trabalho e por consequência otimiza-se a infraestrutura. Na Figura 2.4 podem ser observadas duas soluções que utilizam simultaneamente hipervisores e *containers*. Na primeira os *containers* são executados diretamente no hipervisor e no segundo os *contai-*

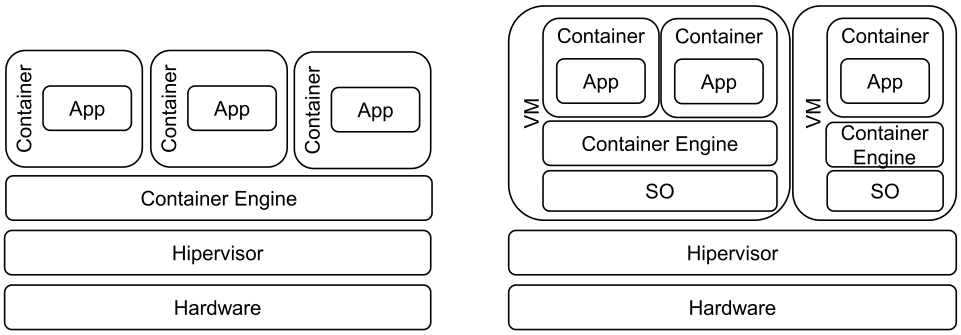

Figura 2.4: Hipervisores e containers trabalhando simultaneamente.

ners são executados dentro um servidor virtual, que por sua vez está sendo executando em um hipervisor.

2.1.6 Sistemas de Arquivos Distribuídos

A constante necessidade de maior poder de armazenamento dos atuais sistemas computacionais exige um sistema de arquivos que seja compartilhado entre diversos nós computacionais. Esses sistemas de arquivos permitem acesso a arquivos a partir de diferentes servidores, possibilitando alta capacidade para aplicações com significativos requisitos de armazenamento [4].

Nesse aspecto, aplicações de computação em nuvem ou virtualização que possuem grandes demandas por armazenamento criaram a necessidade de modificações no modelo tradicional de infraestrutura de armazenamento. No modelo tradicional, conhecido como "*scale-up*" o aumento da capacidade de armazenamento é alcançado adicionando mais recursos a um sistema físico. No modelo modificado, conhecido como *scale-out* o aumento na capacidade de armazenamento é alcançado conectando novas entidades que trabalham juntas para formar uma única unidade lógica [4].

Sistemas de arquivos distribuídos proveem uma solução para as limitações do modelo *scale-up*, permitindo que um único sistema de arquivos permeie múltiplos servidores, seja expandido de forma transparente e aparente ser uma única unidade de armazenamento de alta capacidade. Caso deseja-se prover maior disponibilidade para os arquivos armazenados, o uso da replicação de dados pode ser empregado.

Um sistema de arquivos distribuído realiza o compartilhamento de arquivos de forma persistente em um conjunto de nós conectados por uma rede. Os usuários do sistema de arquivos podem ficar dispersos fisicamente na rede, compartilhando o uso do sistema de arquivos comum. A comunicação entre os nós usa o método RPC (*Remote Procedure Call*) por ser independente dos sistemas operacionais e rede subjacentes [26].

A habilidade de usar hardware comum é uma característica dos sistemas de arquivos distribuídos, pois permite aumentar a escala de maneira simples e econômica de acordo com a demanda dos usuários. A vantagem dessa característica é que quando os nós sofrem melhorias, o desempenho do sistema de arquivo distribuído é aprimorado de forma automática e natural, aumentando a rentabilidade da ampliação do sistema. Além disso a escalabilidade pode ser incremental, permitindo adicionar dispositivos de forma pontual para ampliar a capacidade do sistema [26].

Para superar limitações de processamento nos nós, os sistemas de arquivos distribuídos fazem uso de *caches*, que podem ser posicionados no lado dos nós ou no cliente. Para fornecer uma visão consistente dos dados em todos os clientes e confiabilidade no caso de falhas, as operações de escrita só podem ser concluídas após os dados terem sido transferidos para um armazenamento estável. Dessa forma, uma parte do processamento do nó fica comprometida com as gravações sendo realizadas no sistema de arquivos distribuído. Nesse caso permitir que o processador escreva dados diretamente em um *cache*, e posteriormente esses dados sejam gravados em um armazenamento estável, permite reduzir a processamento decorrente de uma carga de escrita nos nós. Embora o armazenamento em *cache* de dados em clientes locais melhore o desempenho, muitas operações de arquivos ainda usam trocas de mensagens síncronas entre cliente e nós para manter a consistência do *cache* e proteger contra falhas [26].

A transparência e redundância é obtida ao permitir que dados distribuídos em vários nós diferentes sejam agrupados logicamente. Por esse motivo o desempenho dos sistemas de arquivos distribuídos pode ser baixos em comparação com os sistemas de arquivos locais, devido a execução de operações de I/O síncronas para coerência de cache e segurança de dados [26].

Outra questão importante acontece quando a falha de um nó não pode ser distinguida de uma falha de comunicação, ou de respostas lentas devido a uma sobrecarga. Como o nó não responde, não é possível determinar se houve falha que interrompeu o processamento ou se a comunicação falhou, mas o sistema ainda está operando. Nesse último caso, deve-se supor que o sistema inacessível ainda é capaz de processar solicitações de arquivos. O sistema de arquivos distribuído deve lidar com este caso de forma que a consistência e as garantias semânticas do sistema não sejam violadas [26].

Os conceitos de dados e metadados são importantes em sistemas de arquivos distribuídos, e diferentes sistemas podem ser categorizados pela maneira que os dados e metadados são tratados. Dados são o conteúdo de arquivos. Metadados são atributos representados como pares (atributo, valor) e contém informações sobre o arquivo tais como tamanho, dono, data, hora, etc. Na maioria dos sistemas de arquivos distribuídos, dados e metadados são armazenados e gerenciados separadamente. No entanto, existem sistemas de

arquivos que armazenam e gerenciam dados e metadados juntos [27].

Um arquivo é composto de dados e metadados, enquanto um diretório é somente um metadado. Armazenar dados e sua hierarquia é possível através de um diretório. Cada diretório contém entradas de arquivos ou diretórios e seus metadados. A estrutura de diretórios aninhada é chamada de *namespace* (espaço de nome). Gerenciar *namespaces* ou diretórios é equivalente de gerenciar metadados para responder adequadamente às solicitações do sistema de arquivos. Selecionar um método de indexação apropriado é um dos processos mais importantes para melhorar o acesso aos metadados. Essa indexação é mais importante em sistemas de arquivos distribuídos, porque o método de separar metadados pode ter um grande impacto no tempo de resposta das solicitações, de acordo com a latência da rede [27].

2.1.7 Sistemas de Arquivos Paralelos

Sistemas de arquivos paralelos são um caso especial de sistema de arquivos distribuídos, no qual os arquivos são armazenados e acessados a partir de vários servidores. Armazenar um arquivo em n nós, onde os nós podem ser acessados simultaneamente, teoricamente permite uma velocidade n vezes maior do que em um único nó [5].

Nesse sistema os blocos de dados são divididos, paralelamente, em vários os servidores de armazenamento. O suporte ao paralelismo permitir que diversos clientes acessem arquivos a partir de diversos nós, com o sistema de arquivos gerenciando a concorrência na leitura e escrita [26].

2.1.8 Redes de Armazenamento

O crescimento da quantidade de dados nas organizações trouxe à tona a necessidade da utilização de unidades de armazenamento independentes do servidor. A maneira de conectar a unidade de armazenamento evoluiu de soluções nas quais a unidade era conectada diretamente ao servidor (DAS - *Direct Attached Storage*), para a criação de redes de armazenamento independentes, que usam uma combinação de protocolos específicos e interfaces de discos, denominadas redes de armazenamento SAN (*Storage Area Network*) e NAS (*Network Attached Storage*) [20].

A principal diferença entre SAN e NAS é que enquanto uma SAN oferece apenas um meio de armazenamento formado por blocos, sem criar um sistema de arquivos para eles, uma NAS fornece, além do meio físico de armazenamento, um sistema de arquivos. Apesar disso, SAN e NAS podem ser empregadas concomitantemente em uma única solução de armazenamento, ou seja, uma parte da unidade de armazenamento pode ser configurada para oferecer blocos para dados e outra para blocos de dados e um sistema de arquivos.

Uma unidade armazenamento do tipo NAS é baseada em redes IP e é primariamente utilizada para compartilhamento de arquivos. Quando comparado ao DAS é mais escalável, com melhor disponibilidade, além de mais fácil de gerenciar. Normalmente, o seu uso e gerenciamento requerem maior investimento inicial e conhecimento mais especializado.

As NAS usam protocolos de rede e de compartilhamento de arquivos. Esses protocolos incluem o TCP/IP para a transferência de dados e CIFS e NFS para servir arquivos remotamente. Usuários do Windows e do Unix podem compartilhar os mesmos dados armazenados em um servidor NAS, que é acessado por clientes e servidores em uma rede IP, e, muitas vezes, utiliza múltiplas interfaces de rede [20].

O armazenamento do tipo SAN é baseado em redes de armazenamento dedicadas e escaláveis, que conectam servidores e dispositivos de armazenamento usualmente no nível de bloco (dados de aplicação).

Nas redes SAN, a tecnologia de rede pode ser *Fibre Channel* (FC) ou *Ethernet*, enquanto os dados transportados são do tipo "bloco". Nas redes NAS, a tecnologia é quase sempre *Ethernet*, e os dados armazenados, do tipo "arquivo". O entendimento de quando utilizar uma ou outra infraestrutura é complexo e, muitas vezes, confuso, causando problemas quando soluções que deveriam ser baseadas em NAS são feitas em SAN e vice-versa. O protocolo SCSI continua sendo o padrão utilizado na comunicação entre o servidor e a unidade armazenamento. Na prática, protocolos como FC e iSCSI encapsulam os comandos SCSI dentro deles [20].

2.2 Desafios do Armazenamento de Dados na Computação em Nuvem

Essa seção apresenta tópicos que tratam de alguns dos desafios do armazenamento de dados na nuvem.

2.2.1 Armazenamento de Dados em Containers

Um *container* por definição é uma entidade efêmera, ou seja, todos os dados trabalhados pelo *container* durante sua execução são perdidos ao término da execução. Por esse motivo *containers* são ideais para aplicações sem estado (*stateless*), ou seja, aplicações que não salvam dados das sessões. Entretanto existem mecanismos para assegurar a escrita persistente de dados em um *container*.

O armazenamento de dados em uma infraestrutura de *containers* é dividido em dois tipos: "armazenando de imagens e *containers*" e "armazenamento persistente". O armazenamento de imagens e *containers* é um recurso da infraestrutura destinado a guardar as

imagens (*containers* que não estão em execução) e fornecer uma área de armazenamento para *containers* em execução. Entretanto para o *container* em execução esse armazenamento não tem persistência, e todo dado gravado nessa área é perdido após término da execução do *container*. Para suprir esse comportamento existe o armazenamento persistente, que é um local de armazenamento fora do *container* e que persiste mesmo após o seu término [28].

Uma imagem é constituída de uma série de camadas, que interagem entre si, podendo ser compartilhadas entre vários *containers* conforme pode ser visto na Figura 2.5 [28]. Quando uma imagem é executada e se torna um *container*, uma nova camada é criada para permitir leitura e escrita da aplicação contida no *container*, enquanto as demais camadas ficam em somente leitura. Essa arquitetura pode ser observada na Figura 2.6 [28]. Para gerenciar as interações entre as camadas de uma imagem, um sistema de arquivo é montado em cima do sistema de arquivo do *host* ou um dispositivo é alocado exclusivamente para essa finalidade [28].

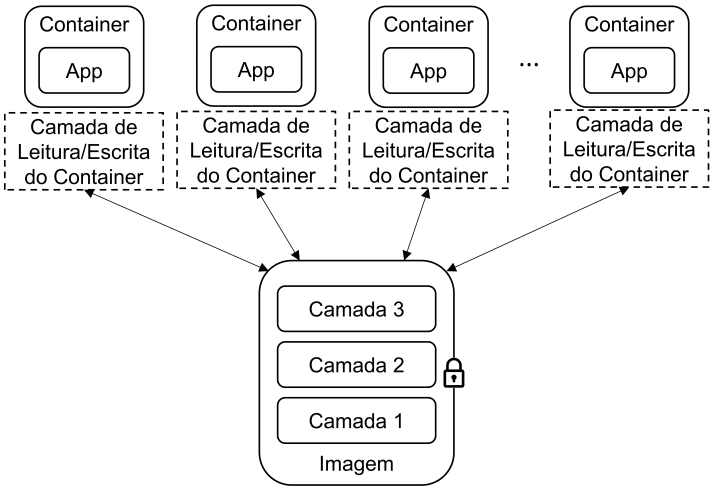

Figura 2.5: Compartilhamento de imagens entre *containers*.

O armazenamento persistente de um *container* pode ser realizado dentro ou fora do *host*. No armazenamento dentro do *host*, quando o *container* acaba, os dados são mantidos no *host*. Porém essa solução não escala, pois os dados podem ser necessários para *containers* em outros *hosts*. No armazenamento externo, quando o *container* acaba, os dados são mantidos em um local externo. Essa solução escala, pois, basta que todos os *hosts* tenham acesso ao local de armazenamento [28].

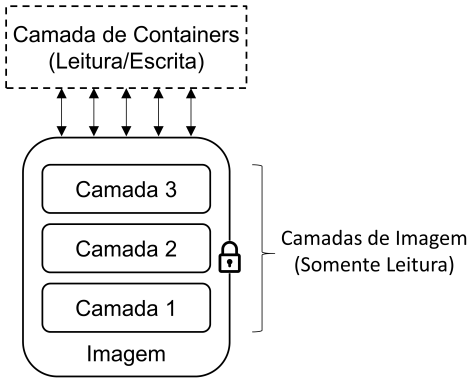

Figura 2.6: Camadas de uma imagem e camada de *container*.

2.2.2 Hiperconvergência

Na última década observa-se que houve uma tendência de convergência entre servidores, rede e gerenciamento. Esse movimento por ser observado no sucesso que os servidores do tipo *blade* fizeram ao integrar em um mesmo equipamento servidores, rede, armazenamento e gerenciamento centralizado. Essa tendência continuou evoluindo ao integrar vários equipamentos no mesmo *rack*, depois vários *racks* e por fim softwares para facilitar a composição de tais sistemas. Essa abordagem se tornou mais atraente à medida que tecnologias mais robustas de software de armazenamento foram aplicadas em *datacenters* visando melhorar escalabilidade e alta disponibilidade [10].

Enquanto essa convergência se limitava a colocar os elementos de computação, rede e armazenamento sob um gerenciamento comum, a hiperconvergência extrai essas funções dessas "ilhas" (computação, armazenamento e rede) e redesenha a arquitetura incluindo o hipervisor, daí o nome hiperconvergência. Sistemas hiperconvergentes são encontrados em dispositivos com recursos de computação, armazenamento, rede e hipervisor aliados à habilidade de aumentar a capacidade de um sistema simplesmente adicionando mais dispositivos, no modelo *scale-out* [10].

A hiperconvergência é um paradigma emergente que vem ganhando popularidade tanto na academia quanto na indústria [12], sendo o resultado da fusão dos componentes de software (computação, armazenamento e rede) no plano de controle e/ou dados. É fundamentada na migração das funções de hardware para software em todos as três "ilhas" que antes eram independentes [10], conforme pode ser observado na Figura 2.7.

Enquanto os tradicionais sistemas de armazenamento utilizam SAN ou NAS com diversos dispositivos físicos de armazenamento agregados em um *rack* controlado por hardware e *firmware* dedicados, um sistema hiperconvergente dispersa o armazenamento entre numerosos servidores físicos com dispositivos de armazenamento diretamente conectados,

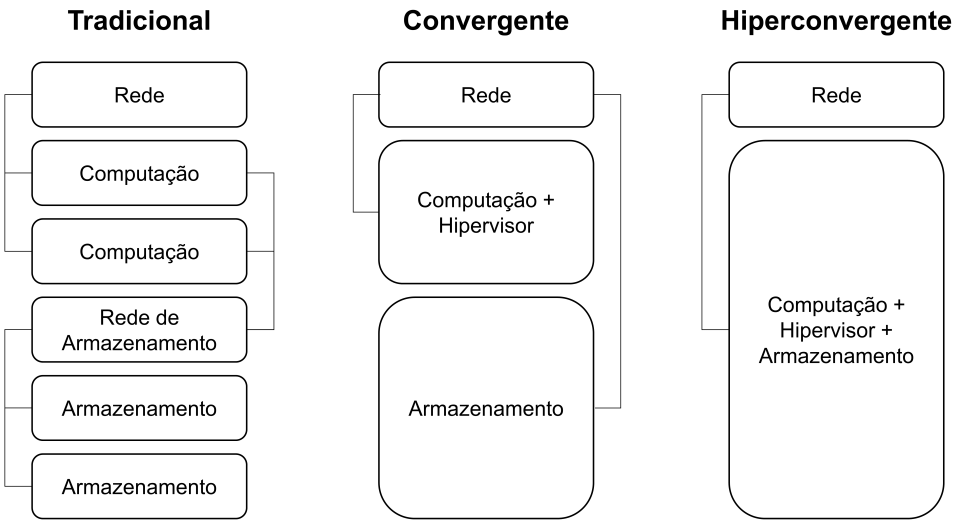

Figura 2.7: Arquitetura Tradicional, Convergente e Hiperconvergente.

agregando a capacidade via software.

Os nós de uma infraestrutura hiperconvergente são geralmente conectados via rede *Ethernet* e por meio de uma camada de software, chamada de VSA (*Virtual Storage Appliance*), que utiliza essa rede para distribuir o armazenamento [12].

Sistemas hiperconvergentes crescem simplesmente ao adicionar mais nós. Ao focar na otimização das unidades de processamento e controladoras de armazenamento, os sistemas hiperconvergentes usam maneiras eficientes de ler e gravar dados em grande escala nos discos dos nós. Assim o paradigma da hiperconvergência gira em torno da eventualidade de aumentar o sistema e suprir uma demanda de armazenamento [12].

Em contraste com as arquiteturas hiperconvergentes, soluções tradicionais de armazenamento não foram projetadas para trabalhar fortemente integradas com recursos de processamento e memória. Diante disso existe uma tendência das soluções tradicionais serem substituídas por sistemas hiperconvergentes usando hardware comum, que são fáceis de implantar e expandir em uma época onde os requisitos de armazenamento crescem em ritmo acelerado [12].

Ao integrar a solução de armazenamento junto ao servidor a latência de acesso aos dados é reduzida em comparação com uma rede de armazenamento (SAN), visto que os dados não precisam percorrer uma rede *Fibre Channel* ou *iSCSI* para chegar as unidades armazenamento.

A hiperconvergência também permite usar o conjunto comum de recursos de processamento para realizar tarefas relacionadas ao armazenamento, enquanto os sistemas de armazenamento centralizado têm recursos de processamento dedicados. Durante uma

ocorrência de uma falha a distribuição da carga de processamento em vários servidores pode acelerar a reconstrução dos dados [10].

Na arquitetura mostrada na Figura 2.8, pode ser visto que os nós hiperconvergentes incluem componentes integrados de computação e armazenamento. Os nós de convergência são conectados a um volume virtual compartilhado que oferece suporte à replicação de dados. Quando um nó precisa ser adicionado, basta que o mesmo tenha conectividade com os demais nós, o que torna a arquitetura flexível e escalável.

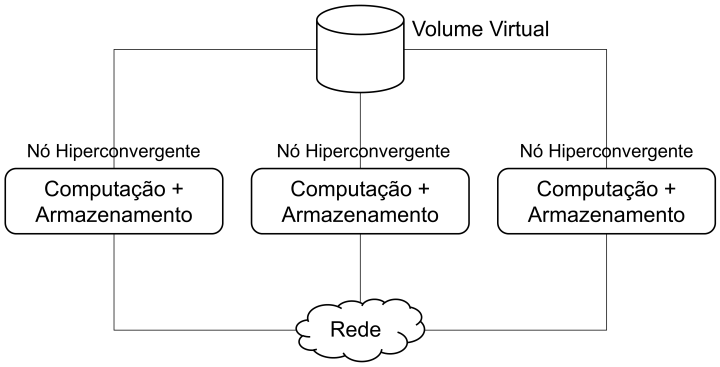

Figura 2.8: Arquitetura Hiperconvergente.

Por padrão um volume virtual de uma solução hiperconvergente tem seus dados replicados em diversos nós. No caso de uma falha em um nó, o acesso aos dados é migrado ineditamente para outro nó, visto que esses são replicados. Dessa forma, os nós hiperconvergentes oferecem um maior grau de tolerância a falhas e recuperação imediata da falha, em contraste com arquiteturas tradicionais de *datacenter*. Por esse motivo e pelo fato de possuir rápida implantação, observa-se uma um aumento no uso dessa arquitetura em *datacenters* desde o ano de 2015 [12].

Em resumo, como os nós de hiperconvergência são fáceis de implantar e ao mesmo tempo oferecem acesso rápido ao armazenamento, alta tolerância a falhas e escalabilidade, o seu uso para hospedagem de *containers* ou servidores virtuais é eficaz.

2.2.3 Armazenamento Definido por Software

"Algo" definido por software (SDx - *Software Defined Anything*) é uma arquitetura para virtualização da rede (SDN - *Software Defined Network*), comunicações (SDR - *Software Defined Radio*), armazenamento (SDS - *Software Defined Storage*), entre outros [29]. Essa arquitetura propõe a formação de uma camada de software entre os hardwares e aplicações, fazendo com que as aplicações abstraiam os recursos físicos.

O armazenamento definido por software é um conceito em evolução para gerenciamento de unidades de armazenamento de dados sob a perspectiva de software, independentemente do hardware subjacente [8]. Esse tipo de armazenamento gerencia o provisionamento de recursos de armazenamento se baseando em políticas, sendo frequentemente usado em ambientes virtualizado para provisionar os recursos necessários.

Tem como objetivo esconder as complexidades do gerenciamento e controle dos recursos físicos de armazenamento. Sua arquitetura propõe o uso de hardware comum sendo gerenciando por software, visando prover armazenamento de padrão empresarial (*enterprise-grade*) com bom custo-benefício [30]. Seus conceitos são similares ao das redes definidas por software (SDN - *Software-Defined Networking*) onde o principal objetivo é fornecer um sistema de armazenamento virtual, abstraindo os serviços e recursos de armazenamento lógico dos sistemas de armazenamento físico subjacentes [8].

Essa arquitetura permite que um provedor de serviços em nuvem utilizem diversos recursos de armazenamento distintos para formar uma única unidade massiva de armazenamento, automatizando e mascarando sua estrutura interna e complexibilidade [31]. Em ambientes de computação em nuvem onde os recursos são compartilhados entre vários consumidores, o sistema de armazenamento definido por software deve prover uma instância particular para cada consumidor, garantindo a separação de dados, o isolamento de desempenho e as garantias de desempenho [8].

Um sistema de armazenamento definido por software tem a responsabilidade de gerenciar grandes sistemas de armazenamento, isolando o plano de controle do plano de dados [32]. O plano de controle é o software que gerencia os recursos de armazenamento, aplicando políticas em unidades de armazenamento diferentes de acordo com os requisitos das aplicações. É a camada mais crítica, pois interage entre a infraestrutura e as aplicações [33]. O plano de dados (ou camada de infraestrutura) são os dispositivos de armazenamento que conservam os dados brutos, enviando, processando e retornando dados armazenados. Uma comparação entre a abordagem tradicional e o uso de armazenamento definido por software pode ser observada na Figura 2.9. Em uma SAN/NFS tradicional a plano de controle e de dados estão fisicamente no mesmo equipamento de armazenamento de dados, enquanto que em um sistema de armazenamento definido por software o plano de controle fica externo aos recursos físicos de armazenamento, executando em um servidor comum.

Dentre as principais características do armazenamento definido por software pode-se citar a escalabilidade, uso de hardware comum, abstração do hardware, gerenciamento centralizado, automatização, programabilidade, orientado a políticas. Suas métricas de desempenho mais comuns são IOPS (*Input/output Operations Per Second*), latência e taxa de transferência.

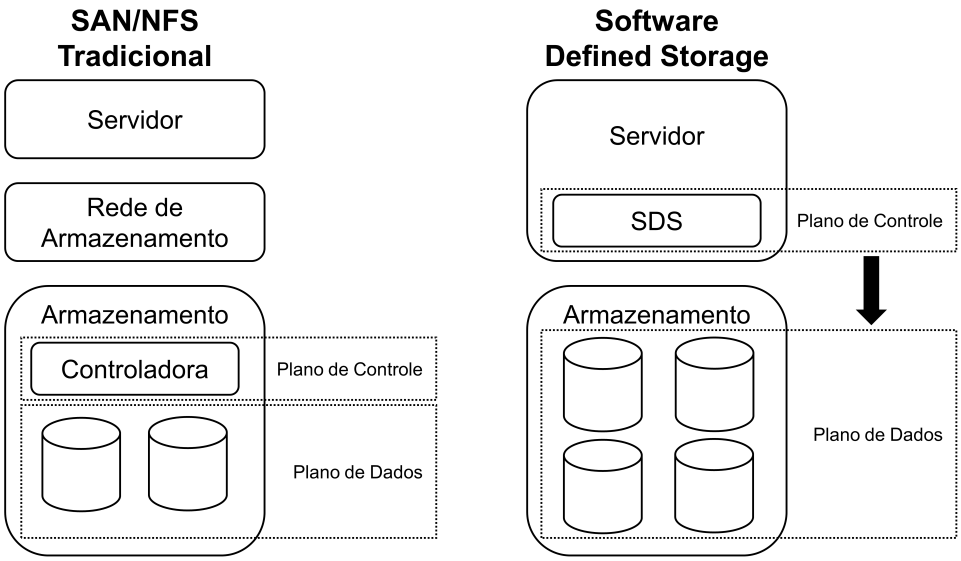

Figura 2.9: Abordagem Tradicional e Definida por Software.

Armazenamento definido por software difere de virtualização de armazenamento, pois o primeiro possui vários componentes, sendo a virtualização de armazenamento um deles. Com uma complexidade reduzida, o armazenamento definido por software simplifica a administração, virtualiza e automatiza os serviços de armazenamento, melhora a utilização dos recursos de armazenamento existentes e facilita a migração de dados entre sistemas [29].

Em ambientes de computação em nuvem os dados dos consumidores podem exigir determinadas políticas de desempenho, a fim de atender requisitos de arquivamento de dados corporativos. Entretanto geralmente os consumidores não têm como especificar seus requisitos ao contratar serviços em nuvem. O uso de armazenamento definido por software nesses casos permite flexibilidade suficiente para suportar essas e outras políticas e restrições, visando garantir o Acordo de Nível de Serviço (SLA) selecionado [8].

Entretanto o padrão oficial ainda precisa ser formado, e os softwares disponíveis ainda são poucos e apropriados para somente alguns tipos de equipamentos [34]. Entre os SDS's comercialmente disponíveis destacam-se o CorpHD [31] que possui código aberto, e os proprietários EMC ViPR [34] e IBM Storwize[33].

Os atuais desafios do armazenamento definido por software são: alocação de dados (bloco, objeto, arquivo), migração de dados, confiabilidade dos dados, ferramentas de comunicação (API), tipo de dados, compressão, manutenção do QoS, gerenciamento de metadados, gerenciamento de falhas e monitoração [32].

2.3 Trabalhos Relacionados

Essa seção apresenta os trabalhos relacionados recentes que tratam dos desafios e definições de diferentes abordagens sobre o armazenamento de dados na nuvem.

2.3.1 Armazenamento de Dados em Containers

Estudos recentes na área de *containers* que abordam a questão do armazenamento de dados focam principalmente na análise das diferentes configurações possíveis para o armazenamento de imagens e *containers*. Em um desses trabalhos [1] analisou-se qual o desempenho dos diferentes "*storage drivers*" usados na infraestrutura de *containers* provida pela ferramenta Docker. *Storage drivers* são sistemas de arquivos montando em cima do sistema de arquivos do *host* que visam o armazenamento de imagens e *containers*, principalmente no que tange a iteração entre as camadas de uma imagem [28]. Nesse estudo [1] uma ferramenta de *benchmark* de sistemas de arquivos foi instalada dentro de um *container* que executou uma carga de trabalho sintética de leitura e escrita em cada um dos *storage drivers*. A conclusão foi de que não existe um *storage driver* com bom desempenho em todos os cenários analisados, sendo que o driver deve ser escolhido de acordo com o padrão de leitura e escrita da aplicação que será empacotada dentro do *container*. Entretanto, nesse trabalho a questão do armazenamento persistente não foi abordada [1].

Em outro trabalho semelhando realizado por [2], a questão da análise do desempenho de diferentes *storages drivers* foi analisada, porém com o foco no uso de unidade de armazenamento de estado sólido (SSD) de alto desempenho instalados no *host*. Nesse estudo tanto o armazenamento de imagens e *container* quanto o armazenamento persistente foram analisados. Em uma análise experimental semelhante a utilizada no trabalho anterior [1], nesse estudo uma ferramenta de *benchmark* de sistemas de arquivos também foi instalada dentro de um *container* que executou uma carga de trabalho sintética de leitura e escrita em cada um dos *storage drivers* em cima do mesmo SSD. Adicionalmente uma unidade de armazenamento persistente foi apresentada para o *container* do experimento e a mesma ferramenta de *benchmark* executou a carga de trabalho no armazenamento persistente. O experimento foi repetido com diferentes *storages drivers*, mantendo-se o mesmo armazenamento persistente. A conclusão foi de que os drivers do tipo "overlay" e "aufs" no geral apresentaram melhor desempenho ao usar um SSD e que as operações no armazenamento persistente não são afetadas pelo *storage driver*.

Em outro trabalho envolvendo o armazenamento de dados em *containers* [3] também é analisado o desempenho usando unidades de armazenamento de estado sólido (SSD) de alto desempenho, tanto localmente quanto remotamente. Diferente dos dois trabalhos

anteriores [1] [2], nesse estudo temos um experimento onde uma unidade externa ao *host* é utilizada para o armazenamento de dados de *containers*, sendo acessada via uma rede de alta velocidade. Nessa configuração foi utilizado o protocolo NVMF (*Non-Volatile Memory express over Fabrics*), utilizado para estender o barramento PCI-e do *host* através de uma rede *Ethernet*. Na análise experimental diversos *containers* com o banco de dados Cassandra foram executados e uma carga de trabalho sintética foi aplicada aos *containers*. O local de armazenamento de dados do Cassandra foi configurado de diferentes maneiras de modo a avaliar o desempenho tanto local quanto remoto. A conclusão foi de que usando essa tecnologia, tanto o armazenamento local quanto remoto obtiveram desempenho próximos.

2.3.2 Hiperconvergência

Os estudos na área de hiperconvergência são escassos na literatura. Dentre poucos estudos destaca-se o que apresenta um cenário de implantação do sistema de arquivos distribuído GPFS da IBM como solução de armazenamento definido por software para sistemas hiperconvergentes [10]. A solução proposta consistem em utilizar o GPFS dentro de uma arquitetura hiperconvergente por meio do uso de *virtual appliances* (servidores virtuais especializados) que gerenciam os recursos físicos de armazenamento (discos) com a habilidade de aumentar a capacidade no modelo *scale-out*. A disponibilidade e proteção dos dados se faz por meio de um mecanismo de software RAID chamado "GNR" (*GPFS native RAID*) que realiza cópias dos dados em outros nós. Apesar do artigo ser produzido pela IBM e apresentar um caso de aplicação de seus produtos, diversos conceitos de hiperconvergência e armazenamento definido por software são bem definidos. A arquitetura apresentada para o uso do GPFS pode ser replicada para o uso em outro sistema de arquivo distribuído de modo a alcançar o mesmo resultado.

Em outro estudo [11] os conceitos de hiperconvergência e caso de uso descritos no trabalho anterior [10] são utilizados para construir uma arquitetura hiperconvergente usando hardware comum e softwares de código aberto para prover infraestrutura a um ambiente de teste para armazenar e processar dados provenientes de dispositivos IoT (*Internet of Things*).

A quantidade de dados que podem ser geradas por dispositivos IoT é novamente abordada em outro trabalho [12], onde após analisar os paradigmas de armazenamento possíveis de serem adotados para guardar e processar dados provenientes de dispositivos IOT aponta a arquitetura hiperconvergente como solução. Apesar do estudo focar na parte de rede e na quantidade de dados que os IoT podem gerar, existe uma discussão detalhada sobre como a hiperconvergência pode atender as necessidades de armazenamento expostas no trabalho. Nesse caso conclui-se que apenas uma infraestrutura hiperconvergente

conseguiria suportar os requisitos de escalabilidade, tolerância a falhas e uso de hardware comum exigidos.

2.3.3 Armazenamento Definido por Software

Os termos "Armazenamento Definido por Software" e "Sistemas de Arquivos Distribuídos" são conceitos semelhantes, por vezes empregados como sinônimos. Um sistema de arquivo distribuído é por definição um software que armazena dados em diversos nós. Um sistema de armazenamento definido por software baseia-se nessa mesma definição, dando ênfase na capacidade de manipulação do fluxo de dados, abstração do hardware, escalabilidade, confiabilidade, tolerância a falhas, interface de gerenciamento e integração com outros softwares [35].

Dessa forma termo "Armazenamento Definido por Software" é uma extensão do termo "Sistemas de Arquivos Distribuídos" [35], pois um sistema de armazenamento definido por software quando implantado em diversos servidores com discos internos exige o uso de um sistema de arquivos distribuído [36]. Essa abordagem na definição também é observada pelo fabricante de dispositivos de armazenamento NetApp [37], que considera os sistemas de arquivos distribuídos como um dos tipos de armazenamento definido por software disponíveis. Ou seja, o armazenamento definido por software move a infraestrutura e o plano de controle de uma plataforma baseada em hardware para uma arquitetura baseada em software, usando uma camada de abstração para separar "dados" dos "discos subjacentes", empregando o uso um sistema de arquivos distribuído para permitir uma arquitetura *scale-out* abrangendo vários nós de armazenamento [38].

Na área de armazenamento definido por software o trabalho mais frequentemente citado propõe-se uma arquitetura de armazenamento definido por software que permite habilitar politicas fim-a-fim nos fluxos de I/O no disco de servidores virtuais [7]. A arquitetura proposta funciona no hipervisor e no servidor de armazenamento, interceptando o fluxo de dados entre o servidor virtual e o armazenamento físico. Para isso é criada uma camada de software entre o servidor virtual e o armazenamento físico (plano de dados) e também é criado um plano de controle centralizado que permite diferenciação no tratamento de I/Os com suporte a chamadas API, trabalhando separado do hipervisor e unidade de armazenamento.

Em estudo recente [8] o armazenamento definido por software é utilizado para gerenciar os recursos de armazenamento de serviços de computação em nuvem. Nesse trabalho é proposto um algoritmo que monitora o espaço em disco dos recursos de armazenamento de um cliente da nuvem e quando esse atinge um limiar determinado de uso, um novo disco é criado e os recursos são balanceado entre os discos antigo e novo.

Outro trabalho recente na mesma linha utiliza o armazenamento definido por software para definir e aplicar políticas de desempenho baseadas no perfil da carga de trabalho em serviços de armazenamento em bloco [9]. Nesse estudo um escalonador é proposto visando fornecer qualidade de serviço para armazenamento em bloco para servidores virtuais. O escalonador recebe dados de desempenho de disco vindo de agentes instalados em servidores virtuais, verifica se o os requisitos de qualidade estão sendo atendidos.

Uma observação frequente nos trabalhos relacionados ao armazenamento definido por software [7] [8] [9] é que o foco é basicamente aproveitar o desacoplamento entre o plano de dados e o plano de controle para manipular os fluxos de dados no plano de controle visando algum objetivo. Em um dos trabalhos [7] a manipulação visou estabelecer qualidade de serviço, em outro [8] a manipulação visou otimizar a utilização dos recursos de armazenamento físico e em outro [9] a manipulação visou prover políticas de desempenho baseadas no perfil da carga de trabalho.

Em mais um trabalho recente nessa área há uma comparação entre os sistemas de arquivos distribuídos para armazenar cálculos científicos [4]. Nesse estudo foram testados o GlusterFS e Ceph, utilizado 8 nós distribuídos em 8 combinações distintas e a análise de desempenho realizada pela ferramenta FIO, um gerador de I/O. Foram realizados testes com 7 padrões distintos de escrita, leitura e tamanho. A conclusão do estudo foi que o GlusterFS superou o Ceph durante cargas de trabalho com grande quantidade de I/O e que a administração do GlusterFS é simplificada, exigindo menos tempo de manutenção e instalação. As conclusões sobre o Ceph foram que esse possui complexidade maior e precisa de servidores separados para metadados, o que aumenta a complexidade da solução.

Outro estudo em uma área de pesquisa semelhante analisou o problema de como armazenar e transferir grandes quantidades de dados científicos entre duas instituições [5]. Nesse estudo verificou-se que a melhor maneira de transferir dados entre duas instituições de pesquisa é usando uma combinação entre uma rede de comunicação de alta velocidade e um sistema de arquivos de alto desempenho. Encontrando um ponto de equilíbrio entre esses dois itens possibilita que os dados consigam ser lidos na origem em alta velocidade, transmitidos rapidamente e por fim gravados em alta velocidade no destino. Os sistemas de arquivos distribuídos analisado nesse estudo foram BeeGFS, Ceph, GlusterFS e OrangeFS. Foram realizados testes com 4 a 8 servidores na origem transmitindo dados via FTP para outros 4 a 8 servidores no destino, com o BeeFS obtendo melhor desempenho.

2.3.4 Discussão sobre os Trabalhos Relacionados

Em trabalho recente é apresentada uma implantação de armazenamento definido por software em um ambiente heterogêneo [34]. Como prova de conceito os sistemas de arqui-

vos distribuídos HDFS, Ceph e GlusterFS foram integrados um sistema de gerenciamento de serviços em nuvem executando OpenStack. Nesse estudo o software EMC ViPR foi utilizado para gerenciar um conjunto físico de recursos de armazenamento. Esses recursos foram então disponibilizados para o OpenStack visando a criação de servidores virtuais. Foram criados conjuntos de servidores virtuais, sendo que cada conjunto utilizou um sistemas de arquivos diferente. Em seguida foram conduzidos experimentos de download e upload de arquivos para esses sistemas de arquivos.

Em outro trabalho semelhante o armazenamento definido por software foi utilizado em uma arquitetura com sistemas de arquivos distribuídos heterogêneos, onde o uso de um plano de controle em software permitiu a implantação de um mecanismo de partição e distribuição de arquivos entre os diferentes tipos de sistemas de aquivos [6].

Apesar da queda recente do uso de servidores virtuais nos ambientes de computação em nuvem, a virtualização de servidores pode ser aproveitada em uma infraestrutura de *containers* de forma a otimizar ainda mais recursos computacionais. Inclusive esse fato foi observado em pesquisa recente [39], onde os entrevistados foram perguntados sobre qual seria a infraestrutura escolhida para hospedar seus *containers*. Nessa pesquisa, 52% dos entrevistados responderam que utilizariam a virtualização de servidores como plataforma para uma infraestrutura de *containers*, 28% responderam que utilizariam diretamente um serviço em nuvem e 20% responderam que utilizariam servidores físicos (*bare metal*).

Em relação aos estudos recentes que abordam a questão do armazenamento de dados de *containers* [1] [2], verifica-se que o foco é o armazenamento local, com exceção do trabalho de [3] onde também é analisada uma forma muito específica de armazenamento externo.

Diante do exposto verifica-se que existem poucos estudos na área de sistemas de arquivos distribuídos com foco em prover armazenamento com desempenho, confiabilidade e baixo custo para serviços em nuvem, especificamente para o armazenamento de dados de *containers*. Menos estudos ainda são encontrados na área de hiperconvergência. Entretanto na área de armazenamento definido por software os problemas relacionados à servidores virtuais é tratado com mais frequência do que os relacionados a *containers*, geralmente abordando o tema de interceptar e manipular as operações de disco para alguma finalidade.

2.4 Conclusão do Capítulo

Este capítulo apresentou a revisão da literatura e descreveu os conceitos que serão usados na solução proposta nesse trabalho. Itens relacionados a computação em nuvem foram descritos, assim como os componentes que serão foco da solução: *containers*, hiper-

convergência e armazenamento definido por software. Foi também realizada uma revisão bibliográfica dos temas relacionados a este trabalho, assim como o posicionamento da proposta deste trabalho em relação aos que trabalhos que envolvem soluções de armazenamento de dados para *containers*.

Capítulo 3

Proposta

Este trabalho tem como objetivo propor uma arquitetura hiperconvergente de *containers* que usa um sistema de armazenamento definido por software. Essa arquitetura pode ser implantada em um provedor de serviços em nuvem com o objetivo de consolidar os recursos de armazenamento disponíveis em diversos servidores dispersos em um *datacenter*, de forma que os recursos se comportem com um único volume massivo, escalável e tolerante a falhas. Esse volume é então apresentado de volta para cada um dos servidores que cederam recursos, provendo-os com uma unidade de armazenamento de alta capacidade e redundante. A solução propõe a construção de um "servidor virtual de armazenamento", que desse ponto para frente será chamado de VSA (*Virtual Storage Appliance*), que gerenciará os recursos físicos de armazenamento do servidor (discos) com um sistema de arquivos distribuído, criará volumes distribuídos de armazenamento que serão usados de forma hiperconvergente pelos *containers* para armazenamento persistente de dados. A contribuição desse trabalho é analisar a viabilidade de uso dessa proposta de infraestrutura hiperconvergente, usando software livre e hardware comum de mercando, em um provedor de serviços em nuvem para fornecer serviços de IaaS (tais como containers e servidores virtuais) e PaaS (aplicações empacotadas, tais como banco de dados e servidores web).

3.1 Arquitetura da Solução Proposta

A arquitetura proposta é apresentada na Figura 3.1 e funciona resumidamente da seguinte maneira:

1. O servidor físico roda um hipervisor do tipo I com pelo menos duas controladoras de discos: uma para o hipervisor, VSA e *Container Host* ❶; e outra com os recursos de armazenamento para o sistema hiperconvergente ❷.

2. Nos discos conectados na controladora gerenciada pelo hipervisor (caminho ❸) estão armazenados os arquivos do próprio hipervisor, o servidor virtual do VSA e o servidor virtual do *Container Host*, conforme podem ser visto nos caminhos ❺.

3. O hipervisor por meio da funcionalidade de "*PCI Passthrough*" repassa o gerenciamento da controladora de discos ❷ para o VSA pelo caminho ❹. Dessa maneira o VSA gerencia diretamente os discos alocados para o sistema hiperconvergente.

4. Cada VSA, com seus próprios recursos de armazenamento, é configurado para participar de um cluster existente ou iniciar um novo. Cada nó disponibiliza seus recursos individuais ❻ para criar um volume distribuído entre os nós do cluster, vide Figura 3.2. A replicação de dados entre os nós é realizada por meio de uma rede Ethernet de baixa latência e alta taxa de transferência.

5. Após criação do volume virtual, o mesmo é utilizado pelo *Container Host* com uma unidade de armazenamento escalável e tolerante a falhas, conforme pode ser visto no caminho ❼. A replicação dos dados é realizada conforme configuração escolhida e a escalabilidade é alcançadas ao se adicionar mais discos.

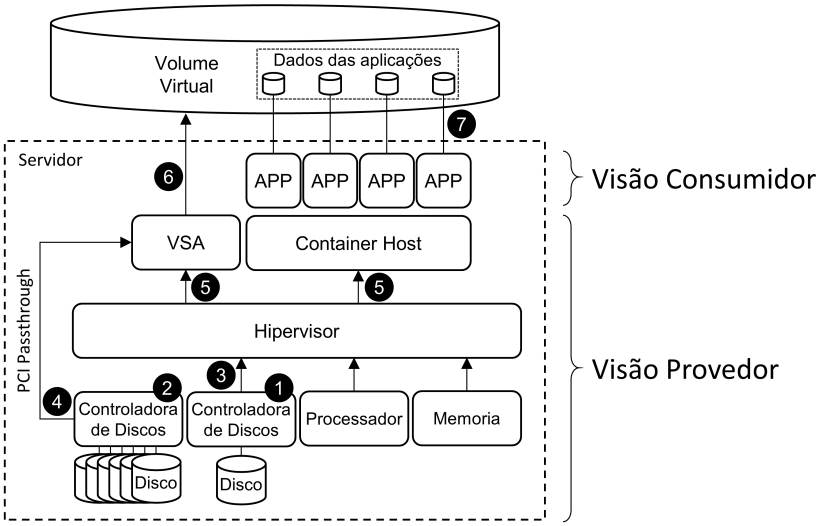

Figura 3.1: Arquitetura da solução proposta.

O componente principal da arquitetura é o VSA, um servidor virtual que roda uma versão mínima do sistema operacional Linux e a aplicação GlusterFS. O GlusterFS será utilizado para gerenciar os recursos de armazenamento apresentados ao VSA na forma de um sistema de arquivos distribuído. A escolha do GlusterFS para o sistema de arquivos distribuído se deve ao fato dos dados e metadados serem armazenados juntos, sem existir um nó específico para guardar metadados [40]. Essa característica torna o GlusterFS

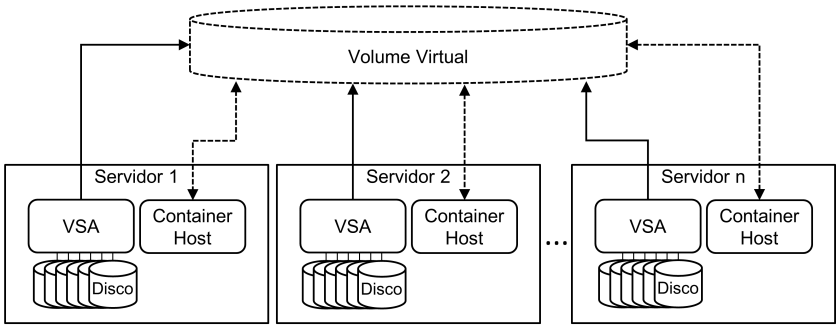

Figura 3.2: Volume virtual criado pelos VSAs.

ideal para uso em uma arquitetura hiperconvergente, visto que os nós funcionam sem dependências externas, simplificando a arquitetura e permitindo a instalação de apenas um VSA por hipervisor.

Os nós do VSA, cada um com seus recursos de armazenamento, permitem a criação de volumes virtuais distribuídos que podem ser utilizados pelos hipervisores para hospedar *containers*, conforme é apresentado na Figura 3.3. Note que a visão do cliente é limitada aos *containers*, enquanto o provedor de serviços em nuvem é responsável por toda a infraestrutura de armazenamento e computação exigida pelos *containers*.

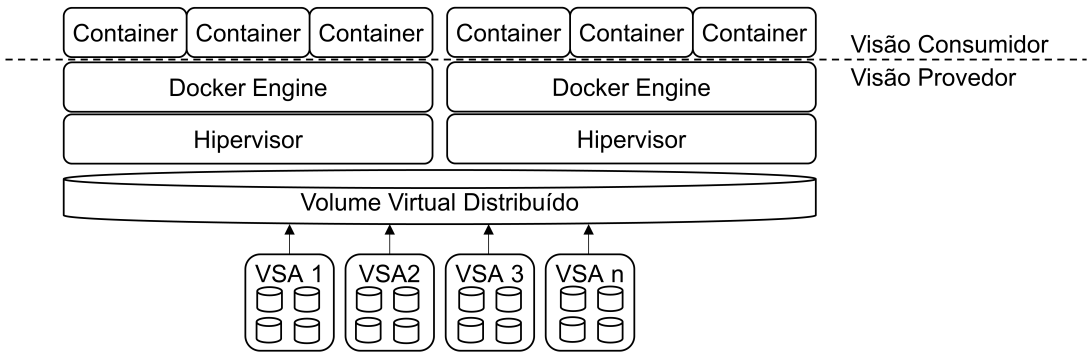

Figura 3.3: Visão do consumidor e do provedor.

Um comparativo entre uma arquitetura tradicional, onde cada servidor tem suas unidades de armazenamento individuais, e uma arquitetura hiperconvergente, onde cada servidor com suas unidades de armazenamento individuais contribuem para formar um volume virtual, pode ser observado na Figura 3.4.

O diagrama de fluxos para operações em discos do ponto de vista do *container* pode ser visto na Figura 3.5. O fluxo "armazenamento de imagens e *container*" fica hospedado no sistema de arquivos do *host* que por sua vez está hospedado em um sistema de arquivos

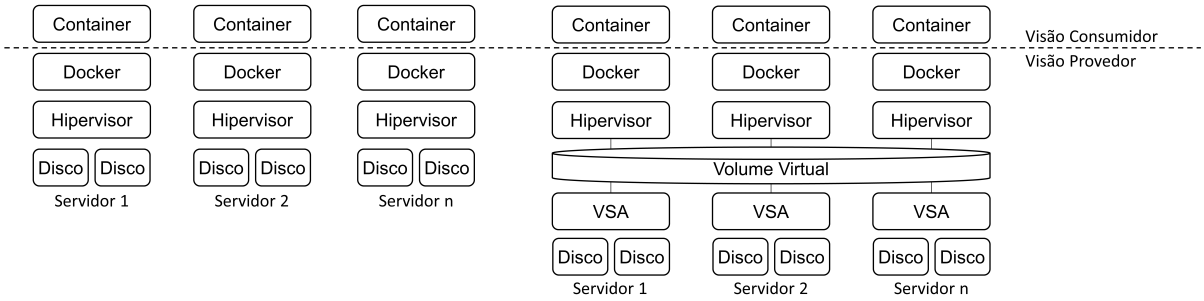

Figura 3.4: Arquitetura tradicional e hiperconvergente.

distribuído. Em relação ao "armazenamento persistente" o *container* utilizará um *data volume* que acessa diretamente o sistema de arquivos distribuído.

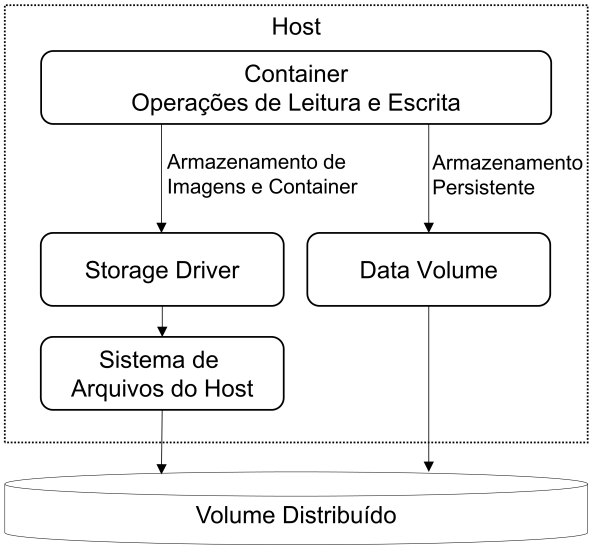

Figura 3.5: Fluxo das operações de disco na arquitetura proposta.

O diagrama da arquitetura proposta com todos os componentes pode ser observado na Figura 3.6. Os VSAs são *hosts* responsáveis pelo armazenamento distribuído. Os arranjos de discos que os VSAs permitem varia entre a soma de todos os recursos sem tolerância a falhas, até uma configuração altamente redundante e tolerante a falhas, que serão analisadas nesse trabalho.

3.2 Ferramentas Utilizadas para Implantação

A arquitetura proposta será implantada em um ambiente baseado no sistema operacional Linux que utilizará o conjunto de ferramentas descritos a seguir.

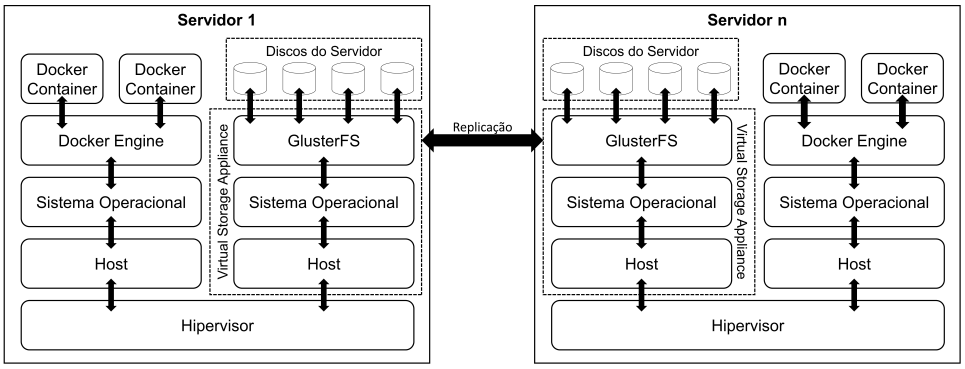

Figura 3.6: Fluxo das operações de armazenamento na arquitetura proposta.

GlusterFS

GlusterFS é um sistema de arquivos distribuído de código aberto que permite distribuir um grande conjunto de dados entre diversos servidores. É escalável e suporta receber um grande número de clientes. Possui simplicidade na operação, sendo suportado pela maioria das distribuições Linux. Não faz separação entre metadados/dados e por esse motivo não precisa de servidores adicionais para essa tarefa. É rápido para a maioria das operações pois calcula a localização de metadados usando algoritmos, em vez de recuperar metadados de outro nó. Todas as operações no GlusterFS estão sendo feitas com um "tradutor", que converte solicitações de usuários em solicitações de armazenamento [40].

Possui um mecanismo nativo de redundância baseado na replicação de arquivos entre um conjunto de "*Bricks*" (recursos de armazenamento) que formam uma unidade lógica resiliente [4]. *Bricks* são a unidade básica de armazenamento no GlusterFS, representada por um diretório em um servidor. Uma coleção de *bricks* forma um "Volume" sendo onde a maioria das operações do GlusterFS acontecem. O GlusterFS permite agrupar *Bricks* para formar volumes de cinco tipos distintos:

- *Volume Distribuído:* Distribui arquivos entre os *bricks*. A capacidade final do volume é a soma da capacidade dos *bricks*. Não possui redundância. A falha em um *brick* corrompe o volume. Recomendado para casos onde a disponibilidade não é importante ou é provida por outro mecanismo.
- *Volume Replicado:* Os arquivos são replicados entre os *bricks* do volume. Recomendado para casos onde exige-se a alta disponibilidade e confiabilidade.
- *Volume Distribuído Replicado:* Distribui arquivos entre subvolumes replicados. Recomendado para casos onde exige-se escalabilidade, alta disponibilidade e bom desempenho na leitura.

- *Volume Disperso:* Utiliza "*erasure codes*" para fornecer proteção eficiente contra falhas de discos ou servidores. Um fragmento codificado do arquivo original é armazenado em cada *brick*, de maneira que a partir de um conjunto de fragmentos seja possivel recuperar o arquivo original.
- *Volume Distribuído Disperso:* Distribui arquivos entre subvolumes dispersos. Possui as vantagens de um volume distribuído, porém usando armazenamento disperso no subvolume.

O GlusterFS também permite o uso de nós heterogêneos com diferentes configurações pertençam ao mesmo volume distribuído. Nesses casos recomenda-se o agrupamento de nós com mesmas características para o melhor desempenho [27].

ESXi

O VMware ESXi é um hipervisor independente, executado diretamente no hardware do hospedeiro. Ou seja, é um monitor de máquina virtual (VMM - *Virtual Machine Monitor*) do tipo I (*bare metal*). Ocupa pouca memória do servidor físico e oferece uma camada de virtualização robusta e de bom desempenho, que abstrai os recursos de hardware do servidor e permite o compartilhamento desses recursos entre vários servidores virtuais. Os recursos de gerenciamento de memória e processamento do VMware ESXi fornecem boas taxas de consolidação e bom desempenho para máquinas virtuais [20].

Docker

O Docker surgiu em 2013 como projeto de PaaS (*Platform as a Service*) da empresa dotCloud para facilitar e integrar a adoção da tecnologia de *containers* [41]. O Docker utiliza as tecnologias de *cgroups* e *namespaces* para isolamento e controle de recursos que permitem o empacotamento de aplicações com todas as suas dependências em uma imagem [24], e possibilita que aplicações diferentes sejam executadas sem conflitos no mesmo servidor. Além disso, também é possível encontrar, baixar e executar imagens que foram criadas por outras pessoas, flexibilizando sua utilização. Uma imagem do Docker é uma série de camadas concatenadas, sendo que a primeira é uma imagem base (geralmente composta pelos arquivos básicos de uma distribuição Linux) e as demais camadas são customizações. Essas camadas são gerenciadas de forma eficiente pelo sistema de arquivos utilizado pelo Docker, que permite que as alterações em uma imagem sejam simplesmente uma atualização diferencial da imagem anterior [28]. Essas imagens podem ser armazenadas em repositórios (*registry*) públicos ou privados. Os repositórios públicos do Docker provêm imagens de *containers* com aplicações prontas para executar, tais como bancos de dados, servidores web entre outros [24].

3.3 Análise da Arquitetura Proposta

A solução proposta possibilita agregar os recursos computacionais de armazenamento de dados, dispersos em diversos servidores individuais dentro de um datacenter, visando consolida-los e disponibiliza-los para o fornecimento de serviços de IaaS e/ou PaaS. Com o uso de hipervisores para virtualizar o processamento e memória, de um sistema de arquivos distribuído para virtualizar os recursos físicos de armazenamento (discos) e da arquitetura proposta nesse trabalho, é possível agregar esses recursos de forma aperfeiçoada e tolerante a falhas (replicando dados entre os nós) para disponbiliza-los como base para outros serviços de um provedor de serviços em nuvem.

A arquitetura proposta permite uma escalabilidade do tipo *scale-out* na infraestrutura de servidores e armazenamento de dados, onde basta-se acrescentar mais servidores para aumentar os recursos. Inclusive permite que um determinado tipo de recurso (processamento, memória ou armazenamento) seja aumentando de forma pontual. Uma infraestrutura hiperconvergente com falta de recursos de processamento pode ser escalada ao incluir na topologia servidores com melhores processadores. De forma análoga outros recursos podem ser aumentados pontualmente, se destacando os recursos de armazenamento. O armazenamento de dados na infraestrutura proposta pode escalar simplesmente ao acrescentar novos nós com mais discos ou aumentar os discos no nós existentes.

A proposta também provê tolerância a falhas ao replicar os dados em discos diferentes de servidores diferentes. A proposta permite replicar dados em diferentes quantidades de cópias, a depender do tipo do volume configurado no VSA. Um problema em um disco não afeta o funcionamento do volume virtual, e a substituição pode ser realizada sem interromper a operação. O rebalanceamento do volume após troca de um disco com falha é realizado automaticamente pelo VSA instalado em cada nó hiperconvergente. O VSA utiliza uma parcela dos recursos de computação do nó hiperconvengete para as operações de distribuição de dados [10], o que permite realiza-las de forma mais eficiente do que um hardware monolítico de armazenamento que possui recursos limitados de processamento.

Os recursos de rede da infraestrutura hiperconvergente é um ponto que precisa estar bem dimensionado para evitar problemas de desempenho. Como todas as operações de armazenamento são replicadas entre os nós, a comunicação entre os nós precisa acontecer sem congestionamentos ou erros. Como as operações de I/O na proposta dependem fortemente da infraestrutura de rede, recomenda um rede com alta taxa de transmissão para dar vazão na carga de trabalho imposta pelos VSAs.

3.4 Conclusão do Capítulo

Este capítulo apresentou os componentes e como os mesmos serão integrados de modo a construir a arquitetura proposta nesse trabalho. Foram detalhados as tecnologias utilizadas e a maneira que as mesmas se integram de forma a prover uma área de armazenamento de dados para containers dentro de uma arquitetura hiperconvergente. Foi descrito também o funcionamento do VSA, componente responsável por gerenciar os recursos físicos de armazenamento (discos), integrá-los a um sistema de arquivo distribuído e disponiliza-los para armazenamento de dados de containers. O capítulo seguinte apresentará a avaliação experimental da arquitetura proposta.

Capítulo 4

Resultados Experimentais

Para avaliar a proposta da arquitetura apresentada no capítulo anterior, neste capítulo são apresentados um conjunto de cenários experimentais e os seus resultados, os quais serão comparados com os valores de uma arquitetura tradicional de armazenamento persistente de dados de containers.

4.1 Ambiente

Para avaliação da arquitetura foram utilizados 10 servidores da marca IBM modelo HS23 com 2 processadores Intel Xeon E5-2670 @ 2.60GHz, 96 GB de RAM e 2 discos. Os servidores foram divididos em 4 grupos, vide Figura 4.1, cada um com diferentes arranjos de servidores e discos:

- Grupo 1: 3 servidores com 2 discos SSD (solid state drive) da marca Lite-On, modelo CV3-CE128, 128GB de capacidade, controlados diretamente pelo VSA.
- Grupo 2: 3 servidores com 2 discos HDD (hard disk drive) da marca Seagate, modelo ST91000640SS, 1TB de capacidade, controlados diretamente pelo VSA.
- Grupo 3: 1 servidor com 1 disco HDD (marca Seagate, modelo ST91000640SS e 1TB de capacidade) e 1 disco SSD (marca Lite-On, modelo CV3-CE128 e 128GB de capacidade). Esse grupo foi utilizado para comparação dos resultados entre a arquitetura proposta uma arquitetura tradicional.
- Grupo 4: 9 servidores com 2 discos HDD (hard disk drive) da marca Seagate, modelo ST91000640SS, 1TB de capacidade, controlados diretamente pelo VSA.

Cada servidor contou também com uma unidade de armazenamento USB (marca SanDisk, modelo Cruzer Glide USB 3.0 e 16GB de capacidade), onde foi instalado o

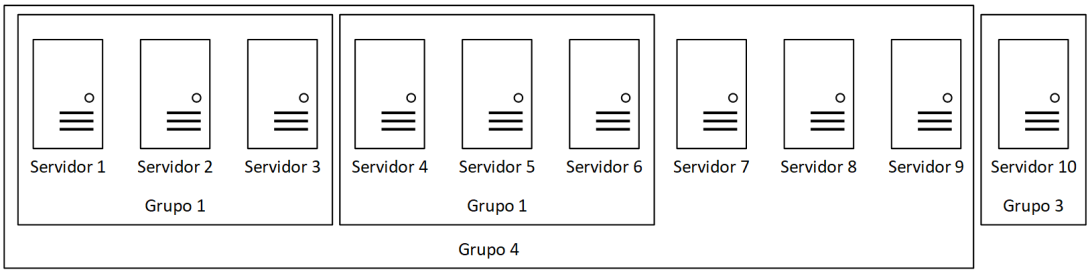

Figura 4.1: Grupos de Servidores.

hipervisor VMware ESXi 6.5, o VSA e o servidor Docker (*Container Host*). A utilização do armazenamento USB foi necessária devido ao fato do servidores possuírem espaço físico somente para instalação de dois discos.

Cada VSA consistiu-se de um servidor virtual com 8 vCPU, 12GB de RAM, uma instalação mínima do sistema operacional CentOS Linux (versão 7.5.1804) e do sistema de arquivos distribuído GlusterFS (versão 3.12.14).

Cada servidor Docker (*Container Host*) consistiu-se de um servidor virtual com 24 vCPU, 80GB de RAM, uma instalação mínima do sistema operacional CentOS Linux (versão 7.5.1804) e do Docker (versão 18.09).

Os servidores físicos foram conectados em uma rede Gigabit Ethernet, por meio de duas NICs (Network Interface Card) de 1Gbps (1000Base-T). Os caminhos de comunicação configurados pode ser observados na Figura 4.2. Foram criados três vSwitches (switches virtuais), sendo um para replicação de dados entre os VSA (vSwitch Replicação), outro para o servidor Docker acessar o servidor VSA (vSwitch SAN) e outro para a comunicação com redes externas (vSwitch Produção). As redes de replicação e produção foram separadas para evitar possíveis problemas de congestionamentos na rede.

4.2 Cargas de Trabalho

Para avaliar o desempenho da arquitetura proposta foram utilizadas cargas de trabalho baseadas nos trabalhos de pesquisadores da Microsoft [42] e Yahoo [43]. Nos cenários 1 e 2 foram utilizadas duas cargas de trabalho baseadas no padrão de I/O de aplicações reais que rodaram nos *datacenters* da Microsoft. No cenário 3 foram utilizadas as 2 cargas anteriores mais 6 cargas de trabalho principais do Yahoo Cloud Serving Benchmark (YCSB) acrescida de mais uma com alta carga de escrita. No cenário 4 foram utilizadas 11 cargas de trabalho diferentes inspiradas nos padrões de I/O da Microsoft [42] e Yahoo [43].

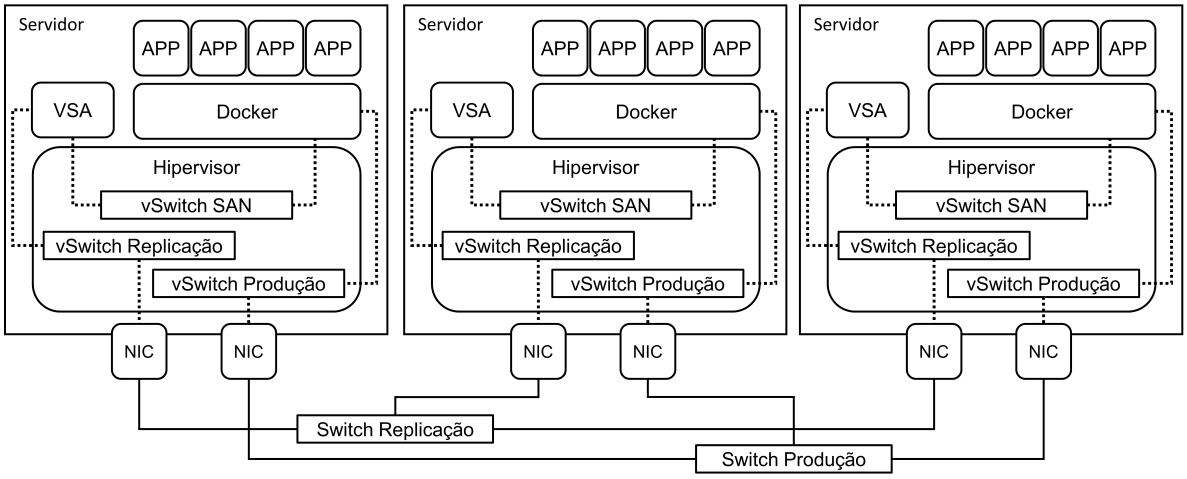

Figura 4.2: Caminhos de comunicação.

4.2.1 Datacenters Microsoft

No trabalho realizado por pesquisadores da Microsoft [42] foram observados perfis de leitura/escrita de dados de aplicações reais próprias que funcionam em seus *datacenters*. Para a execução dos experimentos desse trabalho foram escolhidos dois desses perfis de leitura/escrita:

- **26,2% leitura e 73,8% escrita**: Denominada "Cloud Storage", essa razão de escrita/leitura foi observada nos serviços de armazenamento de dados em nuvem da Microsoft. Foi escolhida para ser executada no experimento por possui uma maior carga de escrita.
- **83% leitura e 17% escrita**: Denominada "Web Search", essa razão de escrita/leitura foi observada no serviço nos serviços de busca da Microsoft. Foi escolhida para ser executada no experimento por possui uma maior carga de leitura.

Para simular uma aplicação real fazendo operações em disco, a ferramenta de benchmark FIO [44] foi instalada em um container e configurada para executar essas duas cargas de trabalho durante sua execução. Essa ferramenta mede o desempenho de um sistemas de arquivos ao executar cargas personalizadas de operações em disco, exibindo ao final informações tais como, taxa de transferência obtida, quantidade de operações por segundo (IOPS) e latência das operações.

Além das duas razões leitura/escrita explanadas, a carga de trabalho variou a quantidade e tamanho dos arquivos, realizando operações em blocos aleatórios de 4KB nos arquivos. A imagem do container foi configurada para iniciar, executar a carga de trabalho usando a ferramenta FIO, salvar o resultado e terminar a execução do container.

4.2.2 Yahoo Cloud Serving Benchmark

O YCSB (Yahoo Cloud Serving Benchmark) é uma ferramenta usada medir desempenho de banco de dados NoSQL [43]. Ela carrega dados no banco NoSQL desejado, executa cargas de trabalho e ao fim fornece o desempenho alcançado. Por padrão o YCSB possui 6 cargas de trabalho principais, baseadas no padrão de I/O de aplicações reais do Yahoo:

- **Workload A - 50% leitura e 50% escrita**: Padrão de uma aplicação que grava e monitora ações recentes de usuários.
- **Workload B - 95% leitura e 5% escrita**: Padrão de uma aplicação que gerencia os textos em fotos (photo tagging), onde a adição do texto é uma operação de escrita, mas a maioria das operações são de leitura desse texto.
- **Workload C - 100% leitura**: Padrão de uma aplicação que armazena dados de usuários, mas o perfil do usuário é montado em outra aplicação.
- **Workload D - 95% leitura dos registros mais recentes e 5% escrita**: Padrão de uma aplicação que armazena o estado (status) de usuários, onde a alteração do estado gera um operação de escrita, mas a maioria da operações são a leitura dos dados recentemente escritos.
- **Workload E - 95% leitura de registros encadeados e 5% escrita**: Padrão de uma aplicação de conversas encadeadas, onde vários registros de texto que geram operações de leitura são encadeados para formar a conversa.
- **Workload F - 50% leitura e 50% de modificação e escrita**: Padrão de uma aplicação que armazena dados de usuário, onde pode ser necessário alterar o valor de algum registro, gerando uma operação de escrita para exibir o valor do registro atual e outra para modificá-lo e salvá-lo.

Além das cargas de trabalho principais, o YCSB permite a execução de cargas personalizas. Dessa forma foram elaboradas mais três cargas de trabalho, sendo as duas da Microsoft descritas anteriormente e outra com alta carga de escrita (100% escrita). Essa última de deve ao fato de nenhum dos trabalhos possuir esse padrão de I/O, sendo que existem aplicações com esse aspecto, como por exemplo operações de backup. A Tabela 4.1 sumariza os padrões de I/O utilizados no YCSB durante os experimentos.

Para simular uma aplicação real fazendo operações em um banco de dados NoSQL, foram escolhidos dois sistemas gerenciados de banco de dados não SQL (NoSQL): Cassandra [45] e MongoDB [46]. Esses dois bancos de dados foram instalados em containers e configurados para receberem conexões externas. O YCSB foi instalado em um outro

Tabela 4.1: Cargas de Trabalho Utilizadas nos Experimentos com YCSB.

Carga de Trabalho	**Padrão de I/O**
YCSB-A	50% leitura, 50% escrita
YCSB-B	95% leitura, 5% escrita
YCSB-C	100% leitura
YCSB-D	95% leitura dos registros mais recentes, 5% escrita
YCSB-E	95% leitura de registros encadeados, 5% escrita
YCSB-F	50% leitura, 50% modificação-escrita
Microsoft Cloud Storage	26,2% leitura, 73,8% escrita
Microsoft Web Search	83% leitura, 17% escrita
Alta Carga de Escrita	100% escrita

container, com a função de disparar a carga de trabalho desejada, exibindo ao final informações tais como, operações por segundo (ops/sec), latência das operações e tempo de execução.

4.2.3 Padrões Diversos de I/O

Visando simular outros tipos de padrões de I/O que não foram testados nas cargas de trabalho anteriores, foram elaboradas uma sequencia de 11 cargas que variam entre 100% escrita até 100% leitura. Dessa forma abrange-se uma grande quantidade de perfis de leitura e escrita que aplicações diversas podem apresentar. A Tabela 4.2 sumariza os padrões utilizados.

Tabela 4.2: Cargas de trabalho para os experimentos com diferentes padrões de I/O.

Carga de Trabalho	**Padrão de I/O**
0/100	100% escrita
10/90	10% leitura, 90% escrita
20/80	20% leitura, 80% escrita
30/70	30% leitura, 70% escrita
40/60	40% leitura, 60% escrita
50/50	50% leitura, 50% escrita
60/40	60% leitura, 40% escrita
70/30	70% leitura, 30% escrita
80/20	80% leitura, 20% escrita
90/10	90% leitura, 10% escrita
100/0	100% leitura

Para simular uma aplicação real fazendo operações em disco, novamente a ferramenta de benchmark FIO [44] foi instalada em um container e configurada para executar uma carga de trabalho durante sua execução, exibindo ao final a taxa de transferência. Dentro de cada padrão de I/O foram manipulados 10 arquivos de 100MB realizando operações em

blocos aleatórios de 4KB nos arquivos. A imagem do container foi configurada para subir, executar a carga de trabalho usando a ferramenta FIO, salvar o resultado e terminar a execução do container.

4.3 Métricas, Validação e Fatores

O ambiente experimental foi avaliado com base em uma metodologia estatística, onde os experimentos foram repetidos e a média calculada com intervalo de confiança de 95%. As métricas avaliadas nos experimentos foram:

- Cenários 1 e 2:
  - **Taxa de escrita e leitura**: Quantidade de dados que são transferidos por segundo para o disco.
  - **IOPS** (*Input/output Operations Per Second*): Quantidade de operações em disco realizadas por segundo.
- Cenário 3:
  - **Operações por Segundo**: Quantidade de operações em banco de dados realizadas por segundo.
- Cenário 4:
  - **Taxa de escrita e leitura**: Quantidade de dados que são transferidos por segundo para o disco.

A validação do ambiente experimental foi realizada por meio de comparação com os valores das métricas em uma arquitetura tradicional de armazenamento onde os dados são armazenados diretamente nos discos dos servidores.

Parâmetros são elementos que quando sofrem variações modificam o desempenho de um sistema, entretanto ficam fixos durante os experimentos. O parâmetros que de fato serão variáveis nos experimentos são chamados de fatores [47]. Seguindo essas definições, os parâmetros e fatores dessa proposta foram:

- **Parâmetros**
  - Desempenho do servidor (processamento, memória, barramentos, etc.).
  - Desempenho individual dos discos (taxa de transferência, IOPS, latência, etc.).
  - Desempenho da rede (taxa de transferência, latência, perda de pacotes, etc.).

  - Overhead na estrutura de armazenamento de dados imposto pelo sistema de arquivos distribuído.
  - Capacidade de processamento e memória do VSA.
  - Tamanho do bloco (4KB).
  - Acesso randômico aos blocos.

- **Fatores**
  - Cenários 1 e 2:
    * Tamanho dos arquivos (1MB, 10MB, 100MB e 1000MB).
    * Quantidade de arquivos (1 e 10).
    * Padrão de I/O.
    * Tipo do volume.
  - Cenário 3:
    * Padrão de I/O.
    * Tipo do volume.
  - Cenário 4:
    * Padrão de I/O.
    * Quantidade de discos no volume.

A escolha pela variação dos tamanhos e quantidade de arquivos operados simultaneamente, se deu pela necessidade de analisar o comportamento da arquitetura proposta sobre uma ampla gama de perfis de leitura/escrita observados na literatura.

4.4 Cenários de Avaliação

O cenário 1 foi elaborado no início dos experimentos como prova de conceito da proposta. Neste cenário foi realizado apenas uma rodada de testes, portanto não foram gerados intervalos de confiança para os resultados obtidos. Para os cenários 2, 3 e 4, após a prova de conceito, foram realizados testes repetitivos com o cálculo de médias estatísticas e definição de intervalos de confiança.

4.4.1 Cenário 1

Nesse cenário foi utilizada a ferramenta FIO e avaliado o desempenho da arquitetura proposta usando discos de estado sólido (SSD, do inglês *Solid State Drive*), utilizando o grupo 1 de servidores. As duas cargas de trabalho da Microsoft foram aplicadas (26,2/73,8

e 83/17), variando-se os tipos de volume (replicado distribuído, disperso e disperso distribuído), tamanho dos arquivos (1MB, 10MB, 100MB e 1000MB), quantidade de arquivos sendo operados simultaneamente (1 e 10). Os resultados obtidos foram comparados com a mesma carga de trabalho sendo executada no servidor do grupo 3, onde os dados são armazenados diretamente no SSD local do servidor.

Nesse cenário o GlusterFS instalado em cada VSA possibilita a configuração de três tipos diferentes de volumes virtuais, cada um com diferentes desempenhos. Diante disso os três tipos de volumes foram avaliados, sendo eles:

- **Volume Replicado Distribuído**: Os discos são subdivididos em dois grupos, com cada arquivo em um subgrupo possuindo 3 réplicas, vide Figura 4.3.

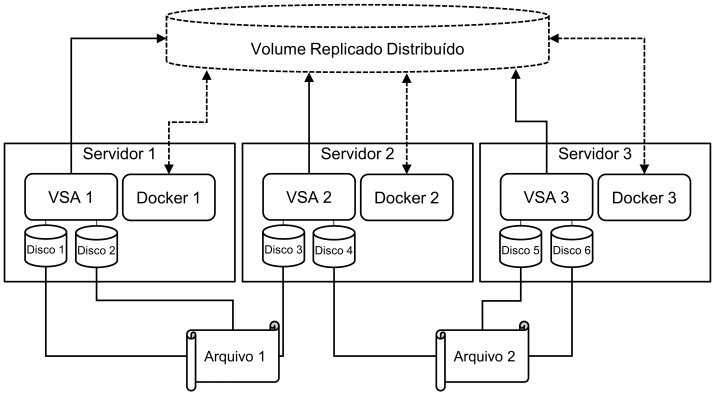

Figura 4.3: Volume Replicado Distribuído.

- **Volume Disperso**: Os arquivos são quebrados fragmentados e codificados. Esses fragmentos são multiplicados e armazenados em diferentes discos. Caso um disco apresente problema, os fragmentos do arquivo armazenados em outros discos são capazes de reconstruir o arquivo, vide Figura 4.4.

- **Volume Disperso Distribuído**: Os discos são subdivididos em dois grupos, com cada arquivo em um subgrupo usando o algoritmos de dispersão apresentado no item anterior, vide Figura 4.5.

Os resultados obtidos nesse cenário pode ser observados nas Figuras 4.6 e 4.7. Na Figura 4.6 estão os resultados referentes a carga de trabalho “Cloud Storage” e na Figura 4.7 estão os resultados referentes a carga de trabalho “Web Search”, sendo ambas da Microsoft [42]. Na primeira coluna da figura estão os resultados referentes a realização da carga de trabalho com um único arquivo de tamanhos variados (1MB, 10MB, 100MB e 1000MB). Na segunda coluna estão os resultados referente a execução da carga de trabalho

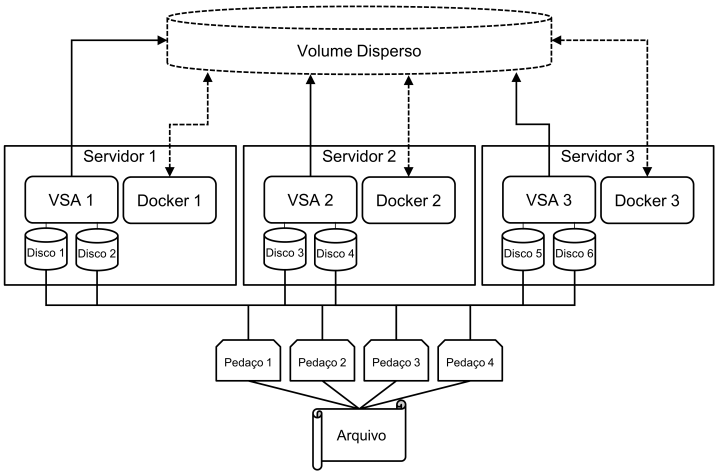

Figura 4.4: Volume Disperso.

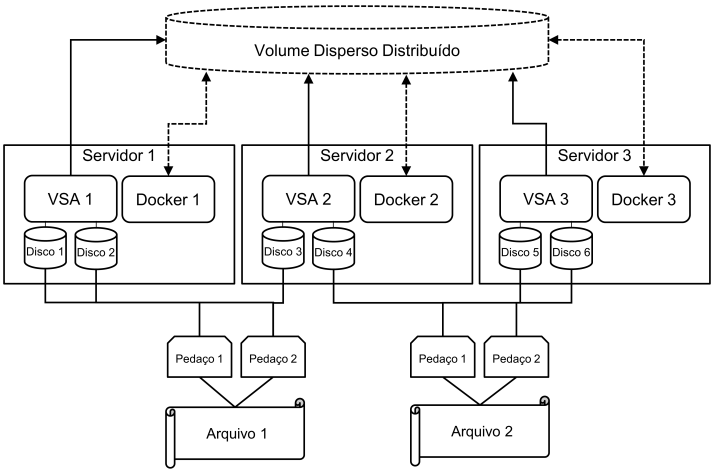

Figura 4.5: Volume Disperso Distribuído.

com 10 arquivos simultâneos. A primeira linha nas Figuras 4.6 e 4.7 exibem a taxa de transferência de leitura, a segunda a taxa de escrita e a terceira a quantidade de IOPS.

Ainda dentro desse cenário foram realizadas modificações nas configurações do GlusterFS visando aumentar seu desempenho. Após reconfiguração o ambiente foi submetido as mesmas cargas de trabalho e os resultados podem ser observados na nas Figuras 4.8 e 4.9.

O comparativo entre o desempenho do GlusterFS com configurações padrão (Figuras 4.6 e 4.7) e com configurações modificadas (Figuras 4.8 e 4.9) podem ser observadas nas Figuras 4.10 e 4.11. Na Figura 4.10 estão os resultados referentes a carga de trabalho "Cloud Storage" e na Figura 4.11 estão os resultados referentes a carga de trabalho "Web Search". Na primeira coluna das figuras estão os dados referentes a Leitura e na segunda

coluna os dados relativos a Escrita.

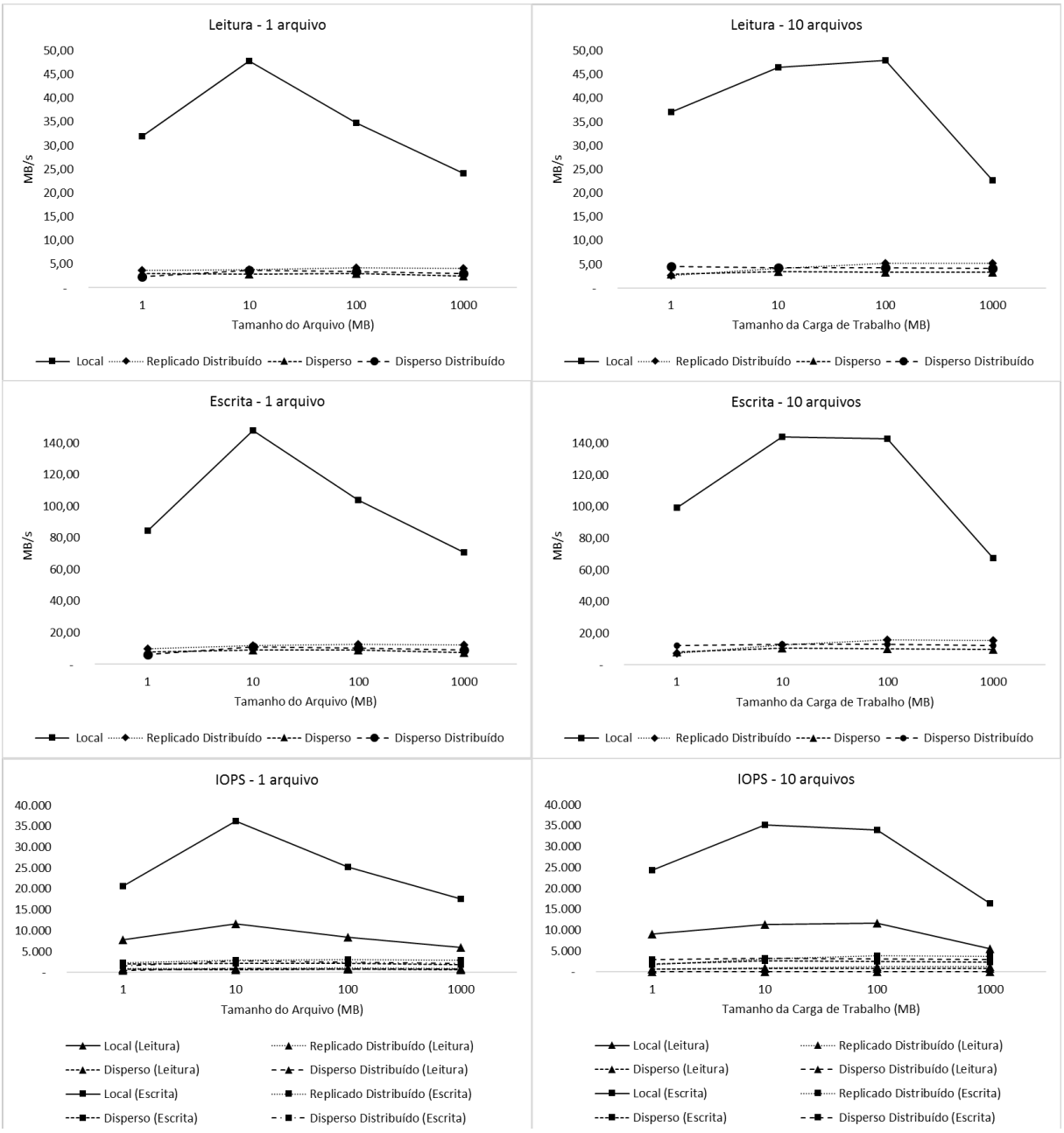

Figura 4.6: Resultados Cenário 1 - Carga de trabalho “Microsoft Cloud Storage” (26,2% leitura e 73,8% escrita) usando SSD.

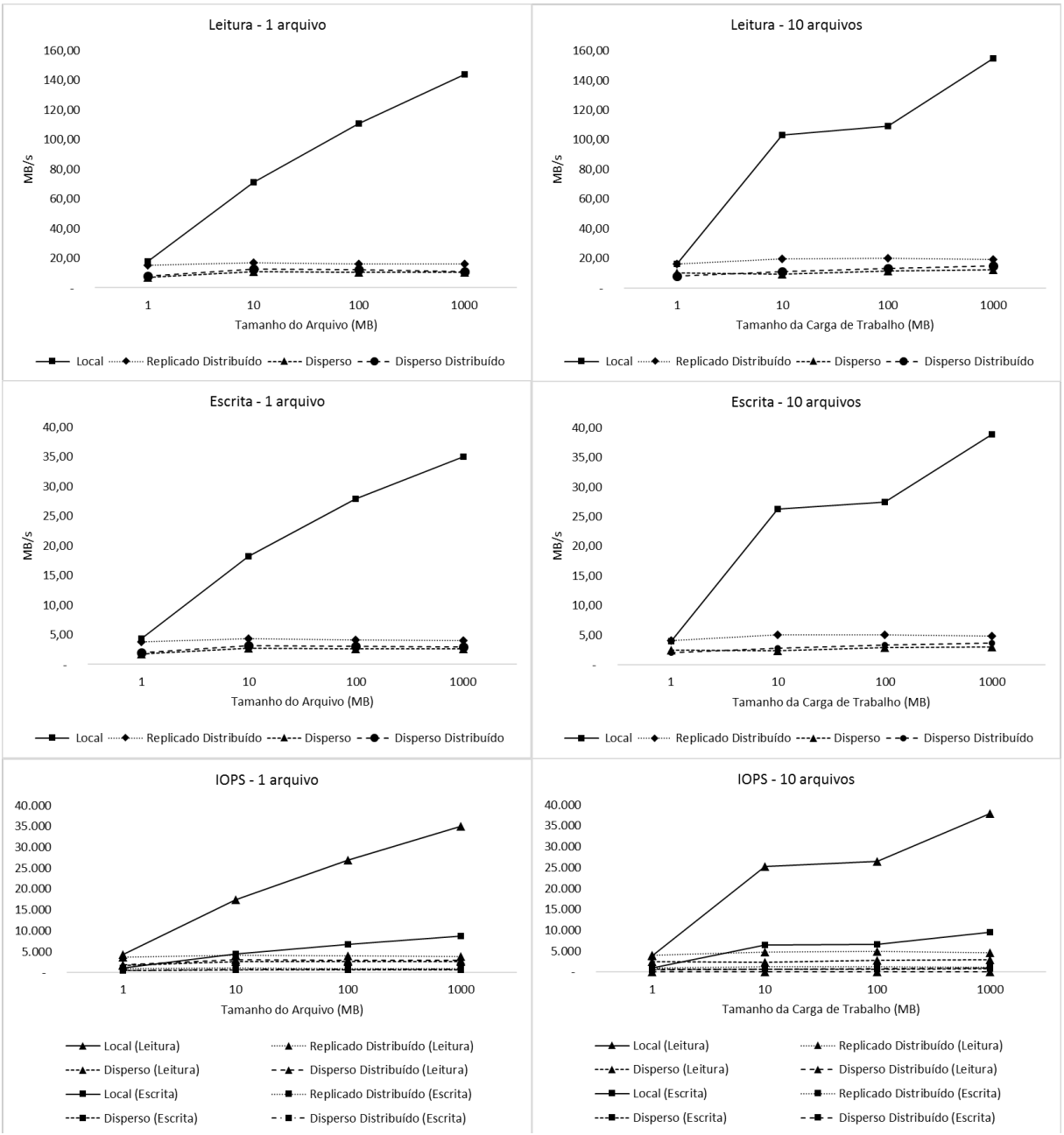

Figura 4.7: Resultados Cenário 1 - Carga de trabalho “Microsoft Web Search” (83% leitura e 17% escrita) usando SSD.

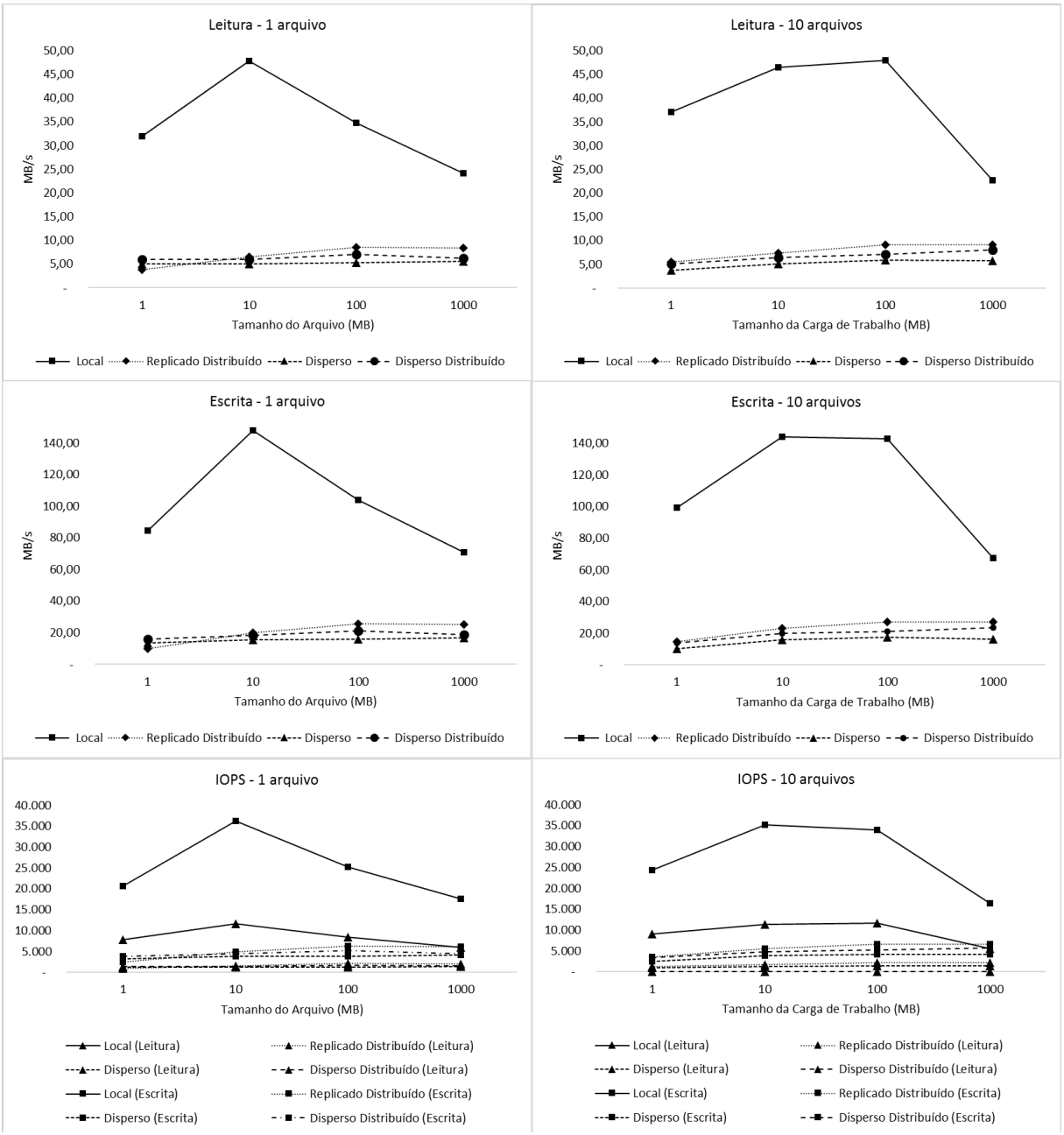

Figura 4.8: Resultados Cenário 1 - Carga de trabalho “Microsoft Cloud Storage” (26,2% leitura e 73,8% escrita) usando SSD e GlusterFS Modificado.

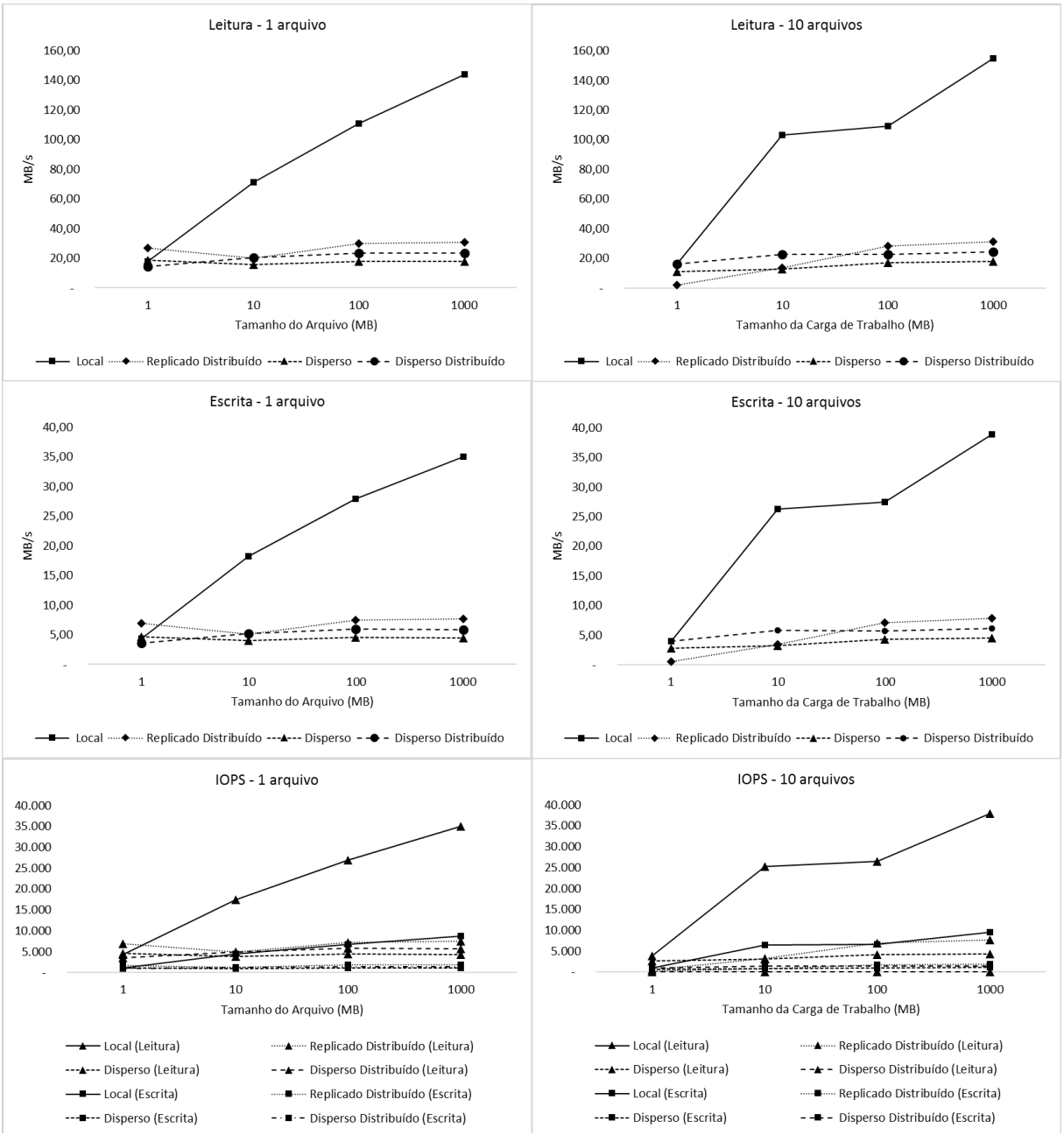

Figura 4.9: Resultados Cenário 1 - Carga de trabalho “Microsoft Web Search” (83% leitura e 17% escrita) usando SSD e GlusterFS Modificado.

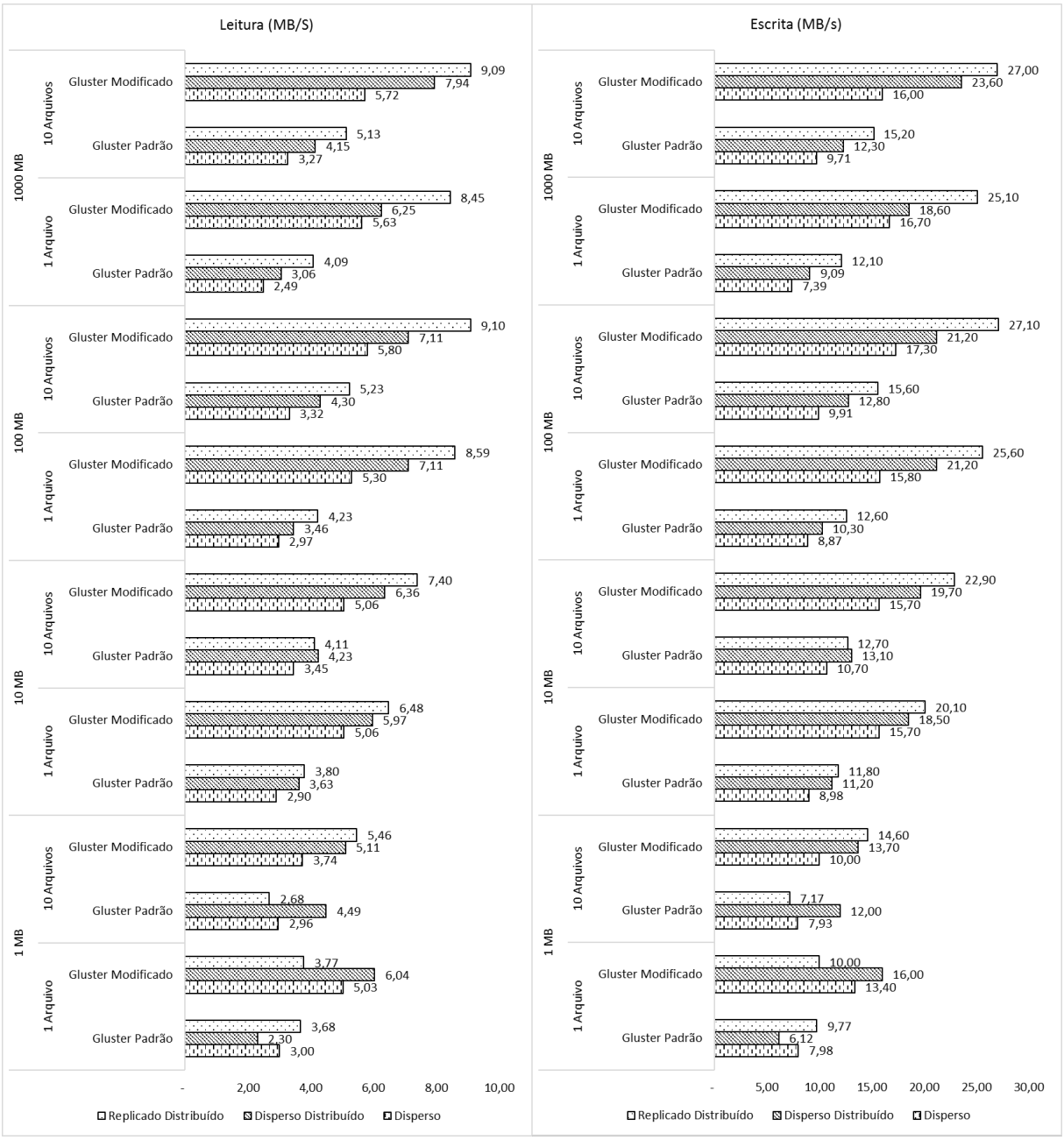

Figura 4.10: Resultados Cenário 1 - Comparativo entre GlusterFS padrão e modificado, usando SSD e Carga de Trabalho “Microsoft Cloud Storage” (26,2% leitura e 73,8% escrita).

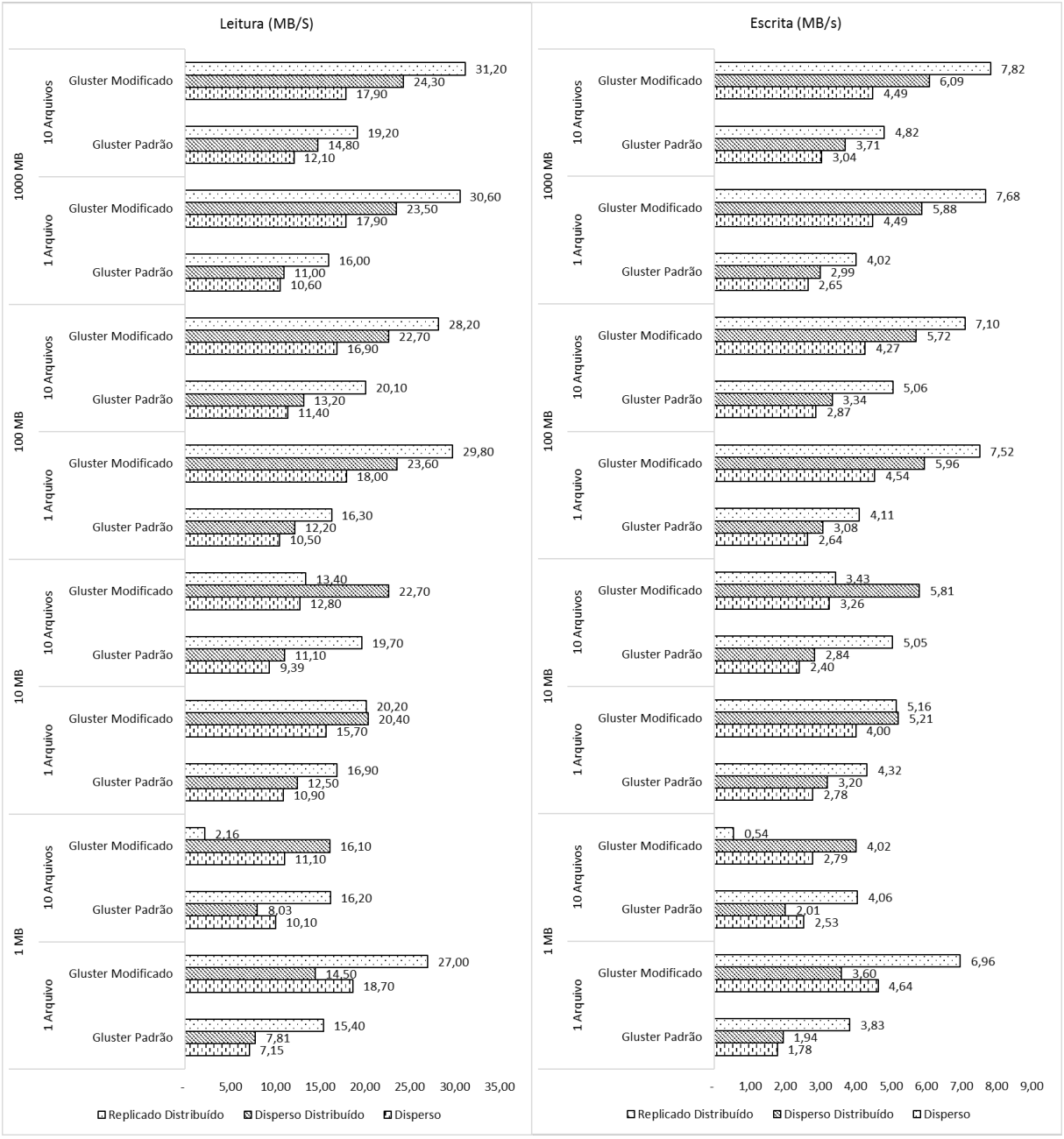

Figura 4.11: Resultados Cenário 1 - Comparativo entre GlusterFS padrão e modificado, usando SSD e Carga de Trabalho “Microsoft Web Search” (83% leitura e 17% escrita) .

Observando os resultados obtidos no Cenário 1 (Figuras 4.6 e 4.7), que utiliza os servidores do Grupo 1 equipados com discos do tipo SSD, nota-se que o desempenho da arquitetura proposta é inferior ao obtido pelo servidor do Grupo 3 que utiliza o armazenamento local. Esse comportamento é conhecido na literatura e se deve ao fato do GlusterFS ser otimizado para uso de discos convencionais do tipo HDD [48] [49]. O GlusterFS foi concebido para ser um sistema de arquivos distribuído baseado no uso de HDDs

e por esse motivo possui diversos algoritmos para funcionar melhor desempenho usando esse tipo de disco.

Usando discos do tipo SSDs é possível obter um melhor desempenho ao desativar algum desses algoritmos que otimizam o uso de HDD. Dessa forma, ainda dentro do Cenário 1, a configuração do GlusterFS foi modificada, desativando-se os algoritmos de "io-cache", "read-ahead" e "write-behind" [48]. Os resultados obtidos com essa modificação pode ser visto nas Figuras 4.8 e 4.9. Porém mesmo com essa modificação os resultados ficam a baixo dos obtidos pelo servidor do Grupo 3 que utiliza o armazenamento local. Apesar disso, os resultados do GlusterFS com configurações modificadas são em média 50% melhores aos resultados obtidos com a configuração padrão (Figuras 4.10 e 4.11).

4.4.2 Cenário 2

Nesse cenário foi utilizada a ferramenta FIO e avaliado o desempenho da arquitetura proposta usando discos magnéticos comuns (HDD, do inglês *Hard Disk Drive*), utilizando o grupo 2 de servidores. As duas cargas de trabalho da Microsoft foram aplicadas (26,2/73,8 e 83/17), variando-se os tipos de volume (replicado distribuído, disperso e disperso distribuído), tamanho dos arquivos (1MB, 10MB, 100MB e 1000MB), quantidade de arquivos sendo operados simultaneamente (1 e 10). Os resultados obtidos foram comparados com a mesma carga de trabalho sendo executada no servidor do grupo 3, onde os dados são armazenados diretamente no HDD local do servidor.

Nesse cenário o GlusterFS instalado em cada VSA também possibilita a configuração de três tipos diferentes de volumes virtuais, cada um com diferentes desempenhos. Dessa forma os três tipos de volumes descritos nas Figuras 4.3, 4.3 e 4.5 foram avaliados.

Os resultados obtidos nesse cenário pode ser observados nas Figuras 4.12 e 4.13. Na Figura 4.12 estão os resultados referentes a carga de trabalho "Cloud Storage" e na Figura 4.13 estão os resultados referentes a carga de trabalho "Web Search", sendo ambas da Microsoft [42]. Na primeira coluna da figura estão os resultados referentes a realização da carga de trabalho com um único arquivo de tamanhos variados (1MB, 10MB, 100MB e 1000MB). Na segunda coluna estão os resultados referente a execução da carga de trabalho com 10 arquivos simultâneos.

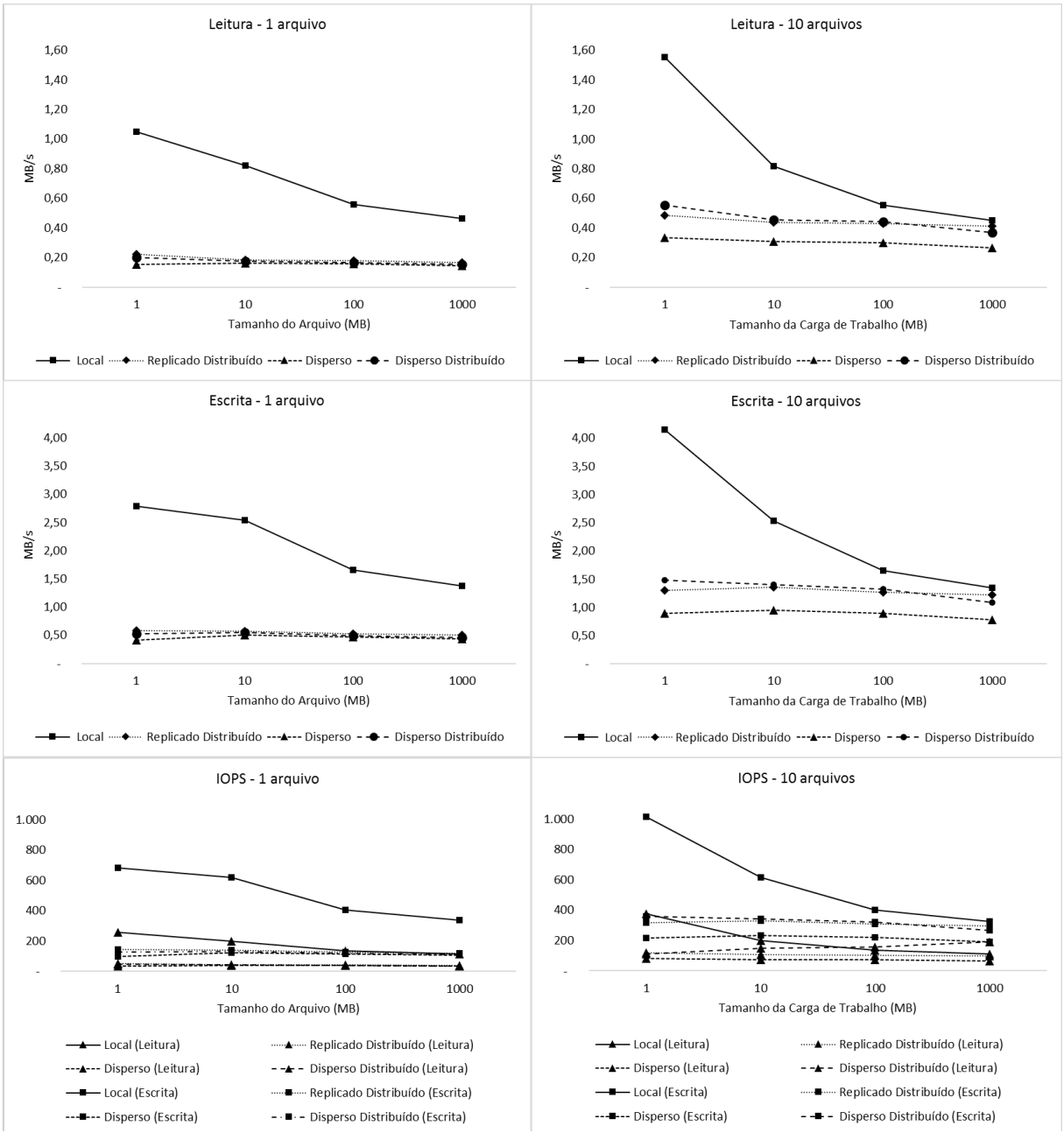

Figura 4.12: Resultados Cenário 2 - Carga de trabalho “Microsoft Cloud Storage” (26,2% leitura e 73,8% escrita) usando HDD.

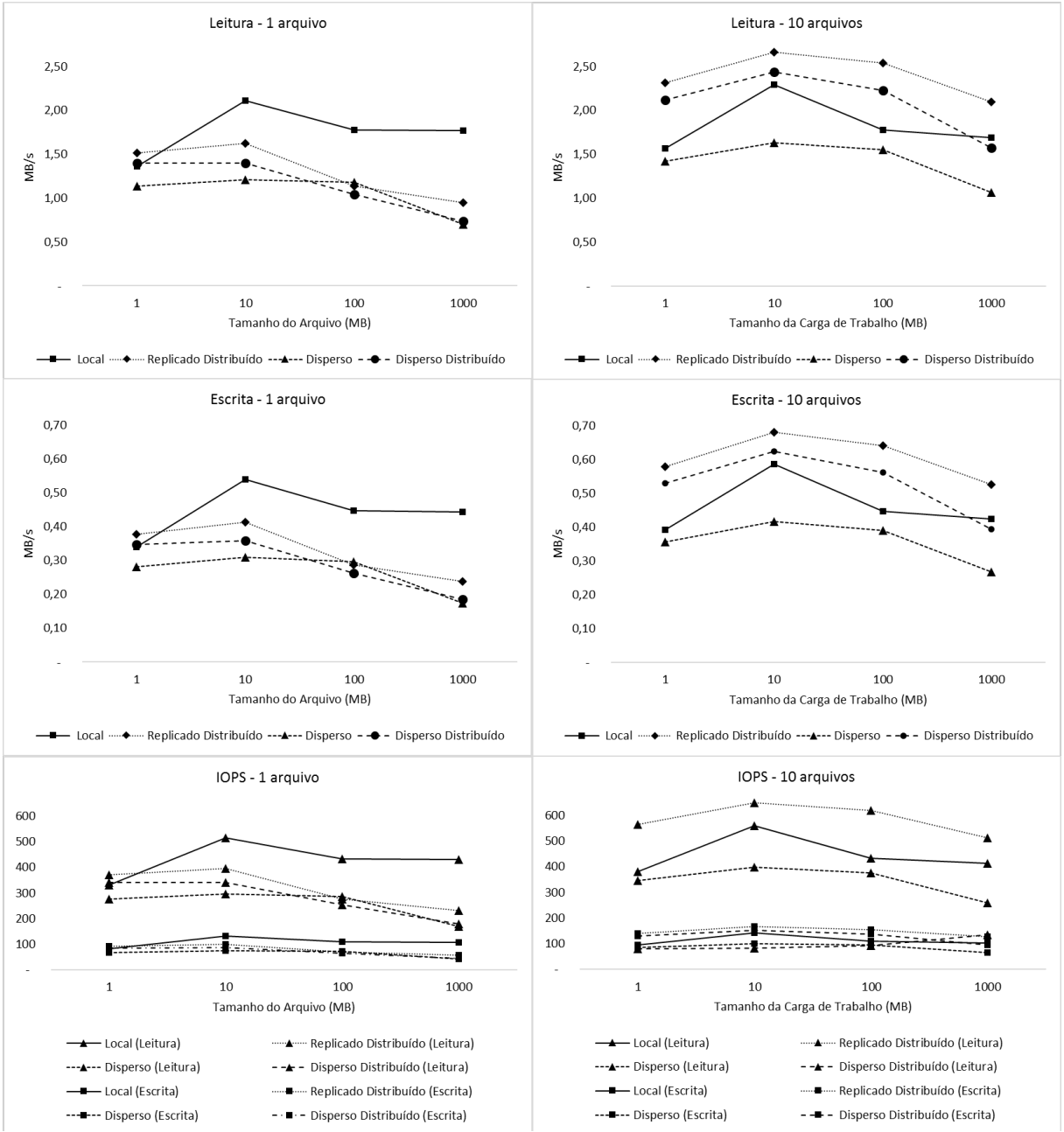

Figura 4.13: Resultados Cenário 2 - Carga de trabalho "Microsoft Web Search" (83% leitura e 17% escrita) usando HDD.

Analisando os resultados do Cenário 2 (Figuras 4.12 e 4.13), que utiliza os servidores do Grupo 2 equipados com discos do tipo HDD, nota-se que o desempenho da arquitetura proposta, em alguns casos, é superior ao obtido pelo servidor do Grupo 3 que utiliza o armazenamento local. No geral as operações com arquivos pequenos (1MB e 10MB) na proposta não obteve desempenho melhor que a arquitetura tradicional de armazenamento local. Nota-se ainda que a quantidade de arquivos sendo operados na arquitetura proposta

tem pouca influência no desempenho, vide que os resultados com 1 e 10 arquivos são próximos.

Entretanto na arquitetura tradicional nota-se um decaimento exponencial na taxa de transferência e IOPS a medida que o tamanho dos arquivos aumenta na carga de trabalho. Enquanto isso na arquitetura proposta o comportamento é linear, indicando que a o desempenho é praticamente o mesmo independente do tamanho de arquivos sendo operados. Esse comportamento pode ser observados na Figura 4.12 onde a carga de trabalho possuía 26,2% leitura e 73,8% escrita. Com arquivos de 1000MB o desempenho da arquitetura tradicional se aproxima bastante do resultado obtido pela arquitetura proposta, podendo-se inferir que com arquivos maiores a proposta tende a ficar com o desempenho melhor que o tradicional em arquivos maiores que 1GB.

Nos resultados descritos na Figura 4.13 onde a carga de trabalho possuía 26,2% leitura e 73,8% escrita, a arquitetura proposta obteve desempenho superior ao armazenamento local ao utilizar cargas de trabalho que manipularam 10 arquivos simultaneamente. Foram observados valores até 48% maiores na taxa de transferência e IOPS (tanto leitura quanto escrita). Esse comportamento indica que a arquitetura proposta possui desempenho superior ao armazenamento local tradicional ao usar cargas de trabalho com alta leitura e alta manipulação de arquivos.

4.4.3 Cenário 3

Como o Cenário 2 obteve melhor desempenho utilizando HDDs, esse grupo de servidores foi escolhido para realização de novos experimentos com uma aplicação real. Nesse caso as aplicações de banco de dados NoSQL Cassandra e MongoDB foram submetidos a testes repetitivos com o cálculo de médias estatísticas e definição de intervalos de confiança.

Nesse cenário foi utilizada a ferramenta YCSB acessando os banco de dados Cassandra e MongoDB e utilizando o grupo 2 de servidores. Cada container executando um banco de dados NoSQL foi carregado com 1.000.000 de registros de 1KB usando a própria ferramenta YCSB. Os containers foram executados em diferentes volumes (replicado distribuído, disperso e disperso distribuído) e a quantidade de conexões simultâneas no banco de dados foi variada entre 1 a 100 em incrementos de 10. Cada uma das opções possíveis foi submetida a 2 cargas de trabalho da Microsoft, 6 do YCSB e 1 carga personalizada com alta carga de escrita. Em cada experimento desse cenários foram realizadas 100.000 transações e média foi calculada com intervalo de confiança de 95%. Os resultados obtidos foram comparados com a mesma carga de trabalho sendo executada no servidor do grupo 3, onde os dados são armazenados diretamente no armazenamento local do servidor.

Nesse cenário o GlusterFS instalado em cada VSA também possibilita a configuração de três tipos diferentes de volumes virtuais, cada um com diferentes desempenhos. Dessa forma os três tipos de volumes descritos nas Figuras 4.3, 4.3 e 4.5 foram avaliados.

Os resultados obtidos nesse cenário pode ser observados nas Figuras 4.14, 4.15 e 4.16. Na Figura 4.14 estão os resultados referentes as 6 cargas de trabalho principais do YCSB submetidas à bancos de dados NoSQL (Cassandra e MongoDB) instalados em 4 tipos diferentes de volumes (local, replicado distribuído, disperso e disperso distribuído) sendo acessados por 11 tipos de acessos simultâneos (1, 10, 20, 30, 40, 50, 60, 70, 80, 90 e 100).

Na Figura 4.15 estão apresentados os resultados referentes as cargas de trabalho personalizadas, baseado nas cargas de trabalho "Cloud Storage" e "Web Search" da Microsoft. Na Figura 4.16 estão apresentados os resultados referente a carga de trabalho personalizada com 100% de escritas.

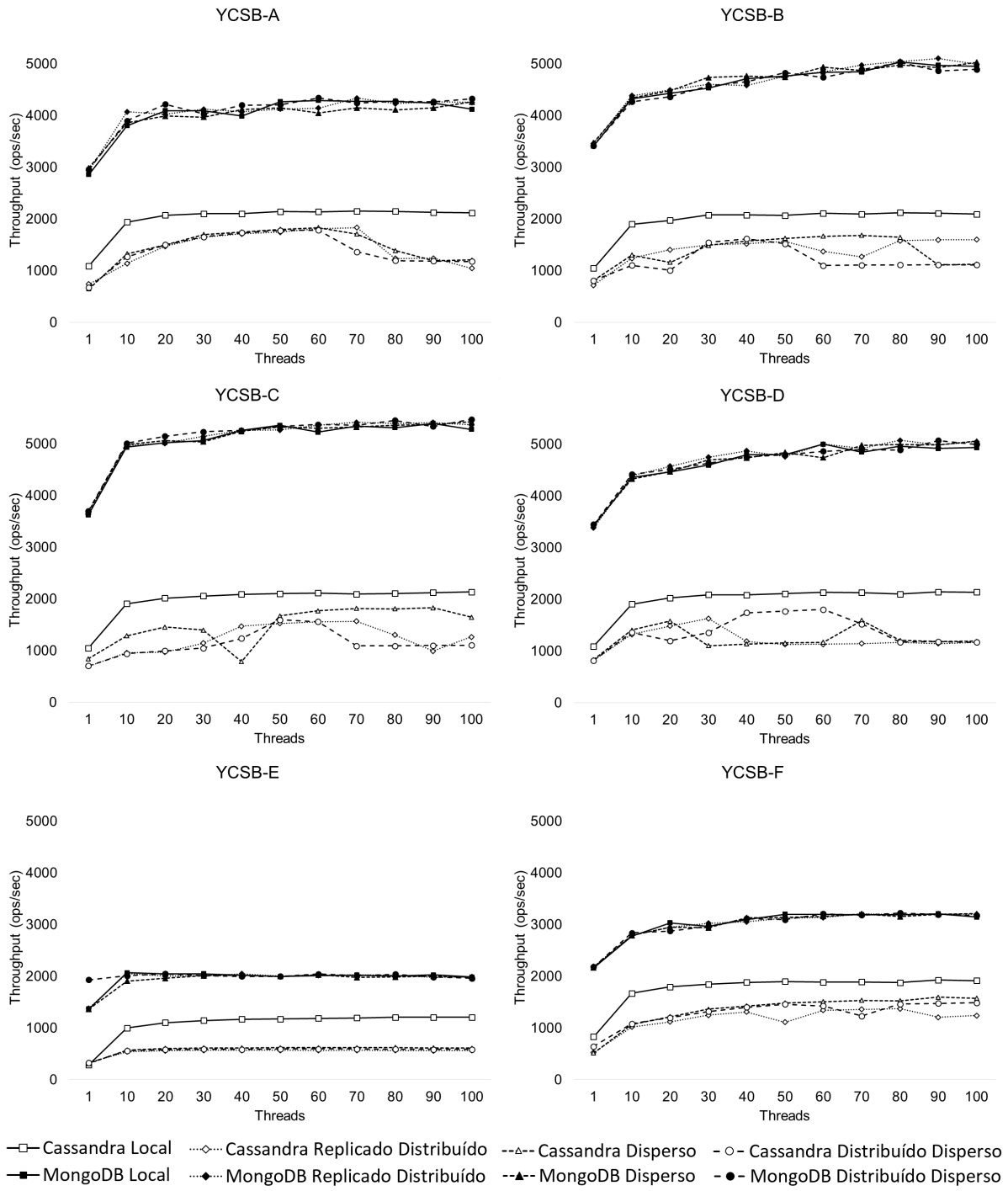

Figura 4.14: Resultados Cenário 3 - Cargas de trabalho do YCSB.

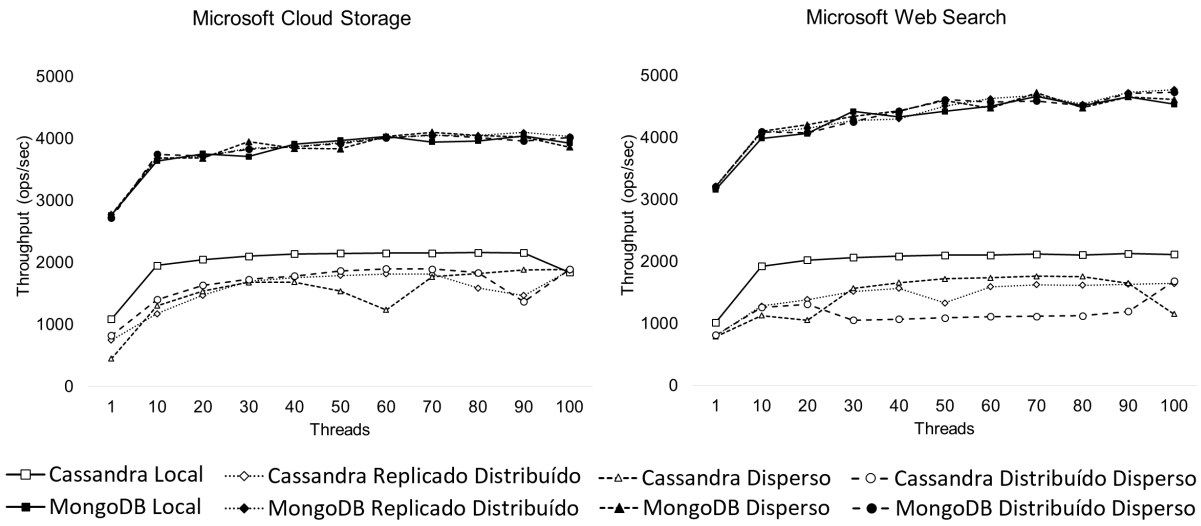

Figura 4.15: Resultados Cenário 3 - Cargas de trabalho da Microsoft.

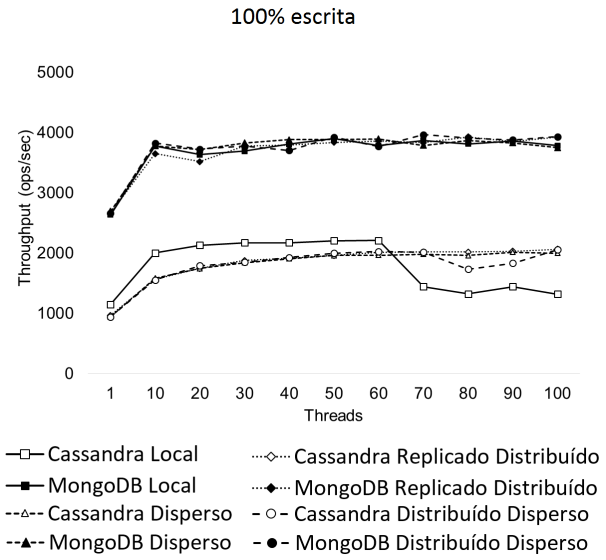

Figura 4.16: Resultados Cenário 3 - Carga de trabalho com alta taxa de escrita.

Conforme pode ser observado nos resultados expostos nas Figuras 4.14, 4.15 e 4.16, o MongoDB apresentou melhor desempenho que o Cassandra em todos os experimentos. O MongoDB executando na arquitetura proposta apresentou desempenho equivalente ao armazenamento tradicional nos discos locais do servidor, independente do número de acessos simultâneos ao banco de dados. Em alguns casos específicos a arquitetura proposta obteve desempenho até 7% melhor que o de uma arquitetura tradicional de armazenamento. Os resultados obtidos podem ser considerados bons, visto que mesmo com as abstrações

na camada de armazenamento imposta pela arquitetura hiperconvergente, o desempenho foi similar ao de um sistema onde os dados são salvos direto nos discos conectados aos servidores.

O desempenho do Cassandra na arquitetura proposta foi abaixo das expectativas em quase todos os experimentos. O único experimento onde a arquitetura proposta obteve melhor desempenho que o uso do armazenamento local foi na Carga de Trabalho de Alta Carga de Escrita (Figura 4.16), onde nos experimentos com mais de 70 acessos simultâneos a arquitetura proposta obteve até 56% melhor desempenho que o armazenamento local.

A diferença no desempenho do Cassandra e MongoDB é explicada pela maneira que esses banco de dados NoSQL salvam os dados no sistema de arquivos. Enquanto o MongoDB salva os dados em arquivos grandes, o Cassandra salva os dados em arquivos pequenos. Diferenças de desempenho entre operações com arquivos pequenos e grandes são conhecidas na literatura e deve-se ao fato do *overhead* imposto pelos sistemas de arquivos ao operar pequenos arquivos [4] [50].

Nota-se também que o desempenho do Cassandra é afetado pelo tipo de volume utilizado no GlusterFS. Diferenças significativas na quantidade de operações por segundos na mesma quantidade de conexões simultâneas podem ser observadas nos resultados das cargas de trabalho YCSB-B, C e D (Figura 4.14). Esse comportamento indica que cargas de trabalho que operam arquivos pequenos no GlusterFS possuem desempenhos diferentes dependendo da configuração de discos usada no volume.

4.4.4 Cenário 4

Nesse cenário foi utilizada a ferramenta FIO e o grupo 4 de servidores para avaliar o desempenho da escalabilidade da arquitetura a medida que a quantidade de discos e servidores aumenta. As 11 cargas de trabalho descritas na Tabela 4.2 foram executadas em 18 volumes diferentes, que variaram entre 1 a 9 servidores e 1 a 18 discos. Nesse cenário o GlusterFS instalado em cada VSA possibilita a configuração de um volume distribuído altamente escalável, onde os arquivos são aleatoriamente alocados nos discos que compõe o volume, vide Figura 4.17. A alocação de servidores e discos por volume pode ser observada na Tabela 4.3. Nessa tabela a primeira coluna indica o número do volume e as colunas subsequentes indicam quais discos de quais servidores foram alocados para construir o volume. Em cada execução da carga de trabalho em cada volume foram realizadas 500 operações em disco e média foi calculada com intervalo de confiança de 95%. Os resultados obtidos foram comparados com a mesma carga de trabalho sendo executada no servidor do grupo 3, onde os dados são armazenados diretamente no armazenamento local do servidor.

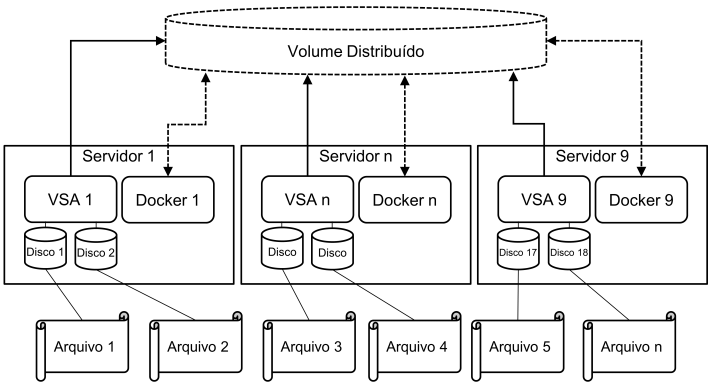

Figura 4.17: Volume Distribuído.

Tabela 4.3: Alocação de Servidores e Discos por Volume

Servidor →	1		2		3		4		5		6		7		8		9	
Disco → Volume ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	X																	
2	X		X															
3	X		X		X													
4	X		X		X		X											
5	X		X		X		X		X									
6	X		X		X		X		X		X							
7	X		X		X		X		X		X		X					
8	X		X		X		X				X		X		X			
9	X		X		X		X		X		X		X		X		X	
10	X	X	X		X		X		X		X		X		X		X	
11	X	X	X	X	X		X		X		X		X		X		X	
12	X	X	X	X	X	X	X		X		X		X		X		X	
13	X	X	X	X	X	X	X	X	X		X		X		X		X	
14	X	X	X	X	X	X	X	X	X	X	X		X		X		X	
15	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	
16	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	
17	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
18	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Os resultados obtidos nesse cenário pode ser observados na Figura 4.18. Nessa figura estão os resultados referentes a execução das 11 cargas de trabalhos (vide Tabela 4.2), sendo que cada uma foi executada em 18 volumes diferentes (vide Tabela 4.3).

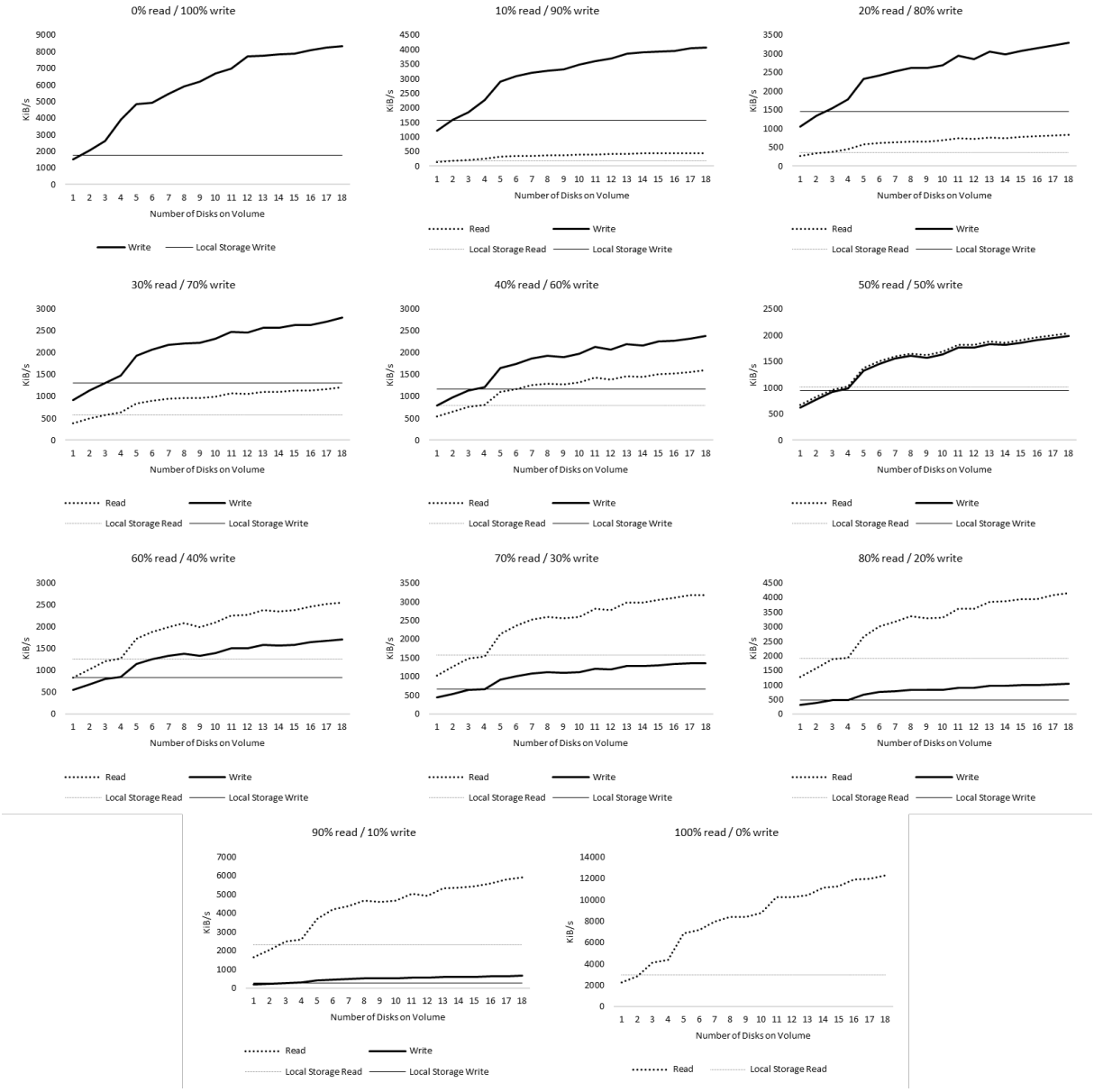

Figura 4.18: Resultados Cenário 4 - Cargas de trabalho com diversos padrões de I/O.

Os resultados obtidos no Cenário 4 demonstram que a arquitetura proposta é escalável e que o desempenho aumenta a medida que o número de servidores e discos cresce, conforme pode ser observado na tendência de crescimento nos gráficos da Figura 4.18. Calculando-se a média de todos os resultados obtidos no Cenário 4 para cada tipo de volume é possível obter o comportamento médio da arquitetura proposta. Analisando esses valores médios em função das cargas de trabalho verifica-se que no geral a arquitetura proposta possui melhor desempenho com cargas de trabalho que possuem maior quantidade de leitura, conforme pode ser observado nos gráficos da Figura 4.19. Ainda

analisando esses valores médios é possível calcular um regressão em função da quantidade de discos por volume, conforme apresentado na Figura 4.20, visando validar a coerência dos resultados. A regressão potência foi a que melhor se adequou a taxa de leitura e escrita, obtendo um $R^2 = 0,97$ e a análise do expoente confirma o crescimento da taxa de transferência a medida que aumenta-se a quantidade de discos no volume.

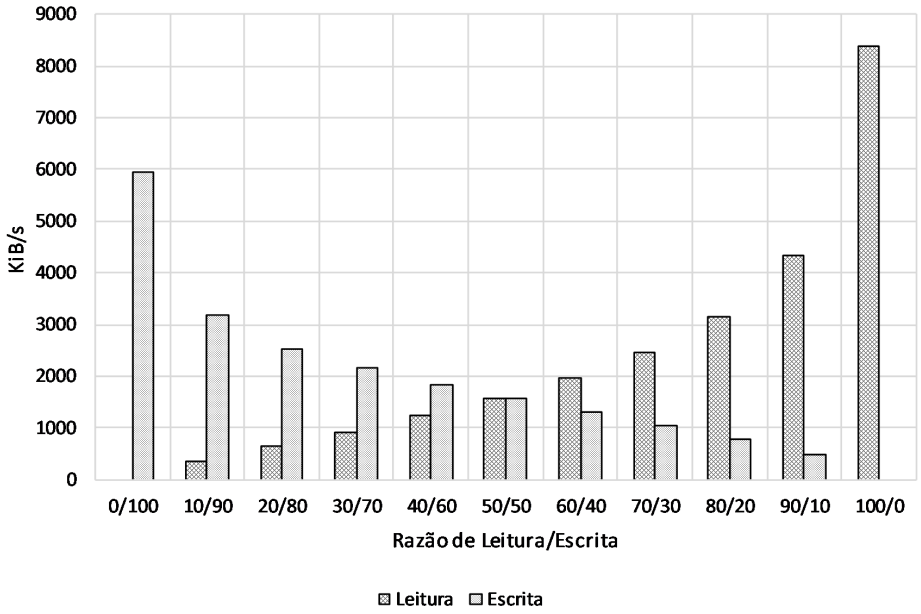

Figura 4.19: Resultados médios do cenário 4 em função das cargas de trabalho.

Por fim, cabe mencionar também que durante a execução dos experimentos não foram observados congestionamentos na rede de comunicação, alto processamento e nem alto consumo de memória, indicando que esses parâmetros não interferiram nos resultados dos experimentos.

4.5 Viabilidade da Hiperconvergência em Ambientes de Nuvem

Umas das características da hiperconvergência é possibilitar agilidade em nuvens públicas e privadas. Um dos dilemas para consumidores de serviços em nuvem é decidir entre mover aplicativos e armazenamento para uma nuvem pública ou mantê-los em uma nuvem privada. Existem muitos parâmetros que precisam ser analisados para tomar esse tipo de decisão, sendo o custo um elemento importante, mas não o único. A agilidade e flexibilidade das nuvens públicas são atraentes porque um consumidor pode desenvolver aplicativos ou implantar um modelo de DevOps para uma liberação mais rápida de

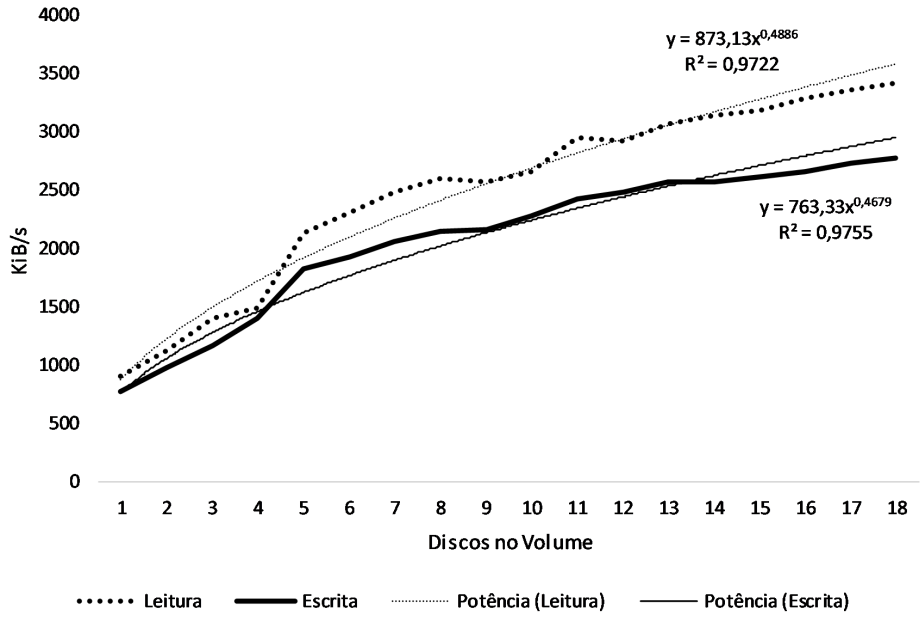

Figura 4.20: Resultados médios do cenário 4 em função da quantidade de discos no volume.

recursos e funcionalidades. Entretanto por muitas vezes consumidores precisam seguir processos complexos para adquirir hardware e software e integrar os componentes de uma arquitetura tradicional, que os prazos de entregas das aplicações acabam sendo comprometidos. A hiperconvergência permite que esses consumidores de serviços em nuvem tenham esse tipo de agilidade em sua nuvem privada. O tempo de instalação e provisionamento de uma plataforma hiperconvergente é mais rápido do que em comparação com uma arquitetura tradicional em razão de basicamente usar "servidores" em vez de "servidores mais equipamentos dedicados de armazenamento interligados por uma rede SAN". Nesse caso um provedor de serviços em nuvem pode usar sua infraestrutura hiperconvergente como um *pool* dinâmico de recursos que são usado por muitos consumidores. Cada consumidor pode ter seus próprios recursos para trabalhar e esses podem voltar ao *pool* principal assim que o projeto é concluído. Isso representa uma economia de custos ao evitar as taxas de nuvem pública, permitindo investir em outros projetos.

Outra característica é uma maior disponibilidade com custos menores. Sistemas hiperconvergentes são arquitetados para alta disponibilidade, o que significa que a arquitetura prevê que componentes falharão e os dados e aplicativos devem ser protegidos. A disponibilidade de sistema hiperconvergente é "de-muitos-para-muitos", e não "de-um-para-um", como em equipamento dedicado tradicional de armazenamento (*storage array*). Nesses equipamentos tradicionais é comum existir duas controladoras para o caso de uma falhar a outra continuar com as operações em disco. Acrescido a essa camada de tole-

rância a falhas, em um equipamento dedicado os volumes de armazenamento são criados agrupando-se discos nas tradicionais configurações de RAID 5 ou RAID 6. No entanto em sistemas que utilizam esse modelo tradicional de armazenamento não existe proteção para falhas ocorridas entre as conexões entre um servidor e o equipamento de armazenamento de dados, exigindo o uso de uma nova camada de proteção nas conexões. Isso significa que para ter alta disponibilidade se faz necessário usar conexões duplas entre servidores e unidades de armazenamento acrescidos de softwares para gerenciar múltiplos caminhos até o volume de dados (*multipath*). Mesmo com todas essas camadas de tolerância a falhas, quando uma controladora de um sistema tradicional falha, 100% da carga de operações em disco vai para a outra controladora. Durante esse momento os aplicativos podem sofrer um grande impacto no desempenho se os controladores individuais estiverem sendo executados com mais de 50% de carga. Basicamente, alcançar alta disponibilidade em projetos de armazenamento com equipamentos dedicados tradicionais é caro. Como um sistema hiperconvergente é uma arquitetura distribuída, os dados são protegidos por padrão, sendo replicados para vários servidores e discos. A proteção é "de muitos-para-muitos", o que significa que, após uma falha em um servidor, o restante dos servidores absorvem a carga e continuam provendo acesso aos dados. Além disso sistemas hiperconvergentes se reequilibram automaticamente quando um servidor ou capacidade é adicionado ou removido. Portanto, os sistemas hiperconvergentes são altamente disponíveis e possuem uma estrutura de custos melhor.

Infraestruturas hiperconvergentes também possuem baixo custo para inicio de uma operação. Um sistema hiperconvergente podem começam com dois ou três servidores e crescerem para centenas de servidores. Isso elimina a tomada de decisões que provedores de serviços em nuvem devem fazer na escolha de quão grande ou pequeno um equipamento dedicado tradicional de armazenamento precisa ser. Uma estrutura de baixo custo de entrada encoraja consumidores a experimentar novos aplicativos e crescer conforme necessário. Arquitetura tradicionais são comercializadas geralmente em configurações pequenas, médias e grandes, com diferenças de custo entre diferentes configurações e uma grande parcela do custo sendo alocada para o equipamento de armazenamento. Isso não quer dizer que unidades dedicadas de armazenamento não possam ser atualizadas, mas longos ciclos de atualização e o custo da atualização poderiam influenciar provedores de serviços em nuvem a obter apenas um novo equipamento de maior capacidade. Nessa linha de atuação o provedor investiria em um novo equipamento com capacidade extra que poderia permanecer inativa por anos apenas para evitar os ciclos de atualização. Em uma arquitetura hiperconvergente, os sistemas custam menos por natureza, porque são construídos em arquitetura x86. Além disso, o sistema pode começar pequeno e ser suavemente atualizado adicionando mais servidores. Quando servidores são adicionados, o

sistema se reequilibra para aproveitar o armazenamento, a memória e as processamento adicionados.

A custo total de propriedade é uma medida de gastos com investimento e despesas operacionais durante um período de tempo. Normalmente em um provedor de serviços em nuvem esse cálculo é baseado em um período médio de 3 a 5 anos, que é o período tradicional de ciclos de atualização de hardware e software. Os gastos com investimento em um provedor de serviços em nuvem para o ambiente de armazenamento de dados é calculado com base nas compras de hardware para servidores, unidades de armazenamento dedicadas, equipamentos de rede, licenças de software, gastos com energia, instalações e assim por diante. Estudos demonstraram que a economia em gastos de investimento ao usar sistemas hiperconvergentes em vez de sistemas tradicionais está na faixa de 30%, considerando economias em hardware, software e energia [51].

Além disso, a facilidade de implantação e operação e o conhecimento geral existente das tecnologias envolvidas em um ambiente hiperconvergente podem resultar em cerca de 50% de economia em despesas operações [51]. Esse número varia amplamente dependendo de quais produtos são comparados. Em resumo conclui-se que hiperconvergência executada em sistemas baseados em x86 são mais econômicos do que as soluções tradicionais que utilizam equipamentos dedicados para armazenamento, agregando os benefícios de ciclos de implementação mais curtos e maior receita gerada pela com eficiência na alocação de recursos para consumidores.

Com a popularização da hiperconvergência, empresas com serviços em nuvem privada estão percebendo que com produtos melhor integrados e melhores ferramentas de automação é possível duplicar o modelo de nuvem pública para seus consumidores. Dessa forma não é preciso pagar por serviços em nuvens públicas se uma nuvem privada hiperconvergente permite iniciar servidores virtuais ou containers em minutos, alocar armazenamento sob demanda, compartilhar a infraestrutura com vários departamentos, pagar conforme você cresce, automatizar tarefas ou duplicar um modelo em diferentes regiões. Uma nuvem privada hiperconvergente nesse contexto se assemelha com um *datacenter* comum, porem com uma integração melhores, uso de armazenamento definido por software e um novo modelo de consumo.

O uso de serviços de nuvem pública ou privada não são mutualmente exclusivos. Em determinados casos é possível utilizar a noção de "nuvens híbridas" o que significa que alguns dos serviços pode ser executados em nuvem privada e alguns outros serviços pode ser executados em uma nuvem pública. Um aplicativo pode ser executado em uma nuvem privada hiperconvergente e seu armazenamento pode estar em uma nuvem pública. Uma empresa pode ter sua própria infraestrutura de hiperconvergência executando serviços de computação, rede e armazenamento, aproveitando recursos de provedores públicos tais

como Amazon Web Services (AWS), Azure ou Google Cloud Platform. Um exemplo de tais serviços é mineração de dados, onde a empresa terceirizar os serviços para a nuvem pública por uma fração do custo de construí-los internamente.

4.6 Conclusão do Capítulo

Este capítulo descreveu a validação experimental da arquitetura proposta com diferentes configurações e intensidades de carga de trabalho. Foram analisadas as métricas de taxa de escrita, taxa de leitura, quantidade de operações por segundo (IOPS) e quantidade de operações de banco de dados por segundo e quantidade de discos por volume, que demonstraram a qualidade da proposta apresentada. A arquitetura proposta neste trabalho foi comparada com uma arquitetura tradicional de armazenamento de dados (não convergente). A arquitetura proposta usando discos HDD conseguiu melhor desempenho ao operar arquivos grandes, mantendo uma tendência linear a medida que o tamanho dos arquivos aumenta, sendo que na arquitetura tradicional o mesmo experimento observou-se um queda exponencial no desempenho. Observou-se também que a arquitetura proposta obtem um desempenho superior ao armazenamento local ao usar cargas de trabalho com alta taxa de escrita e alta manipulação de arquivos, bons resultados nos experimentos com o banco de dados NoSQL MongoDB e escalabilidade com desempenho crescente. Por fim discorreu-se sobre a viabilidade e aplicabilidade da proposta apresentada em provedores de serviços em nuvem, tanto privados quanto públicos.

Capítulo 5

Conclusões e Trabalhos Futuros

Esse trabalho apresentou uma proposta de arquitetura hiperconvergente para armazenamento de dados de *containers*, com foco na aplicação em provedores de serviços em nuvem, tanto privados quanto públicos.

A computação em nuvem possui características que a tornam interessantes para empresas em geral. A possibilidade do uso dessas características em infraestrutura própria ou de terceiros para prover serviços, pode diminuir custos com operação, compra de equipamentos e software. Além disso, funcionalidades como escalabilidade pode otimizar custos, pois recursos podem escalados de acordo com a demanda.

Os paradigmas de nuvem privada também trazem ganhos para empresas a medida em que são capazes de otimizar recursos, ao agregar os existentes que não são sendo utilizados em sua plena capacidade, traduzindo-se em ganhos operacionais e financeiros.

Neste sentido, foi apresentado nesse trabalho uma arquitetura hiperconvergente para provisão de área de armazenamento de dados, tolerante a falhas e escalável, utilizando os recursos locais de armazenamento de diversos servidores distribuídos em um *datacenter*. Durante a pesquisa do estado da arte de ambientes de computação distribuídos, verificou-se que maioria dos trabalhos propunha o uso de nós dedicados para o armazenamento de dados, ignorando o fato que cada nó também possui recursos de computação que podem ser utilizados.

A proposta deste trabalho é agregar os recursos de armazenamento e computação em um único sistema, com foco na consolidação dos recursos individuais de armazenamento para uma plataforma de *containers*. Para atingir esse objetivo foram integrados hipervisores, sistema de arquivos distribuídos e *containers*. Isso trouxe um caráter inovador para a proposta, pois foram integradas ferramentas de diversos domínios, que atuaram de forma a trazer as funcionalidades idealizadas.

Um sistema hiperconvergente adiciona recursos desejáveis para sistemas de computação em nuvem quando comparado ao armazenamento convencional, tais como esca-

labilidade, elasticidade e tolerância a falhas. Um sistema hiperconvergente pode crescer facilmente adicionando mais servidores, que podem ter diferentes configurações de computação e armazenamento. Enquanto isso em uma arquitetura tradicional, novos servidores individuais podem ser adicionados à infraestrutura, mas os recursos de computação e armazenamento estarão confinados em cada servidor.

A proposta foi testada, usando cargas de trabalho simples, com as quais a viabilidade da proposta foi confirmada, e também com cargas complexas, com as quais também se obteve bom desempenho. Os testes com a ferramenta foram extensos, sendo que em alguns cenários, a arquitetura foi submetida a cargas durante mais de uma semana, vinte e quatro horas por dia. Isso demonstra a robustez da proposta, mesmo essa ainda estando na fase de protótipo.

A arquitetura proposta foi comparada com uma arquitetura padrão de mercado onde os dados são armazenados localmente em cada servidor e os resultados dos testes comparativos demonstraram o potencial da proposta, com a provisão de uma área de armazenamento escalável e tolerante a falhas.

Com base nos resultados obtidos pode-se concluir que o uso de uma infraestrutura hiperconvergente é viável para provedores de serviços de nuvem na prestação de serviços de IaaS (por exemplo, máquinas virtuais e *containers*) e serviços de PaaS (por exemplo, bancos de dados em *containers*). Uma aplicação de usuário que requerer mais recursos de armazenamento do que um único servidor pode oferecer tem desafios para dimensionada em um sistema não convergente, mas teria os recursos alocados facilmente em uma arquitetura hiperconvergente. Esse tipo de recurso é desejado por *datacenters* de provedores em nuvem, pois otimiza a infraestrutura, permitindo que dois tipos de recursos (computação e armazenamento) se colapsem em um, aumentando a eficiência.

5.1 Trabalhos Futuros

A sequência deste trabalho pode ir no caminho da análise de outros hipervisores, cluster maiores, diferentes configurações de servidor e outras cargas de trabalho. Outra evolução interessante seria a integração desta proposta com ferramentas de orquestração de servidores virtuais e *containers*, tornando a proposta um *pool* de recursos de armazenamento e computação.

A implantação desta proposta em um ambiente de produção será importante para a verificação de possíveis questões que não ficaram evidentes durante os testes realizados com cargas sintéticas, ainda que estas fossem criadas a partir de cargas de trabalho reais.

A avaliação deste trabalho foi feita em um ambiente privado e limitado, portanto, seria interessante a realização de testes com esta proposta em um ambiente maior. Neste caso,

pode ser feito um estudo que verificasse a economia financeira ao utilizar a proposta em uma escala maior.

Referências

[1] Tarasov, Vasily, Lukas Rupprecht, Dimitris Skourtis, Amit Warke, Dean Hildebrand, Mohamed Mohamed, Nagapramod Mandagere, Wenji Li, Raju Rangaswami e Ming Zhao: *In search of the ideal storage configuration for docker containers*. Em *Foundations and Applications of Self* Systems (FAS* W), 2017 IEEE 2nd International Workshops on*, páginas 199–206. IEEE, 2017. 3, 24, 25, 28

[2] Xu, Qiumin, Manu Awasthi, Krishna T Malladi, Janki Bhimani, Jingpei Yang e Murali Annavaram: *Docker characterization on high performance ssds*. Em *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*, páginas 133–134. IEEE, 2017. 3, 24, 25, 28

[3] Xu, Qiumin, Manu Awasthi, Krishna T Malladi, Janki Bhimani, Jingpei Yang e Murali Annavaram: *Performance analysis of containerized applications on local and remote storage*. Em *Proc. of MSST*, 2017. 3, 24, 28

[4] Roch, Loïc M, Tyanko Aleksiev, Riccardo Murri e Kim K Baldridge: *Performance analysis of open-source distributed file systems for practical large-scale molecular ab initio, density functional theory, and gw+ bse calculations*. International Journal of Quantum Chemistry, 118(1), 2018. 3, 14, 27, 34, 60

[5] Mills, Nicholas, F Alex Feltus e Walter B Ligon III: *Maximizing the performance of scientific data transfer by optimizing the interface between parallel file systems and advanced research networks*. Future Generation Computer Systems, 79:190–198, 2018. 3, 16, 27

[6] Cheng, Wei Hsun, Chun I Chiang, Chao Tung Yang, Shuo Tsung Chen e Jung Chun Liu: *The implementation of supporting uniform data distribution with software-dened storage service on heterogeneous cloud storage*. Em *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*, páginas 610–615. IEEE, 2016. 3, 28

[7] Thereska, Eno, Hitesh Ballani, Greg O'Shea, Thomas Karagiannis, Antony Rowstron, Tom Talpey, Richard Black e Timothy Zhu: *Ioflow: a software-defined storage architecture*. Em *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, páginas 182–196. ACM, 2013. 3, 26, 27

[8] Maenhaut, Pieter Jan, Hendrik Moens, Bruno Volckaert, Veerle Ongenae e Filip De Turck: *A dynamic tenant-defined storage system for efficient resource management in cloud applications*. Journal of Network and Computer Applications, 93:182–196, 2017. 3, 22, 23, 26, 27

[9] Ravandi, Babak e Ioannis Papapanagiotou: *A self-learning scheduling in cloud software defined block storage.* Em *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*, páginas 415–422. IEEE, 2017. 3, 27

[10] Azagury, Alain C, Robert Haas, Dean Hildebrand, Steven W Hunter, Todd Neville, Sven Oehme e Anees Shaikh: *Gpfs-based implementation of a hyperconverged system for software defined infrastructure.* IBM Journal of Research and Development, 58(2/3):6–1, 2014. 3, 19, 21, 25, 36

[11] Park, Sun, Byungrae Cha e Jongwon Kim: *Preparing and inter-connecting hyperconverged smartx boxes for iot-cloud testbed.* Em *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, páginas 695–697. IEEE, 2015. 3, 25

[12] Verma, Shikhar, Yuichi Kawamoto, Zubair Md Fadlullah, Hiroki Nishiyama e Nei Kato: *A survey on network methodologies for real-time analytics of massive iot data and open research issues.* IEEE Communications Surveys & Tutorials, 19(3):1457–1477, 2017. 3, 19, 20, 21, 25

[13] Mell, Peter M. e Timothy Grance: *Sp 800-145. the nist definition of cloud computing.* 2011. 5, 6, 7

[14] Badger, Lee, Tim Grance, Robert Patt-Corner, Jeff Voas *et al.*: *Cloud computing synopsis and recommendations.* NIST special publication, 800:146, 2012. 6, 7

[15] Yang, Chao Tung, Wen Chung Shih, Chih Lin Huang, Fuu Cheng Jiang e William Cheng Chung Chu: *On construction of a distributed data storage system in cloud.* Computing, 98(1-2):93–118, 2016. 8

[16] García-Valls, Marisol, Tommaso Cucinotta e Chenyang Lu: *Challenges in real-time virtualization and predictable cloud computing.* Journal of Systems Architecture, 60(9):726–740, 2014. 8, 9, 11, 12

[17] Sahoo, Jyotiprakash, Subasish Mohapatra e Radha Lath: *Virtualization: A survey on concepts, taxonomy and associated security issues.* Em *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, páginas 222–226. IEEE, 2010. 8

[18] Rimal, Bhaskar Prasad, Eunmi Choi e Ian Lumb: *A taxonomy and survey of cloud computing systems.* Em *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, páginas 44–51. IEEE, 2009. 8

[19] Ogunyemi, Abiodun e Kevin Johnston: *Is server virtualization implementation in business and public organizations a worthwhile investment?* International Journal of Information Technology & Decision Making, 16(03):711–736, 2017. 10

[20] Veras, Manoel e Alexandre Carissimi: *Virtualização de Servidores.* Escola Superior de Redes, 2015. 11, 12, 16, 17, 35

[21] Wang, Li e Yunchuan Wen: *Design and implementation of ceph block device in userspace for container scenarios*. Em *Computer, Consumer and Control (IS3C), 2016 International Symposium on*, páginas 383–386. IEEE, 2016. 12

[22] Liu, Wei, Weibei Fan, Peng Li e Liangde Li: *Survey of big data platform based on cloud computing container technology*. Em *Conference on Complex, Intelligent, and Software Intensive Systems*, páginas 954–963. Springer, 2017. 12

[23] Peinl, René, Florian Holzschuher e Florian Pfitzer: *Docker cluster management for the cloud-survey results and own solution*. Journal of Grid Computing, 14(2):265–282, 2016. 12

[24] Docker: *The definitive guide to docker*. `https://www.innovate-systems.de/downloads/docker/The_definitive_Guide_to_Docker.pdf`, 2018. Acessado em 04/02/2019. 13, 35

[25] Rolf Neugebauer: *Linux Containers on Windows*. `https://blog.docker.com/2017/09/preview-linux-containers-on-windows/`, 2017. Acessado em 11/07/2018. 13

[26] Thanh, Tran Doan, Subaji Mohan, Eunmi Choi, SangBum Kim e Pilsung Kim: *A taxonomy and survey on distributed file systems*. Em *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 1, páginas 144–149. IEEE, 2008. 14, 15, 16

[27] Kaneko, Shun, Takaki Nakamura, Hitoshi Kamei e Hiroaki Muraoka: *A guideline for data placement in heterogeneous distributed storage systems*. Em *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*, páginas 942–945. IEEE, 2016. 16, 35

[28] Docker: *About storage drivers*. Relatório Técnico, Docker, 2018. 18, 24, 35

[29] Cha, ByungRae, Sun Park e JongWon Kim: *Design and verification of software-defined raid for hybrid cloud storage*. Em *Cloud Computing Research and Innovations (ICCCRI), 2016 International Conference on*, páginas 128–133. IEEE, 2016. 21, 23

[30] Huang, Ming Jen, Chun Fang Huang e Wen Shyen Eric Chen: *Architecting a software-defined storage platform for cloud storage service*. Em *Services Computing (SCC), 2015 IEEE International Conference on*, páginas 379–386. IEEE, 2015. 22

[31] Bokare, Shreya, Sanjay Pawar e Shikha Nema: *Evaluation of sds controller (corphd) for various storage demands*. Em *Software Defined Systems (SDS), 2017 Fourth International Conference on*, páginas 26–31. IEEE, 2017. 22, 23

[32] Jararweh, Yaser, Mahmoud Al-Ayyoub, Elhadj Benkhelifa, Mladen Vouk, Andy Rindos *et al.*: *Software defined cloud: Survey, system and evaluation*. Future Generation Computer Systems, 58:56–74, 2016. 22, 23

[33] Darabseh, Ala, Mahmoud Al-Ayyoub, Yaser Jararweh, Elhadj Benkhelifa, Mladen Vouk e Andy Rindos: *Sdstorage: a software defined storage experimental framework*. Em *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, páginas 341–346. IEEE, 2015. 22, 23

[34] Yang, Chao Tung, Wei Hsiang Lien, Yu Chuan Shen e Fang Yi Leu: *Implementation of a software-defined storage service with heterogeneous storage technologies*. Em *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*, páginas 102–107. IEEE, 2015. 23, 27

[35] The New Stack: *Understanding Software-Defined Storage*. `https://thenewstack.io/understanding-software-defined-storage/`, 2018. Acessado em 15/05/2018. 26

[36] Wikipedia: *Software-Defined Storage*. `https://en.wikipedia.org/wiki/Software-defined_storage`, 2018. Acessado em 15/05/2018. 26

[37] NetApp: *What Is Software-Defined Storage (SDS)?* `https://www.netapp.com/us/info/what-is-software-defined-storage.aspx`, 2018. Acessado em 15/05/2018. 26

[38] Yottabyte: *Software-Defined Storage*. `http://www.yottabyte.com/technologies/software-defined-storage/`, 2018. Acessado em 20/03/2018. 26

[39] Kontzer, Tony: *What's holding up the adoption of containers?* `http://www.baselinemag.com/storage/slideshows/whats-holding-up-the-adoption-of-containers.html`, 2017. Acessado em 04/02/2019. 28

[40] Fattahi, Tahereh e Reza Azmi: *A new approach for directory management in glusterfs*. Em *Information and Knowledge Technology (IKT), 2017 9th International Conference on*, páginas 166–174. IEEE, 2017. 31, 34

[41] Martin, Nick: *A brief history of docker containers' overnight success*. `https://searchservervirtualization.techtarget.com/feature/A-brief-history-of-Docker-Containers-overnight-success`, 2015. Acessado em 04/02/2019. 35

[42] Huang, Jian, Anirudh Badam, Laura Caulfield, Suman Nath, Sudipta Sengupta, Bikash Sharma e Moinuddin K Qureshi: *Flashblox: Achieving both performance isolation and uniform lifetime for virtualized ssds.* Em *FAST*, páginas 375–390, 2017. 39, 40, 45, 53

[43] Cooper, Brian F, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan e Russell Sears: *Benchmarking cloud serving systems with ycsb*. Em *Proceedings of the 1st ACM symposium on Cloud computing*, páginas 143–154. ACM, 2010. 39, 41

[44] Axboe, Jens: *Fio-flexible io tester*. `http://freecode.com/projects/fio`, 2014. Acessado em 04/02/2019. 40, 42

[45] Apache Cassandra: *Apache Cassandra*. `http://cassandra.apache.org`, 2018. Acessado em 04/02/2019. 41

[46] MongoDB: *MongoDB*. `https://www.mongodb.com`, 2018. Acessado em 04/02/2019. 41

[47] Jain, Raj: *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* John Wiley & Sons, 1990. 43

[48] Kim, Deoksang, Hyeonsang Eom e Heonyoung Yeom: *Performance optimization in glusterfs on ssds.* KIISE Transactions on Computing Practices, 22(2):95–100, 2016. 52, 53

[49] Oh, Myoungwon, Sejin Park, Jugwan Eom, Seungmin Kim, Sangjae Kim, Kang won Lee e Heon Y Yeom: *Lalca: Locality-aware lock contention avoidance for nvme-based scale-out storage system.* Em *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, páginas 1143–1152. IEEE, 2018. 52

[50] Vaidya, Madhavi e Shrinivas Deshpande: *Comparative analysis of various distributed file systems & performance evaluation using map reduce implementation.* Em *Recent Advances and Innovations in Engineering (ICRAIE), 2016 International Conference on*, páginas 1–6. IEEE, 2016. 60

[51] Halabi, Sam: *Hyperconverged Infrastructure Data Centers: Demystifying HCI.* Cisco Press, 2019. 66

ANEXO 1

Artigo publicado no CLOSER 2019

(9th International Conference on Cloud Computing and Services Science)

Heraklion / Grécia

Performance Analysis of an Hyperconverged Infrastructure using Docker Containers and GlusterFS

Rodrigo Leite[1], Priscila Solis[1], Eduardo Alchieri[1]
[1]*Applied Computing Masters Program, Department of Computer Science, University of Brasilia (UnB), Braslia, Brazil*
rodrigo123@gmail.com, pris@unb.br, alchieri@unb.br

Keywords: Hyperconverged Infrastructures, GlusterFS, Containers, DFS, Cassandra, MongoDB, Docker, Cloud, Storage

Abstract: The adoption of hyperconverged infrastructures is a trend in datacenters, because it merges different type of computing and storage resources. Hyperconverged infrastructures use distributed file systems (DFS) to store and replicate data between multiple servers while using computing resources of the same servers to host virtual machines or containers. In this work, the distributed file system GlusterFS and the hypervisor VMware ESXi are used to build an hyperconverged system to host Docker containers, with the goal of evaluate the storage performance of this system compared to traditional approach where data is stored directly on the server's disks. The performance of the container's persistent storage is evaluated using the benchmark tool Yahoo Cloud Service Benchmark (YCSB) against the NoSQL databases Cassandra and MongoDB under differents workloads. The NoSQL database's performance was compared between the hyperconverged system with multiples disk configurations and a traditional system with local storage.

1 INTRODUCTION

In the last years, the amount of data generated by different devices and users is growing constantly and exponentially. This brings a challengue to the traditional storage solutions that need to cope with massive data storage and processing. For this, in the last years several large-scale Distributed File Systems (DFS) were developed to overcome these limitations. Some benefits of DFS are scalability, parallelism, the ability to run on commodity hardware and fault-tolerance. These systems assemble many nodes across a network and provide a distributed file system with large storage capacity, where data can be transparently accessed (Roch et al., 2018). The ability to use commodity hardware makes scalability economically viable, allowing the addition of more devices to scale up the system in an incremental fashion. Because the file system runs on several commodity hardware components, which are highly prone to failure, techniques such as replication and erasure codes are used to avoid data loss and increase fault-tolerance in case of hardware failures.

Traditionally, computing and storage resources in datacenters are separated into different pods. The computing pod usually uses virtualization technologies to optimize resources, while the storage pod is usually based on monolithic hardware. Recently, hyperconvergence is an emerging paradigm that has been gaining popularity, both in academia and industry (Verma et al., 2017). This paradigm allows to merge computing and storage components with hypervisors and DFS solutions. While traditional storage systems use dedicated hardware to storage and compute, a hyperconverged system uses a DFS and a hypervisor to aggregate the storage capacity and computing resources, providing an unique pod of compute and storage.

In a cloud computing environment, containers can benefit from a hyperconverged infrastructure. The traditional container deployment uses local storage on the node to start and run. Then, the ability to store container's persistent data on a hyperconverged system may improve scalability, elasticity and fault-tolerance in these environments.

Considering the above, the motivation of this work is to evaluate storage performance of a proposed hyperconverged system, based on GlusterFS as the DFS and VMware ESXi as the hypervisor. The goal is to evaluate the performance and potential of container's storage I/O on an hyperconverged system, by running inside Docker containers two applications highly dependent on storage capabilities, the NoSQL databases Cassandra and MongoDB (Abramova and Bernardino, 2013), under several workloads of the Yahoo Cloud Service Benchmark (YCSB) (Cooper

et al., 2010). The results of the proposed hyperconverged system are compared with a traditional configuration using local storage, with the intention of verifying the viability of its use in place of direct-attached storage.

This paper is organized as follows: Section 2 presents related works and the literature review; Section 3 presents and describes the proposed architecture; Section 4 details the experiments and results and finally, Section 5 presents the conclusions and future work of this research.

2 THEORETICAL CONCEPTS AND RELATED WORK

Permanent storage consists of a named set of objects that are explicit created, are immune to temporary failures of the system, and persist until explicitly destroyed. A file system is a refinement that describes a naming structure, a set of operations on objects described by explicit characteristics. DFS allow multiple users who are physically dispersed in a network of autonomous computers share in the use of a common file system. Cloud storage is a system that provides data storage and assembles a large number of different types of storage devices through the application software which are based on the functions of cluster applications, grid techniques and DFS. Cloud storage is a cloud computing system with large capacity storage. An hyperconverged environment combines compute, storage and networking components into a single system or node based on commodity servers, with the aim to reduce hardware costs when compared to specialized storage arrays, servers and other equipment, which appears today as an interesting option to cloud providers.

In the DFS area, recent studies evaluated the common problem of how to store and handle large amounts of data. For example, in (Roch et al., 2018) there is a comparison between DFS for storing molecular calculations of quantum chemistry. In this study the DFS GlusterFS and Ceph were tested, using 8 nodes distributed in 8 different combinations. The experiments were performed with 7 different patterns of reading and writing. The conclusion was that GlusterFS outperforms Ceph during large I/O workloads and that GlusterFS administration is simplified, requiring less maintenance and installation time. The conclusions about Ceph were that it has greater complexity and needs separate servers for metadata (data about other data), which increases the complexity of the solution.

Another DFS study in a similar area investigates the problem of how to store and transfer large amounts of genetic data between two institutions (Mills et al., 2018). This study shows that the best way to transfer data between two research institutions is to use a combination of a high-speed communications network and a high-performance file system. Finding a balance between these two items enables data to be read from the source at high speed, transmitted quickly, and then recorded at high speed at the destination. The DFS analyzed in this study were BeeGFS, Ceph, GlusterFS and OrangeFS. Experiments were performed with 4 to 8 servers at the source transmitting data via FTP to another 4 to 8 servers at the destination, with BeeFS getting the best performance.

In the work of (Kaneko et al., 2016) a guideline for data allocation in DFS is proposed, recommending that data stored in multiple nodes with few disks have a better performance than a few nodes with many disks. In the experiments, volumes built with disks on different servers were compared with volumes built with several disks on the same server. The comparison was made analyzing the read and write throughput while copying files to and from the volumes. In the results it was verified that the proposed guideline obtained better performance in larger multiplicities. The multiplicity is defined by the author as being the number of clients accessing simultaneously a number of volumes. For example, multiplicity 24 means 4 clients simultaneously accessing 6 volumes. The analysis concluded that distributing the volume across several servers results in improved data throughput because of the parallelism in data access. This guideline can be considered a good practice for building DFS. Building volumes with few server the advantages of a parallel file system are lost and server resources (processor, disk controller, network card, etc.) can impact the performance. Spreading a volume across multiple servers allowed them to be accessed by different servers, thinning resource consumption between each server and improving data throughput.

Studies on performance of hyperconvergence environments are still scarce. The fundamental concept behind this architecture is based on the Beowulf Systems, that already been vastly studied by academia. In a Beowulf system a cluster of commodity-grade computers are networked to run some software that can perform parallel computing and storage. A hyperconverged system is a recent evolution of this concept, mashing virtualized servers and virtualized storage onto the same clusters with the goal of simplifying the infrastructure.

In one of the first studies specifically about hyperconvergence, the General Parallel File System

(GPFS) is presented as a software defined storage solution for hyperconverged systems (Azagury et al., 2014). In this work, several concepts of hyperconvergence and software defined storage are outlined, and the proposed solution is to use GPFS within a hyperconverged architecture using virtual storage appliances (VSA) that manage physical storage resources. The VSA allow the increase of capacity in a scale-out model and data protection through the "GPFS native RAID" (GNR) that makes copies of the data in several nodes.

Recent studies on containers storage performance analyze the possible configurations for storage access (Tarasov et al., 2017) (Xu et al., 2017a), how to provide encryption in image manipulation (Giannakopoulos et al., 2017) and performance using solid state disks (SSD) (Xu et al., 2017b) (Bhimani et al., 2016). In (Tarasov et al., 2017) and (Xu et al., 2017a), the mechanisms for containers access storage areas are described and analyzed, but both are limited to local storage, as well as in (Giannakopoulos et al., 2017) (Xu et al., 2017a). In (Xu et al., 2017b) the authors analyze the use of remote storage for containers, using non-volatile memory express over fabrics (NVMF) technology being accessed by a remote direct access memory (RDMA) over a 40 Gbps ethernet network. This study is closer to our proposal, since the container's data will be stored in a DFS.

The amount of data generated by IoT devices is addressed in another study, in witch the storage paradigms for store and process data from IoT devices are analyzed. The conclusion points to the hyperconverged architecture as one solution because of its scalability, fault-tolerance and use of COTS hardware (Verma et al., 2017).

3 PROPOSED ARCHITECTURE

The proposed architecture is presented in Figure 1 and works as follows:

- The server runs a Type I Hypervisor with at least two disk controllers: one for hypervisor, VSA and container host ❶; and one for the hyperconverged system ❷.
- On the disks connected to the disk controller managed by the hypervisor on path ❸ are stored the hypervisor itself and both VSA and the Container Host virtual machines, as can be seen on path ❺.
- Using the "PCI Passthrough" feature on the hypervisor, the second disk controller ❷ is directly managed by the VSA virtual machine via path ❹.

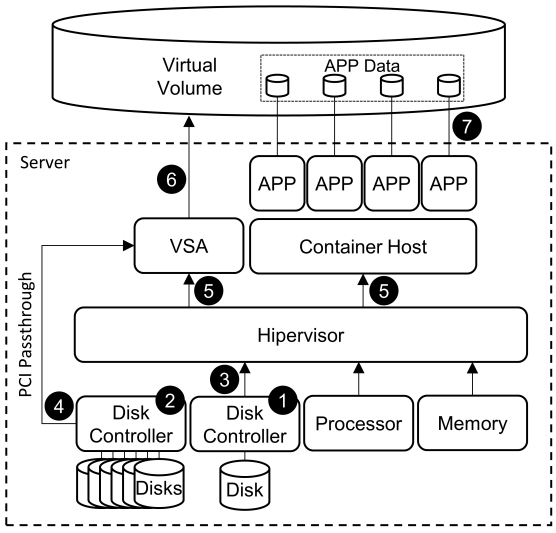

Figure 1: Hyperconverged Server Architecture.

- Each VSA, with its own storage resources, is configured to participate in an existing cluster or start a new one. The nodes provide their individual resources ❻ to create a distributed virtual volume across the nodes of the cluster, as presented on Figure 2. Data replication between VSA nodes is performed over a low-latency, high-bandwidth Ethernet network.

- After the virtual volume is created, it can be accessed by the container engine as a scalable and fault-tolerant data storage unit, as can be seen on path ❼.

The use of a disk controller dedicated to the VSA is a desirable characteristic because it allows direct control of the disks, dropping an abstraction layer in the hypervisor. Another desirable requirement is a high bandwidth and low latency network between the servers to avoid congestion or any kind of degradation during storage operations. The use of disks with same model to build a volume is another important aspect to get a better performance (Kaneko et al., 2016).

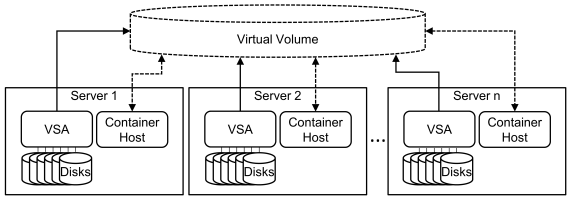

Figure 2: Virtual Volume build by the VSAs.

3.1 Implementation

In the next subsections, we describe the tools that were integrated to implement the proposed arquitecture.

3.1.1 GlusterFS

GlusterFS is an open-source DFS that allows to distribute a large set of data across multiple servers. It has simple operation, being supported by most Linux distributions. It does not separate metadata/data and therefore does not need additional servers for this task (Fattahi and Azmi, 2017). It has a native mechanism of redundancy based on the replication of files between a set of "bricks" (a basic unit of storage) that form a resilient logical unit (Roch et al., 2018). It allows the use of heterogeneous nodes, enabling nodes with different configurations on the same distributed volume. In these cases, it is recommended to group nodes with the same characteristics for better performance (Kaneko et al., 2016).

Bricks are the basic storage unit in GlusterFS, represented by an export directory on a server. A collection of blocks forms a "Volume" in which GlusterFS operations take place. GlusterFS allows you to group Bricks to form five different types of volumes:

- Distributed Volume: Distribute files between bricks. The final volume capacity is the sum of bricks' capacity. It has no redundancy. Failure on a brick corrupts the volume. Recommended for cases where availability is not important or is provided by another mechanism.
- Replicated Volume: The files are replicated between the volume bricks. Recommended for cases where high availability and reliability are required.
- Distributed Replicated Volume: Distributes files between replicated subvolumes. Recommended for cases where scalability, high availability and good reading performance are required.
- Dispersed Volume: They are based on erasure codes and provide efficient protection against disk or server failures. A coded fragment of the original file is stored in each brick, so that from a set of fragments it is possible to retrieve the original file.
- Distributed Dispersed Volume: Distributes files between dispersed subvolumes. It has the advantages of a distributed volume, but using dispersed storage within the subvolume.

3.1.2 ESXi

VMware ESXi is a hypervisor that runs directly on the host hardware, i.e., it is a virtual machine monitor (VMM) type I (bare metal). It takes minimal memory from the physical server and provides a robust, high-performance virtualization layer that abstracts the server's hardware resources and enables the sharing of these resources across multiple virtual servers.

3.1.3 Docker

Docker is a application that performs operating-system-level virtualization also known as containerization. It uses the resource isolation features of kernel to allow independent containers to run within a single host, avoiding the overhead of starting and maintaining virtual machines (VMs) (Xu et al., 2017a). Containers and virtual machines have similar resource isolation and allocation benefits but different architectural approaches, which allows containers to be more portable and efficient compared to bare metal and virtual machines (Bhimani et al., 2016).

3.1.4 Virtual Storage Appliance

The core component of our proposed architecture is the Virtual Storage Appliance (VSA), a virtual machine that runs a minimal version of the Linux operating system and the GlusterFS software. GlusterFS is configured to manage the physical storage resources presented to the VSA and use these to build a DFS. The choice of GlusterFS for the DFS is due to the fact that data and metadata are stored together in the nodes, without a specific node to handle metadata (Fattahi and Azmi, 2017). This feature makes GlusterFS ideal for use in a hyperconverged architecture since the nodes are independent and can run without external dependencies, simplifyng the architecture and allowing the deployment of just one VSA per hypervisor and nothing else. The VSA nodes, each with its own storage resources, allow the creation of DFS volumes that span across many server and can be used by hypervisor and container engine to store data with scalability and fault-tolerance.

4 EXPERIMENTAL ANALYSIS

4.1 Testbed

In our experiments we built a hyperconverged system using three IBM HS23 servers with dual Intel Xeon CPU E5-2670 2.60GHz CPUs, 96GB of RAM,

one 16GB SSD (to host ESXi, VSA and Docker Host), two 1TB HDD (to the hyperconverged system), hypervisor ESXi version 6.5 and network connection via two gigabit Ethernet ports. The VSA on each hypervisor has 4 vCPU, 8 GB of RAM, runs CentOS Linux release 7.5.1804, GlusterFS version 3.12.15 and XFS as the file system for disks managed by Gluster. The Container Host has 12 vCPUs, 80 GB of RAM, runs CentOS Linux release 7.5.1804 and Docker 18.09. The connection between the Docker Host and the virtual volume built by the VSAs is made through Gluster Native Client. To evaluate the storage I/O performance of the hyperconverged system we choose the NoSQL databases Cassandra (version 3.11.3) and MongoDB (version 4.0.3) running in Docker containers and storing persistent data on the hyperconverged virtual volume.

To compare results from the hyperconverged system we built a traditional system where storage resources are provided by local disks on the server. For this system we used another IBM HS23 server with same configuration as the hyperconverged servers, but the persistent storage for the containized NoSQL databases were provided by the hypervisor using the local server's disk.

Since the objective is to analyze storage performance, the memory cache feature on both NoSQL databases were disabled, meaning that all database operations had to hit the storage volume and consequently the underlying disks.

4.2 Workload

To evaluate our proposal we used the Yahoo Cloud Serving Benchmark (YCSB) framework (Cooper et al., 2010), a tool built do load, run and measure performance of different NoSQL databases. After loading data on the desired database, there are six core workloads that can be run to evaluate database's performance. These core workloads are named A, B, C, D, E and F, each with different I/O patterns. Besides those, YCSB also allows using custom workloads, which we used to test three more workloads: two from Microsoft's datacenters (Huang et al., 2017) and one developed by us, called "Heavy Update", with a heavy write load. The 9 different workloads used to evaluate our propose is summarized in Table 1.

4.3 Metrics, Validation and Factors

The metric evaluated during the execution of the experiments is the operations per second (ops/sec), i.e., how many read, update or insert operations can be submitted to the NoSQL database every second.

Table 1: Application workloads used for evaluation.

Workload	I/O Pattern
YCSB-A	50% read, 50% update
YCSB-B	95% read, 5% update
YCSB-C	100% read
YCSB-D	95% read, 5% insert
YCSB-E	95% scan, 5% insert
YCSB-F	50% read, 50% read-modify-write
Microsoft Cloud Storage	26.2% read, 73.8% update
Microsoft Web Search	83% read, 17% update
Heavy Update	100% update

The validation of the results is performed evaluating the results from the proposed hyperconverged system in comparison with the results from a traditional system where data is stored on the local server's disks. For our experiments, the factors we chose to analyze were: type of GlusterFS volumes; type of NoSQL database; and number of threads.

4.4 Evaluation Scenarios

In the hyperconverged system built for this study, GlusterFS can be configured to create five types of volumes: Distributed, Replicated, Distributed Replicated, Dispersed and Distributed Dispersed. Since the Distributed volume does not offer fault-tolerance and the Replicated volume does not scale, we did not used these two types of disk configurations. The other three types of volume offer equal fault-tolerance protection, but different types of performance. So we tested all those three types of volumes with all workloads.

To evaluate the hyperconverged storage I/O performance we used the NoSQL databases Cassandra and MongoDB, each one submitted to the same workloads and storing persistent data on the same types of volumes. The choice for these two is due to the fact that databases are applications that depend on the storage performance where data is stored and retrieved, with Cassantra and MongoDB being popular among recent studies in both academia and industry.

Both NoSQL databases were loaded with 1.000.000 records of 1KB using the YSCB tool. Then the databases were submitted to the workloads with different threads numbers (ranging from 1 to 100, in steps of 10). In each thread test, 100.000 transactions were made. The mean resulted of the amount of transactions made in each second were calculated with a 95% confidence interval.

4.5 Experiments

- **Experiment 1 - YCSB core workloads**: This experiment was conducted using the six core workloads available in the YCSB framework against

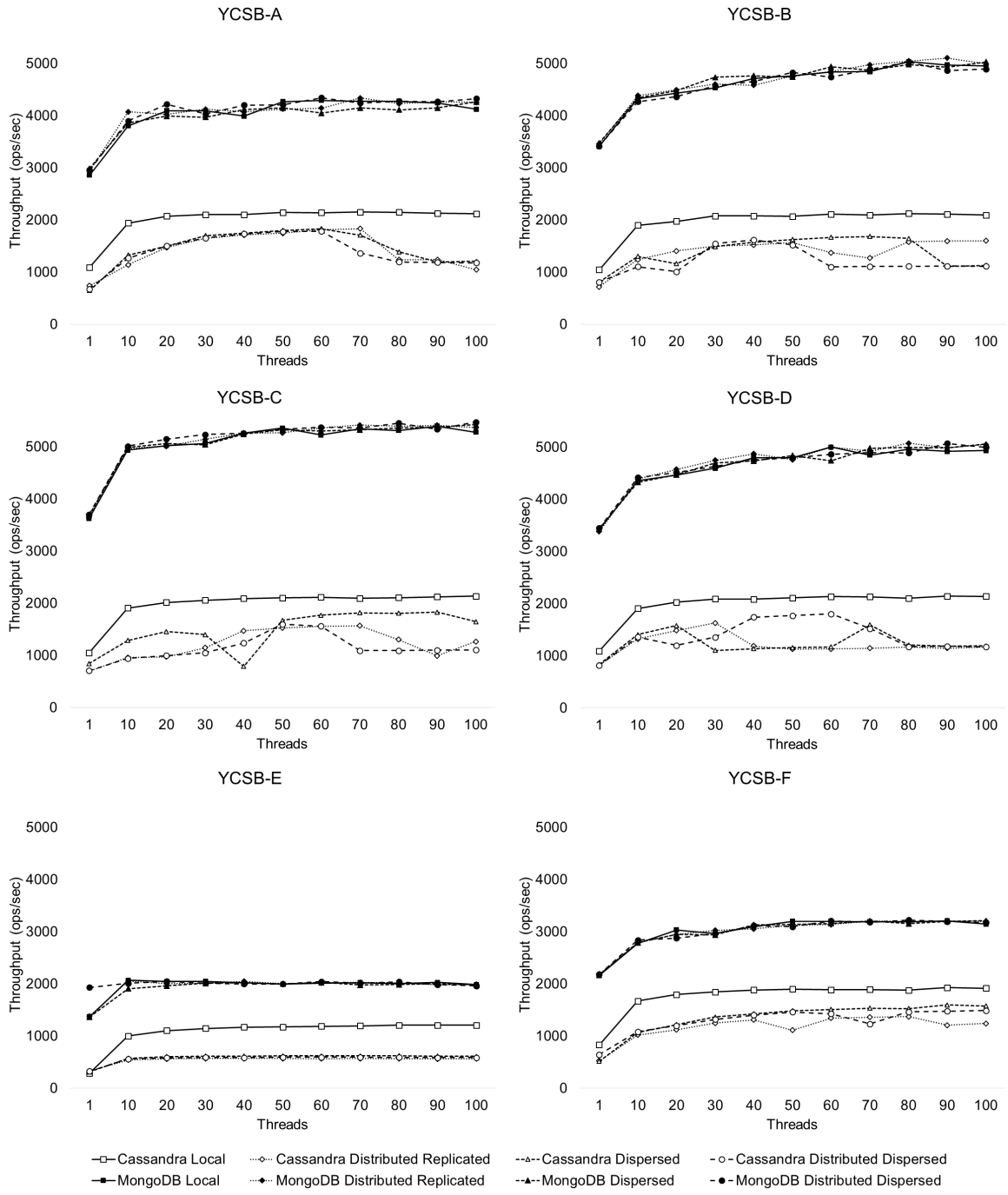

Figure 3: YCSB Workloads Results.

Cassandra e MongoDB containerized instances. The results can be seen on Figure 3.

- **Experiment 2 - Microsoft workloads**: This experiment was conducted using two Microsoft's datacenters workloads. The results can be seen on Figure 4.
- **Experiment 3 - Heavy Update workload**: To analyze the performance of write I/O storage operation, we this experiment with only updates on the database. The results can be seen on Figure 5.

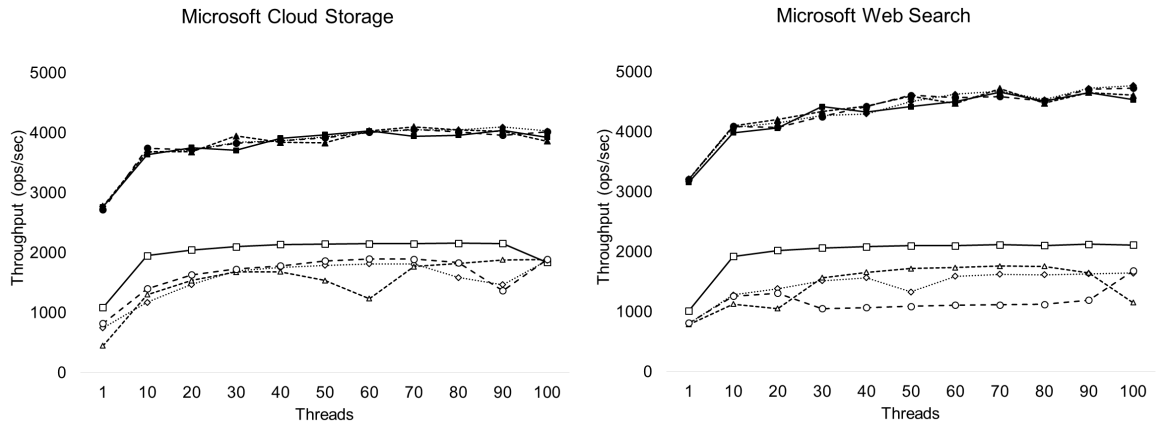

Figure 4: Microsoft Workloads Results.

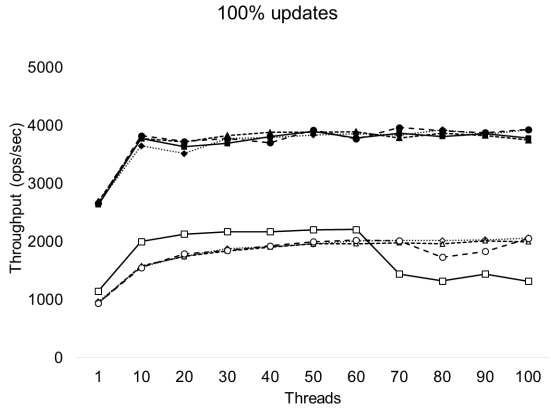

Figure 5: Heavy Update Workloads Results.

4.6 Analysis of Results

The experimental results shown on Figures 3, 4 and 5 indicates that MongoDB performed better than Cassandra in all experiments. MongoDB running on the hyperconverged system presented a performance equivalent to that of a traditional system, independent of the number of threads. In some specific cases we archived a 7% better performance on the hyperconverged system than on local storage. We consider this a good result because, even with the abstractions in storage layer imposed by the hyperconverged system, the performance was similar to the traditional system where data is saved in a direct-attached storage.

Cassandra performance on the hyperconverged system was below the expectations on almost all workloads. The only workload where our proposal performed better than direct-attached storage was in Heavy Update workload above 70 concurrent threads, where the hyperconverged system had 56% better performance than on local storage.

The difference in performance results between Cassandra and MongoDB can be explained by the way these NoSQL databases store their data in the file system. MongoDB stores data in large files, while Cassandra does it in several small files. Storage performance differences between using large and small files is already known in the literature due to overhead imposed by the DFS (Roch et al., 2018).

We also note that the performance of Cassandra is changed by the disk configuration used in GlusterFS. Significant differences in throughput value for the same workload and number of threads can been observed in the results obtained with the YCSB-B, C and D workloads on Figure 3. This behavior indicates that workloads operating with small files in GlusterFS have different performance depending on the volume disk configuration used.

It should be mentioned that during the experiments no network congestion, high CPU usage or high memory usage were observed, meaning that these parameters did not interfered on the experiment's results.

5 CONCLUSIONS AND FUTURE WORK

This paper presented a storage performance analysis on a hyperconverged infrastructure that uses an open-source DFS to store and replicate data between multiple servers while using the computing resources of the same servers to host containers and store its persistent data. The proposal was evaluated under several different scenarios, using 2 NoSQL databases, 9 workloads, 3 disk configurations and 11 different number of concurrent sessions (threads), compared to a traditional direct-attached storage infrastructure.

The results show that applications working with large files in the hyperconverged system performs similarly to traditional local storage. But a hyperconverged system adds desired features for cloud computing systems when compared to conventional storage, like scalability, elasticity and fault-tolerance. A hyperconverged system can grow easily by adding more servers, which may have different computing and storage configurations. Meanwhile in a traditional architecture, new stand-alone servers can be added to the infrastructure, but computing and storage resources are confined to each server. Besides that, a small hyperconverged system can easily bypass the resources of a traditional local storage-based system.

We believe that the use of our proposal of hyperconverged infrastructure is feasible for cloud service providers in the provision of IaaS services (e.g. vitual machines and containers) and PaaS services (e.g. containerised databases). An user's application that requires more storage resources than a single server can provide, has challenges for scaling in a non-convergent system but would have the resources allocated easily in a hyperconverged architecture. This type of feature is desired by datacenters from cloud providers because it optimizes infrastructure by allowing two resource pods (computing and storage) to collapse into one, increasing efficiency.

The network requirements for the servers of a hyperconverged infrastructure must be properly sized to minimize the possibility of congestion. The throughput between the server and its disks should occur without limitations imposed by throughput of the network. As the scale of the hyperconverged system built for the experiments was small, the network infrastructure did not interfere with the results. However if the experiment's testbed scale were larger, then network resources would need to be increased.

In further research, we intend to evaluate our proposal with bigger clusters, other hypervisor solutions, different type of disks (e.g., SSD), server configurations and workloads.

REFERENCES

Abramova, V. and Bernardino, J. (2013). Nosql databases: Mongodb vs cassandra. In *Proceedings of the international C* conference on computer science and software engineering*, pages 14–22. ACM.

Azagury, A. C., Haas, R., Hildebrand, D., Hunter, S. W., Neville, T., Oehme, S., and Shaikh, A. (2014). Gpfs-based implementation of a hyperconverged system for software defined infrastructure. *IBM Journal of Research and Development*, 58(2/3):6–1.

Bhimani, J., Yang, J., Yang, Z., Mi, N., Xu, Q., Awasthi, M., Pandurangan, R., and Balakrishnan, V. (2016). Understanding performance of i/o intensive containerized applications for nvme ssds. In *Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International*, pages 1–8. IEEE.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154. ACM.

Fattahi, T. and Azmi, R. (2017). A new approach for directory management in glusterfs. In *Information and Knowledge Technology (IKT), 2017 9th International Conference on*, pages 166–174. IEEE.

Giannakopoulos, I., Papazafeiropoulos, K., Doka, K., and Koziris, N. (2017). Isolation in docker through layer encryption. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 2529–2532. IEEE.

Huang, J., Badam, A., Caulfield, L., Nath, S., Sengupta, S., Sharma, B., and Qureshi, M. K. (2017). Flashblox: Achieving both performance isolation and uniform lifetime for virtualized ssds. In *FAST*, pages 375–390.

Kaneko, S., Nakamura, T., Kamei, H., and Muraoka, H. (2016). A guideline for data placement in heterogeneous distributed storage systems. In *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*, pages 942–945. IEEE.

Mills, N., Feltus, F. A., and Ligon III, W. B. (2018). Maximizing the performance of scientific data transfer by optimizing the interface between parallel file systems and advanced research networks. *Future Generation Computer Systems*, 79:190–198.

Roch, L. M., Aleksiev, T., Murri, R., and Baldridge, K. K. (2018). Performance analysis of open-source distributed file systems for practical large-scale molecular ab initio, density functional theory, and gw+ bse calculations. *International Journal of Quantum Chemistry*, 118(1).

Tarasov, V., Rupprecht, L., Skourtis, D., Warke, A., Hildebrand, D., Mohamed, M., Mandagere, N., Li, W., Rangaswami, R., and Zhao, M. (2017). In search of the ideal storage configuration for docker containers. In *Foundations and Applications of Self* Systems (FAS* W), 2017 IEEE 2nd International Workshops on*, pages 199–206. IEEE.

Verma, S., Kawamoto, Y., Fadlullah, Z. M., Nishiyama, H., and Kato, N. (2017). A survey on network methodologies for real-time analytics of massive iot data and open research issues. *IEEE Communications Surveys & Tutorials*, 19(3):1457–1477.

Xu, Q., Awasthi, M., Malladi, K. T., Bhimani, J., Yang, J., and Annavaram, M. (2017a). Docker characterization on high performance ssds. In *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*, pages 133–134. IEEE.

Xu, Q., Awasthi, M., Malladi, K. T., Bhimani, J., Yang, J., and Annavaram, M. (2017b). Performance analysis of containerized applications on local and remote storage. In *Proc. of MSST*.

ANEXO 2

Artigo aceito no CLEI 2019

(XLV Latin American Computing Conference)

Cidade do Panamá / Panamá

Análise de Desempenho do Armazenamento de Dados em uma Infraestrutura Hiperconvergente usando Docker e GlusterFS

Rodrigo Leite
Departamento de Ciência da Computação
Universidade de Brasília (UnB)
Brasília, Brasil
rodrigo123@gmail.com

Priscila Solis
Departamento de Ciência da Computação
Universidade de Brasília (UnB)
Brasília, Brasil
pris@unb.br

***Resumo*—As infraestruturas hiperconvergentes usam sistemas de arquivos distribuídos para armazenar e replicar dados entre múltiplos servidores enquanto usam os recursos de processamento desses mesmos servidores para hospedar máquinas virtuais e *containers*. Neste trabalho, o sistema de arquivos distribuído GlusterFS e o hipervisor VMware ESXi são utilizados para construir um sistema hiperconvergente para hospedar *containers* Docker com o objetivo de avaliar o desempenho desse sistema em comparação com uma abordagem tradicional em que os dados são armazenados diretamente nos discos locais dos servidores. O desempenho do armazenamento persistente de dados dos *containers* é avaliado usando a ferramenta de *benchmark* de armazenamento FIO (Flexible I/O tester) com diferentes cargas de trabalho oriundas dos *datacenters* da Microsoft e com múltiplas configurações de discos no sistema hiperconvergente. Os resultados experimentais mostram que com uma maior quantidade de operações de escrita e que manipulam grandes arquivos, o desempenho na arquitetura hiperconvergente é melhor que no armazenamento local, o que apresenta uma opção economicamente viável na implementação de armazenamento para *datacenters*.**

***Index Terms*—Hyperconverged Infrastructures, GlusterFS, Containers, DFS, FIO, Docker, Cloud, Storage**

I. Introdução

Nos últimos anos, a grande quantidade de dados gerados por múltiplos dispositivos, usuários e suas aplicações têm crescido de forma exponencial e se tornando um desafio às soluções tradicionais de armazenamento. Por esse motivo nos últimos anos vários Sistemas de Arquivos Distribuídos (SAD) foram desenvolvidos para suportar grandes demandas por armazenamento, agregando características interessantes tais como escalabilidade, tolerância a falhas, paralelismo e capacidade de executar em plataformas de hardware tradicional. Esses sistemas reúnem os recursos de diversos nós conectados em uma rede para fornecer um SAD com grande capacidade de armazenamento, em que os dados podem ser acessados de forma transparente [1] [2]. A capacidade de usar hardware comum de mercado torna a escalabilidade economicamente viável, permitindo a adição de mais nós para aumentar o sistema de maneira incremental [3]. Como o sistema de arquivos é executado em componentes de hardware comuns que são altamente propensos a falhas, técnicas como replicação e *erasure codes* são usadas para evitar a perda de dados e aumentar a tolerância a falhas em caso de falhas de hardware [4].

Tradicionalmente os recursos de computação e armazenamento em *datacenters* são separados em diferentes infraestruturas. A infraestrutura de computação geralmente usa tecnologias de virtualização para otimizar recursos, enquanto que a infraestrutura de armazenamento é geralmente baseada em hardware monolítico. Recentemente, a hiperconvergência surgiu como novo paradigma que vem ganhando popularidade, tanto na academia quanto na indústria [5]. Esse paradigma permite unir os componentes de computação e armazenamento, usando um hipervisor e SAD. Enquanto os sistemas tradicionais de armazenamento usam hardware dedicado para armazenamento e computação, um sistema hiperconvergente usa um SAD e um hipervisor para agregar os recursos de armazenamento e computação, fornecendo uma infraestrutura unificada.

Em um ambiente de computação em nuvem, os *containers* podem se beneficiar de uma infraestrutura hiperconvergente. Uma implantação tradicional de *containers* costuma usar o armazenamento local do servidor para iniciar uma imagem e executar o *container*. A possibilidade de armazenar dados persistentes de *containers* em um sistema hiperconvergente pode permitir melhorar a escalabilidade, elasticidade e tolerância a falhas nesses ambientes.

Considerando o exposto, a motivação deste trabalho é propor e avaliar o desempenho do armazenamento de dados em um sistema hiperconvergente baseando no GlusterFS como SAD, VMware ESXi como Hipervisor e Docker como *Container Engine*. O objetivo é avaliar o desempenho e potencial do armazenamento de dados de *containers* em uma infraestrutura hiperconvergente. A avaliação utiliza a execução dentro de *containers* da ferramenta de benchmark de armazenamento FIO sob duas cargas de trabalho dos *datacenters* da Microsoft. Os resultados do sistema hiperconvergente proposto são comparados com uma configuração tradicional usando armazenamento local, com a intenção de verificar a viabilidade

de seu uso no lugar do armazenamento direto nos discos (*Direct-Attached Storage*).

O restante desse trabalho está estruturado da seguinte forma: a Seção 2 apresenta a revisão da literatura e trabalhos relacionados; a Seção 3 apresenta e descreve a arquitetura proposta; a Seção 4 detalha os experimentos e resultados; a Seção 5 discorre sobre a viabilidade de uso da hiperconvergência em ambientes de nuvem; por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

II. Conceitos Teóricos e Trabalhos Relacionados

O armazenamento permanente de dados consiste em um conjunto nomeado de objetos que são explicitamente criados, são imunes a falhas temporárias do sistema e persistem até serem explicitamente destruídos. Um sistema de arquivos é um refinamento que descreve uma estrutura de nomes e um conjunto de operações de objetos descritos por características explícitas. Conceitualmente, um SAD permite que vários usuários que estão fisicamente dispersos em uma rede de computadores autônomos compartilhem o uso de um sistema de arquivos comum de forma transparente.

O armazenamento de dados em nuvem é um sistema que fornece grande capacidade e agrega um grande número de diferentes dispositivos de armazenamento por meio de software, que são baseados nas funções de *cluster* e técnicas de *grid*. Conceitualmente, a nuvem usa um SAD para armazenar dados com o uso de recursos de hardware tais como *storage arrays*. Um sistema hiperconvergente combina componentes de computação, armazenamento e rede em um único sistema baseado em servidores comuns, com o objetivo de reduzir custos de hardware quando comparado a *storage arrays* e outros equipamentos que aparecem hoje como opções para provedores de nuvem .

Estudos recentes avaliaram o problema de como armazenar e tratar grandes quantidades de dados. Por exemplo no trabalho realizado em [2] há uma comparação entre os sistemas de arquivos distribuídos para armazenar cálculos moleculares de química quântica. Nesse estudo foram testados sistemas de arquivos distribuidos tais como GlusterFS e Ceph, utilizado 8 nós distribuídos em 8 combinações distintas e a análise de desempenho realizada pela ferramenta FIO, um gerador de I/O. Foram realizados testes com 7 padrões distintos de gravação, leitura e tamanho. A conclusão do estudo foi que o GlusterFS superou o Ceph durante cargas de trabalho com grande quantidade de I/O e que a administração do GlusterFS é simplificada, exigindo menos tempo de manutenção e instalação. As conclusões sobre o Ceph foram que este possui complexidade maior e precisa de servidores separados para metadados, o que aumenta a complexidade da solução.

Outro estudo analisou o problema de como armazenar e transferir grandes quantidades de dados genéticos [6] entre duas instituições. Nesse estudo verificou-se que a melhor maneira de transferir dados é usando uma combinação entre uma rede de comunicação de alta velocidade e um sistema de arquivos de alto desempenho. Ao encontrar um ponto de equilíbrio entre esses dois itens possibilita-se que os dados consigam ser lidos na origem em alta velocidade, transmitidos rapidamente e por fim gravados em alta velocidade no destino. Os sistemas de arquivos distribuídos analisado nesse estudo foram BeeGFS, Ceph, GlusterFS e OrangeFS. Foram realizados testes com 4 a 8 servidores na origem transmitindo dados via FTP para outros 4 a 8 servidores no destino. Por fim o BeeFS obteve o melhor desempenho nos testes.

No trabalho de [7] é apresentada uma diretriz para alocação de dados em sistemas de arquivos distribuídos, de forma que os dados sejam armazenados em vários nós com poucos discos em vez de poucos nós com vários discos, aumentando o paralelismo no acesso aos dados. O trabalho propõe criar volumes compostos de discos de diferentes servidores e compara com volumes criados com diversos discos no mesmo servidor. A comparação foi realizada usando os valores de vazão de dados de escrita e leitura durante a cópia de arquivos partir de diferentes clientes. Os dados foram adquiridos por meio da ferramenta Sysstat. Analisando os resultados verificou-se que a diretriz proposta obteve melhores resultados em multiplicidades maiores. A multiplicidade nesse caso é a quantidade de clientes acessando simultaneamente uma quantidade de volumes. Por exemplo, multiplicidade 24 significa 4 clientes acessando simultaneamente 6 volumes. Ao distribuir o volume em discos de servidores diferentes, o paralelismo no acesso aos dados melhorou a vazão dos dados. A diretriz proposta no artigo obteve bons resultados e pode ser considerada um boa prática para construção de sistemas de arquivos distribuídos. Ao concentrar todos os discos de um volume em um servidor, as vantagens de um sistema de arquivos paralelo são perdidas e o desempenho é menor devido a limitação de recursos do servidor (processador, controladora de discos, placa de rede, etc.). Espalhar os discos do volume em diversos servidores permitiu que eles fossem acessados por servidores diferentes, diluindo o consumo de recursos entre cada servidor e melhorando a vazão de dados.

Estudos sobre desempenho de ambientes de hiperconvergência ainda são escassos. O conceito fundamental por trás dessa arquitetura é baseado nos sistemas Beowulf [8]. Em um sistema Beowulf, um grupo de computadores comuns são ligados em rede para executar softwares de computação e armazenamento paralelos. Um sistema hiperconvergente é uma evolução recente desse conceito, mesclando servidores virtualizados e armazenamento virtualizado nos mesmos *clusters* com o objetivo de simplificar a infraestrutura.

Em um dos primeiros estudos sobre hiperconvergência, o GPFS (*General Parallel File System*) é apresentando como armazenamento definido por software em sistemas hiperconvergentes. Nesse trabalho muitos conceitos de hiperconvergência e armazenamento definido por software são discutidos. É apresentada uma proposta que usa GPFS como elemento de SAD de um sistema hiperconvergente usando VSAs (*Virtual Storage Appliances*) para gerenciar os recursos de armazenamento. O VSA permite o aumento de capacidade em um modelo de *scale-out* e proteção de dados através do GNR (*GPFS native RAID*) que faz cópias dos dados em vários nós.

Estudos recentes na área de *containers* que abordam a

questão do armazenamento de dados focam principalmente na análise das diferentes configurações possíveis para o armazenamento de imagens e *containers*. No estudo realizado em [9] analisou-se o desempenho dos diferentes *Storage drivers* usados na infraestrutura de *containers* provida pela ferramenta Docker. Os *Storage drivers* são sistemas de arquivos montados em cima do sistema de arquivos do *host* que visam o armazenamento de imagens e *containers*, principalmente na iteração entre as camadas de uma imagem [10]. Nesse estudo uma ferramenta de *benchmark* de sistemas de arquivos foi instalada dentro de um *container* que executou uma carga de trabalho sintética de leitura e escrita em cada um dos *Storage Drivers*. A conclusão foi de que não existe um *Storage Driver* com bom desempenho em todos os cenários analisados, sendo que este deve ser escolhido de acordo com o padrão de leitura e escrita da aplicação que será empacotada dentro do *container*.

No trabalho [11] é abordada a questão da análise do desempenho de diferentes *Storages Drivers*, porém com o foco no uso de unidade de armazenamento de estado sólido (SSD) de alto desempenho instalados no *host*. Nesse estudo foram analisados o armazenamento de imagens e *container* e o armazenamento persistente. Em [9] foi estudada uma ferramenta de *benchmark* de sistemas de arquivos instalada dentro de um *container* que executou uma carga de trabalho sintética de leitura e escrita em cada um dos *Storage Drivers*. Adicionalmente uma unidade de armazenamento persistente foi apresentada para o *container* do experimento e a mesma ferramenta de *benchmark* executou a carga de trabalho no armazenamento persistente. O experimento foi repetido com diferentes *Storages Drivers*, mantendo-se o mesmo armazenamento persistente. A conclusão foi de que os drivers do tipo "overlay" e "aufs" no geral apresentaram melhor desempenho ao usar um SSD e que as operações no armazenamento persistente não são afetadas pelo *Storage Driver*.

Em outro trabalho envolvendo o armazenamento de dados em *containers* [12] também é analisado o desempenho usando unidades de armazenamento de estado sólido (SSD) de alto desempenho, tanto localmente quanto remotamente. Diferente dos trabalhos de [9] e de [11], nesse estudo temos um experimento onde uma unidade externa ao *host* onde o *container* roda é utilizada como armazenamento, sendo acessada via uma rede de alta velocidade. Nessa configuração foi utilizado um protocolo chamado NVMF (*Non-Volatile Memory express over Fabrics*) que é utilizado para estender o barramento PCI-e do *host* através de uma rede *Ethernet*. Na análise experimental diversos *containers* com o banco de dados Cassandra foram executados e uma carga de trabalho sintética foi aplicada aos *containers*. O local de armazenamento de dados do Cassandra foi configurado de diferentes maneiras de modo a avaliar o desempenho tanto local quanto remoto. A conclusão foi de que usando essa tecnologia tanto o armazenamento local quanto remoto obtiveram desempenho próximos.

No estudo realizado em [13] é apresentada uma implantação de armazenamento definido por software em um ambiente heterogêneo. Como prova de conceito, os sistemas de arquivos distribuídos HDFS, Ceph e GlusterFS foram integrados em um sistema de gerenciamento de serviços em nuvem rodando OpenStack. Nesse estudo o software EMC ViPR foi utilizado para gerenciar um conjunto físico de recursos de armazenamento. Esses recursos foram então disponibilizados para o OpenStack visando a criação de servidores virtuais. Em outro trabalho semelhante [14], o armazenamento definido por software foi utilizado em uma arquitetura com sistemas de arquivos distribuídos heterogêneos, onde o uso de um plano de controle em software permitiu a implantação de um mecanismo de partição e distribuição de arquivos entre os diferentes tipos de sistemas de aquivos.

Apesar da queda recente do uso de servidores virtuais nos ambientes de computação em nuvem, a virtualização de servidores pode ser aproveitada em uma infraestrutura de *containers* de forma a otimizar ainda mais recursos computacionais. Inclusive esse fato foi observado em pesquisa recente [15], onde os entrevistados foram perguntados sobre qual seria a infraestrutura escolhida para hospedar seus *containers*. Nessa pesquisa 52% dos entrevistados responderam que utilizariam a virtualização de servidores como plataforma para uma infraestrutura de *containers*, 28% responderam que utilizariam diretamente um serviço em nuvem e 20% responderam que utilizariam servidores físicos (*bare metal*).

Diante do relação de trabalhos relacionados exposta, verifica-se que existem poucos estudos na área sistemas de arquivos distribuídos com foco em prover armazenamento com desempenho, confiabilidade e baixo custo para serviços em nuvem, especificamente para o armazenamento de dados de *containers*. Menos estudos ainda são encontrados na área de hiperconvergência que abordem a mesma questão. Entretanto na área de armazenamento definido por software, os problemas relacionados a servidores virtuais é tratado com mais frequência em relação a *containers*, geralmente interceptando e tratando as operações de disco para alguma finalidade específica.

III. Arquitetura Proposta

A arquitetura proposta é apresentada na Figura 1 e funciona resumidamente da seguinte maneira:

1) O servidor físico roda um hipervisor do tipo I com pelo menos duas controladoras de discos: uma para o hipervisor, VSA e *Container Engine* ❶; e outra com os recursos de armazenamento para solução de hiperconvergência ❷.
2) Nos discos conectados na controladora gerenciada pelo hipervisor (caminho ❸) estão armazenados os arquivos do próprio hipervisor, o servidor do virtual do VSA e o servidor virtual do *Container Engine*, conforme podem ser visto nos caminhos ❺.
3) O hipervisor por meio da funcionalidade de "*PCI Passthrough*" repassa o gerenciamento da controladora de discos ❷ para o VSA pelo caminho ❹. Dessa maneira o VSA gerencia diretamente os discos alocados para a solução de hiperconvergência.
4) Cada VSA, com seus próprios recursos de armazenamento, é configurado para participar de um *cluster*

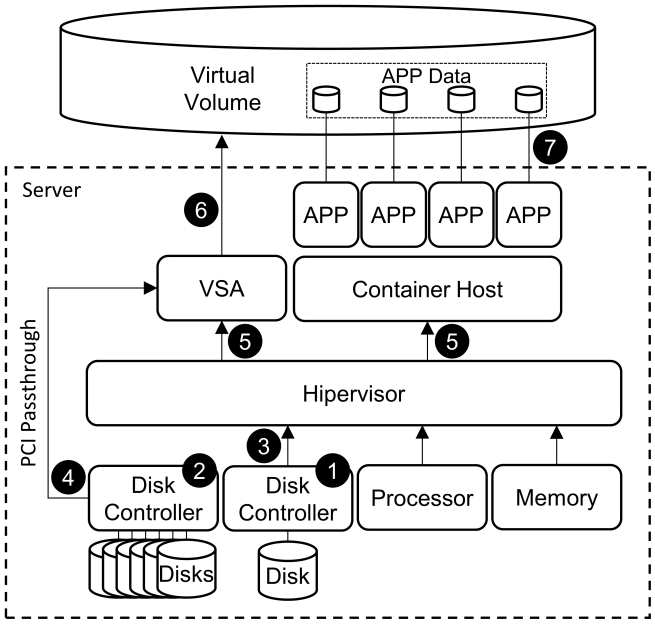

Figura 1. Arquitetura Proposta do Sistema Hiperconvergente.

existente ou iniciar um novo. Cada nó disponibiliza seus recursos individuais ❻ para criar um volume distribuído entre os nós do *cluster*, vide Figura 2. A replicação de dados entre os nós é realizada por meio de uma rede Ethernet de baixa latência e alta taxa de transferência.

5) Após criação do volume virtual, o mesmo é utilizado pelo *Container Host* com uma unidade de armazenamento escalável e tolerante a falhas, conforme pode ser visto no caminho ❼.

O uso de um controlador de disco dedicado ao VSA é uma característica definida para permitir o controle direto dos discos, descartando uma camada de abstração no hipervisor. Outro requisito da arquitetura é uma rede de comunicação com alta taxa de transmissão e baixa latência entre os servidores para evitar o congestionamento ou qualquer tipo de degradação durante as operações de armazenamento. O uso de discos com o mesmo modelo para construir um volume é outro aspecto importante para obter um melhor desempenho [7].

A. Configuração da Arquitetura

Nas próximas subseções são descritas as ferramentas que foram integradas para implementar a arquitetura proposta.

1) GlusterFS: GlusterFS é um SAD de código aberto que permite distribuir um grande conjunto de dados entre diversos servidores. É escalável e suporta receber um grande número de cliente. Possui simplicidade na operação, sendo suportado pela maioria das distribuições Linux. Não faz separação entre metadados/dados e por esse motivo não precisa de servidores adicionais para essa tarefa. É rápido para a maioria das operações pois calcula a localização de metadados usando algoritmos, em vez de recuperar metadados de outro local de armazenamento. Todas as operações no GlusterFS estão sendo feitas no tradutor, que converte solicitações de usuários em solicitações de armazenamento [16].

O GlusterFS possui um mecanismo nativo de redundância baseado na replicação de arquivos entre um conjunto de "*Bricks*" (recursos de armazenamento) que formam uma unidade lógica resiliente [2].

Os *Bricks* são a unidade básica de armazenamento no GlusterFS, representada por um diretório de exportação em um servidor. Uma coleção de blocos forma um "Volume" sendo onde a maioria das operações do GlusterFS acontecem.

O GlusterFS permite o uso de nós heterogêneos com diferentes configurações que pertençam ao mesmo volume distribuído. Nesses casos recomenda-se o agrupamento de nós com mesmas características para o melhor desempenho [7]. O GlusterFS permite agrupar *Bricks* para formar volumes de cinco tipos distintos:

- *Volume Distribuído:* Distribui arquivos entre os *bricks*. A capacidade final do volume é a soma da capacidade dos *bricks*. Não possui redundância. A falha em um *brick* corrompe o volume. Recomendado para casos onde a disponibilidade não é importante ou é provida por outro mecanismo.
- *Volume Replicado:* Os arquivos são replicados entre os *bricks* do volume. Recomendado para casos onde exige-se a alta disponibilidade e confiabilidade.
- *Volume Distribuído Replicado:* Distribui arquivos entre subvolumes replicados. Recomendado para casos onde exige-se escalabilidade, alta disponibilidade e bom desempenho na leitura.
- *Volume Disperso:* São baseados em *erasure codes* e fornecem proteção eficiente contra falhas de discos ou servidores. Um fragmento codificado do arquivo original é armazenado em cada *brick*, de maneira que a partir de um conjunto de fragmentos seja possivel recuperar o arquivo original.
- *Volume Distribuído Disperso:* Distribui arquivos entre subvolumes dispersos. Possui as vantagens de um volume distribuído, porém usando armazenamento disperso dentro do subvolume.

2) ESXi: O VMware ESXi é um hipervisor independente que é executado diretamente no hardware do hospedeiro. Ou seja, é um monitor de máquina virtual (VMM - *Virtual Machine Monitor*) do tipo I (*bare metal*). Ocupa pouco memória do servidor físico e oferece uma camada de virtualização robusta e de bom desempenho, que abstrai os recursos de hardware do servidor e permite o compartilhamento desses recursos entre vários servidores virtuais.

Os recursos de gerenciamento de memória e programação do VMware ESXi fornecem taxas de consolidação e bom desempenho de aplicativo [17].

3) Docker: O Docker começou em 2013 como projeto de PaaS (*Platform as a Service*) da empresa dotCloud para facilitar e integrar a adoção da tecnologia de *containers* [18]. O Docker utiliza as tecnologias de *cgroups* e *namespaces* para isolamento e controle de recursos e permitir o empacotamento de aplicações com todas as suas dependências em uma imagem

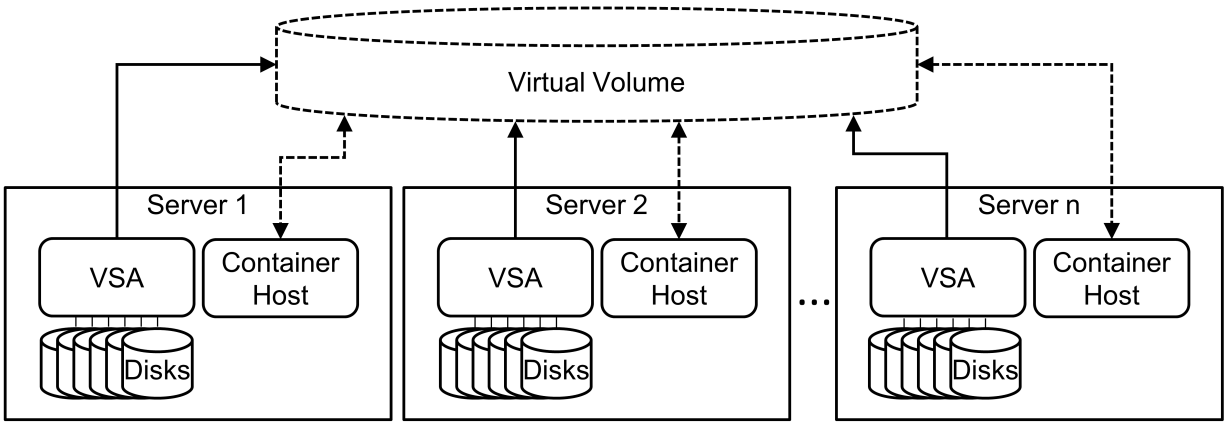

Figura 2. Volume Virtual construído pelos VSAs.

[19]. Essa funcionalidade permite que aplicações diferentes sejam executadas sem conflito no mesmo servidor. Além disso, é possível encontrar, baixar e executar imagens que foram criadas por outras pessoas.

Uma imagem do Docker é uma série de camadas concatenadas, sendo que a primeira é uma imagem base (geralmente composta pelos arquivos básicos de uma distribuição Linux) e as demais camadas são customizações. Essas camadas são gerenciadas de forma eficiente pelo sistema de arquivos utilizado pelo Docker, que permite que as alterações em uma imagem sejam simplesmente uma atualização diferencial da imagem anterior [10]. Essas imagens podem ser armazenadas em repositórios (*registry*) públicos ou privados. Os repositórios públicos do Docker provêm imagens de *containers* com aplicações prontas para executar, tais como bancos de dados, servidores web entre outros [19].

4) VSA: O componente principal da solução é o VSA ("servidor virtual de armazenamento", do inglês *Virtual Storage Appliance*), é um servidor virtual que roda uma versão mínima do sistema operacional Linux e a aplicação GlusterFS. O GlusterFS será utilizado para gerenciar os recursos de armazenamento apresentados ao VSA na forma de um SAD.

A escolha do GlusterFS para o SAD se deve ao fato dos dados e metadados serem distribuídos entre os discos compartilhados entre os nós, sem existir um nó específico para guardar metadados [16]. Essa característica torna o GlusterFS ideal para uso em uma arquitetura hiperconvergente visto que como os nós funcionam sem dependências externas, simplificando a arquitetura e permitindo que a instalação de apenas um VSA por hipervisor.

Os nós do VSA, cada um com seus recursos de armazenamento, permitem a criação de volumes virtuais distribuídos entre diversos servidores que podem ser utilizados pelos hipervisores ou Container Engine para armazenar dados com escalabilidade e tolerância a falhas, conforme é apresentado na Figura 2.

IV. Análise Experimental

A. Plataforma

Para os experimento foi construído um sistema hiperconvergente usando três servidores IBM HS23 com dois processadores Intel Xeon E5-2670 de 2,6 GHz, 96 GB de RAM, um SSD de 16 GB (para hospedar o ESXi, VSA e Docker), dois HDD de 1 TB (para o sistema hiperconvergente), um hipervisor ESXi versão 6.5 e conexão de rede via duas portas Gigabit Ethernet.

O VSA em cada hipervisor foi configurado com 4 vCPU, 8 GB de RAM, CentOS Linux versão 7.5.1804, GlusterFS versão 3.12.15 e o XFS como o sistema de arquivos para discos gerenciados pelo GlusterFS.

O Docker Host foi configurado com 12 vCPUs, 80 GB de RAM, CentOS Linux versão 7.5.1804 e o Docker versão 18.09.

A conexão entre o Docker Host e o volume virtual construído pelos VSAs foi feita através do driver nativo do GlusterFS.

Para avaliar o desempenho de armazenamento de dados do sistema hiperconvergente, foi utilizada a ferramenta de benchmark FIO instalada dentro de *containers* Docker e armazenando dados persistentes no volume virtual hiperconvergente.

B. Carga de Trabalho

Para avaliar o desempenho da proposta foram utilizada duas cargas de trabalho, com diferentes padrões de leitura e escrita, provenientes dos datacenters da Microsoft [20].

As cargas de trabalho foram configuradas para serem executadas dentro de *containers* usando a ferramenta de benchmark FIO, que por sua vez consume recursos de armazenamento e processamento do sistema hiperconvergente proposto. Uma descrição detalhada das cargas de trabalho utilizadas é apresentada na Tabela I.

C. Métricas, Validação e Fatores

As métricas de avaliação definidas nos experimentos foram a taxa de transmissão de dados em megabytes por segundo

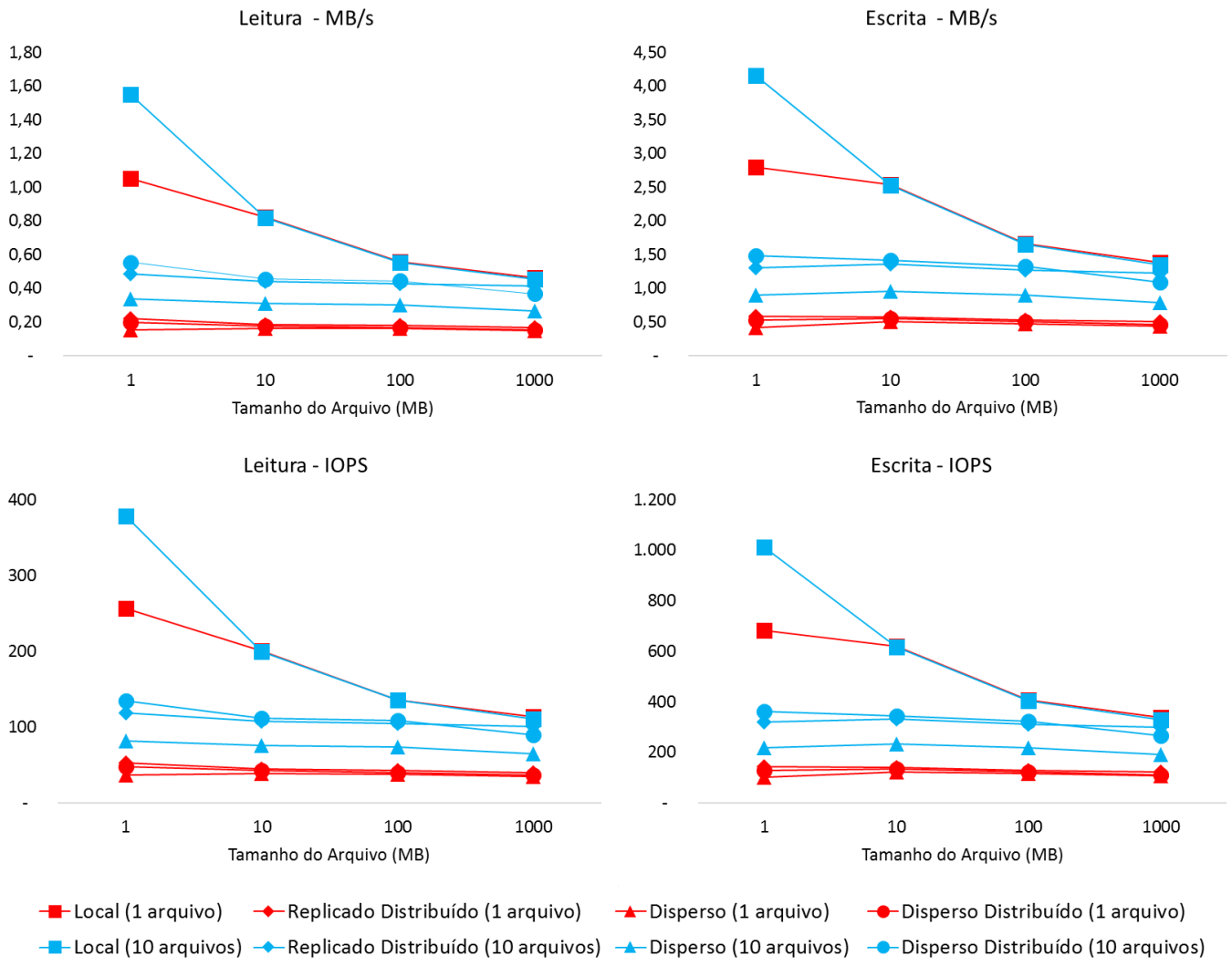

Figura 3. Resultados do Experimento com a Carga de Trabalho Microsoft Cloud Storage (26,2% de leitura e 73,8% de escrita).

Tabela I
CARGAS DE TRABALHO UTILIZADAS PARA AVALIAÇÃO DA ARQUITETURA PROPOSTA.

Carga de Trabalho	Padrão de Leitura/Escrita
Microsoft Cloud Storage	26,2% leitura, 73,8% escrita
Microsoft Web Search	83% leitura, 17% escrita

(MB/s) e a quantidade de operações de entrada/saída por segundo (IOPS).

A validação dos resultados foi realizada por meio da avaliação dos dados obtidos pelo sistema hiperconvergente proposto em comparação com os dados obtidos pelo um sistema de armazenamento tradicional, em que os dados são armazenados diretamente no disco local do servidor.

Durante os experimentos os fatores definidos para avaliação foram: o tipo do volume configurado no GlusterFS; tamanho dos arquivos variando entre 1, 10, 100 e 1000MB e quantidade de arquivos sendo operados simultaneamente, variando entre 1 e 10.

D. Cenários de Avaliação

No sistema hiperconvergente construído para este estudo, o GlusterFS pode ser configurado para criar cinco tipos de volumes: Distribuído, Replicado, Distribuído Replicado, Dispersado e Distribuído Dispersado. Como o volume distribuído não oferece tolerância a falhas e o volume replicado não é escalável, não foram usados nos experimentos esses dois tipos de configurações de disco. Os demais três tipos de volume oferecem a mesma proteção e tolerância a falhas, porém com diferentes tipos de desempenho. Diante disso foram testados esses três tipos de volumes com as cargas de trabalho apresentadas. Esses cenários foram então comparados com um cenário usando o armazenamento local no servidor.

Em cada um dos três tipos de volume um *container* com ferramenta FIO foi iniciado e configurado para executar as cargas de trabalho da Tabela I variando-se o tamanho e quantidade dos arquivos sendo operados, conforme descrito

subseção anterior. Cada *container* rodou a carga de trabalho 100 vezes e média dos resultados foi calculada com um intervalo de confiança de 95%.

E. Experimentos

A seguir são descritos os experimentos realizados para avaliar a arquitetura proposta:

- **Experimento 1 - Microsoft Cloud Storage**: Nesse experimento o armazenamento da arquitetura proposta e tradicional foram analisados com uma carga de trabalho com uma maior quantidade de operações de escrita, operando em um regime de 26,2% de leitura e 73,8% de escrita. Os resultados desse experimento podem ser observados na Figura 3.
- **Experimento 2 - Microsoft Web Search**: Nesse experimento o armazenamento da arquitetura proposta e tradicional foram analisados com uma carga de trabalho com maior quantidade de operações de leitura, operando em um regime de 83% de leitura e 17% de escrita. Os resultados podem ser observados na Figura 4.

Em ambos experimentos os intervalos de confiança calculados foram em média 0,03 MB/s e 7 IOPS e por esse motivo não ficaram visíveis nos gráficos das Figuras 3 e 4.

F. Análise dos Resultados

Analisando os resultados obtidos e consolidados nas Figuras 3 e 4 nota-se que em determinados casos o desempenho da solução proposta é superior ao obtido pela solução convencional de armazenamento local.

No Experimento 1 em que a carga de trabalho realizou mais operações de escrita, nota-se que a solução proposta obteve resultados inferiores ao armazenamento local. Entretanto na solução tradicional observa-se um decaimento exponencial na taxa de transferência e IOPS a medida que o tamanho dos arquivos aumenta na carga de trabalho. Enquanto isso na solução proposta o comportamento é praticamente linear, indicando que a o desempenho é semelhante e independente do tamanho de arquivos sendo operados.

O mesmo comportamento descrito anteriormente pode ser observado na Figura 3 onde a carga de trabalho possuí 26,2% leitura e 73,8% escrita. Com arquivos de 1000MB o desempenho da solução tradicional se aproxima bastante do resultado obtido pelo solução proposta, podendo-se inferir que com arquivos maiores a solução proposta tende a ficar com o desempenho melhor que o tradicional em arquivos maiores que 1GB.

No Experimento 2 em que a carga de trabalho realizou mais operações de leitura, nota-se que a solução proposta obteve resultados superiores ao armazenamento local. Nesse experimento a carga de trabalho possuía 26,2% leitura e 73,8% escrita.

No Experimento 2 a solução proposta obteve desempenho superior ao tradicional ao utilizar as cargas de trabalho que manipulavam 10 arquivos simultaneamente, conforme pode ser observado na Figura 4. Foram observados valores até 48% maiores na taxa de transferência e IOPS (tanto leitura quanto escrita). Esse comportamento indica que a solução proposta possui um desempenho superior ao armazenamento local tradicional ao usar cargas de trabalho com alta leitura e alta manipulação de arquivos.

Observa-se também que a quantidade de arquivos sendo operados em ambas carga de trabalho tem influência no desempenho, vide os que os resultados obtidos pelas cargas de trabalho operando 10 arquivos simultaneamente são superiores aos resultados da mesma carga operando 1 arquivo de cada vez. O tipo de volume virtual configurado no VSA e entregue para o sistema hiperconvergente também tem influencia no desempenho, vide resultados da Figura 4 em que o volume do tipo Replicado Distribuído obteve o melhor desempenho em todas as métricas avaliadas.

Cabe mencionar também que durante a execução dos experimentos não foram observados congestionamentos na rede de comunicação, alto processamento e nem alto consumo de memória, indicando que esses parâmetros não interferiram nos resultados dos experimentos.

V. Viabilidade da Hiperconvergência em Ambientes de Nuvem

Esta seção apresentará uma discussão sobre a viabilidade da hiperconvergência em ambientes de nuvem considerando os resultados apresentados na seção anterior.

Umas das características da hiperconvergência é possibilitar a agilidade de nuvem pública em uma nuvem privada. Um dos dilemas para consumidores de serviços em nuvem é decidir entre mover aplicativos e armazenamento para uma nuvem pública ou mantê-los em uma nuvem privada. Existem muitos parâmetros que precisam ser analisados para tomar esse tipo de decisão, sendo o custo um elemento importante, mas não o único. A agilidade e flexibilidade da nuvem pública são atraentes porque um consumidor pode lançar aplicativos ou implantar um modelo de DevOps para uma liberação mais rápida de recursos e funcionalidades. Entretanto por muitas vezes os consumidores podem demorar nos processos decisórios e administrativos para adquirir hardware e software para integrar os componentes de uma arquitetura tradicional em função dos prazos de entregas e implementação.

A hiperconvergência permite que esses consumidores de serviços em nuvem tenham esse tipo de agilidade em sua nuvem privada. O tempo de instalação e provisionamento de uma plataforma hiperconvergente é mais rápido do que em comparação com uma arquitetura tradicional em razão de basicamente usar servidores comuns em vez de servidores e equipamentos dedicados de armazenamento interligados por uma rede SAN. Nesse caso um provedor de serviços em nuvem pode usar sua infraestrutura hiperconvergente como um *pool* dinâmico de recursos que são usado por muitos consumidores. Cada consumidor pode ter seus próprios recursos para trabalhar e esses podem voltar ao *pool* principal assim que o projeto é concluído. Isso representa uma economia de custos ao evitar as taxas de nuvem pública, permitindo investir em outros ativos.

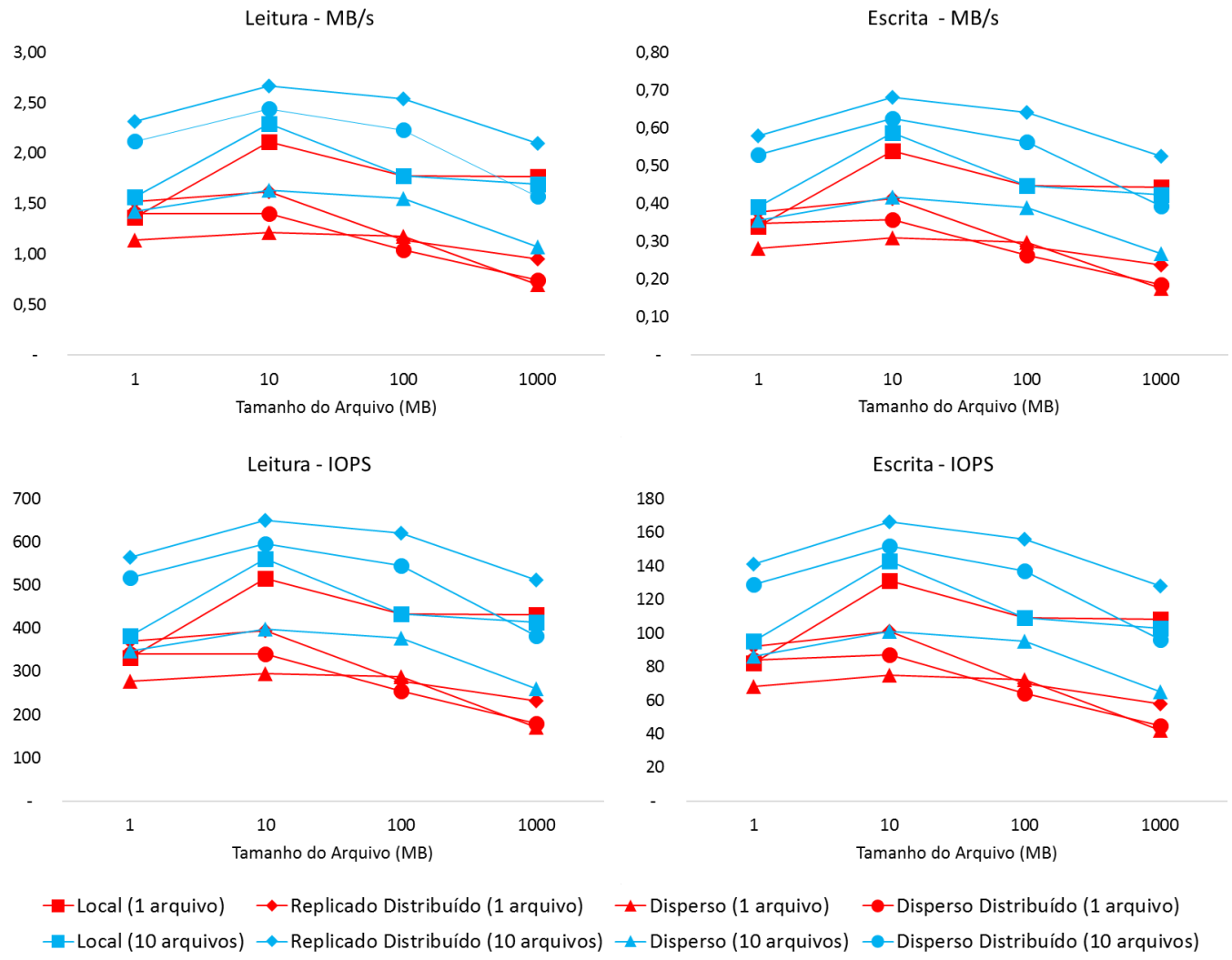

Figura 4. Resultados do Experimento com a Carga de Trabalho Microsoft Web Search (83% de leitura e 17% de escrita).

Outra característica é uma maior disponibilidade com custos menores. Os sistemas hiperconvergentes são arquitetados para alta disponibilidade, o que significa que a arquitetura prevê que componentes falharão e os dados e aplicativos devem ser protegidos. A disponibilidade de sistema hiperconvergente é de muitos para muitos, e não de um para um, como em equipamento dedicado tradicional de armazenamento (*storage arrays*). Nesses equipamentos tradicionais é comum existir duas controladoras para o caso de uma falhar a outra continuar com as operações em disco. Acrescido a essa camada de tolerância a falhas em um equipamento dedicado os volumes de armazenamento são criados agrupando-se discos nas tradicionais configurações de RAID 5 ou RAID 6. No entanto em sistemas que utilizam esse modelo tradicional de armazenamento não existe proteção para falhas ocorridas entre as conexões entre um servidor e o equipamento de armazenamento de dados, exigindo o uso de uma nova camada de proteção nas conexões. Isso significa que para uma alta disponibilidade se faz necessário usar conexões duplas entre servidores e unidades de armazenamento acrescidos de softwares para gerenciar múltiplos caminhos até o volume de dados (*multipath*). Mesmo com todas essas camadas de tolerância a falhas, quando um controladora de um sistema tradicional falha, 100% da carga de operações em disco vai para a outra controladora. Durante uma falha, os aplicativos podem sofrer um grande impacto no desempenho se os controladores individuais estiverem sendo executados com mais de 50% de carga. Basicamente, alcançar alta disponibilidade em projetos de armazenamento com equipamentos dedicados tradicionais é caro. Como um sistema hiperconvergente é uma arquitetura distribuída, os dados são protegidos por padrão, sendo replicados para vários servidores e discos. A proteção é muitos-para-muitos, o que significa que, após uma falha em um servidor, o restante dos servidores absorvem a carga e tem acesso aos mesmos dados. Além disso sistemas hiperconvergentes se reequilibram automaticamente quando um servidor

ou capacidade é adicionado ou removido. Portanto, os sistemas hiperconvergentes são altamente disponíveis e possuem uma melhor estrutura de custos.

Infraestruturas hiperconvergentes também possuem baixo custo para inicio de uma operação. Um sistema hiperconvergente podem começar com dois ou três servidores e crescer para centenas de servidores. Isso elimina a tomada de decisões que provedores de serviços em nuvem devem fazer na escolha de quão grande ou pequeno um equipamento dedicado tradicional de armazenamento precisa ser. Uma estrutura de baixo custo de entrada encoraja consumidores a experimentar novos aplicativos e crescer conforme necessário. As arquitetura tradicionais são comercializadas geralmente em configurações pequenas, médias e grandes, com muita diferença de custo entre diferentes configurações e uma grande parcela do custo sendo alocada para o equipamento dedicado tradicional de armazenamento. Isso não quer dizer que unidades dedicadas de armazenamento não possam ser atualizadas, mas os longos ciclos de atualização e o custo da atualização poderiam influenciar provedores de serviços em nuvem a obter apenas um novo equipamento de maior capacidade. Nessa linha de atuação o provedor investiria em um novo equipamento com capacidade extra que poderia permanecer inativa por anos apenas para evitar os ciclos de atualização.

Em uma arquitetura hiperconvergente, os sistemas são muito mais econômicos por natureza porque são construídos em arquitetura aberta. Além disso, o sistema pode começar pequeno e ser suavemente atualizado adicionando mais servidores. Quando servidores são adicionados, o sistema se reequilibra para aproveitar o armazenamento, a memória e o processamento adicionado.

A custo total de propriedade é uma medida de gastos com investimento e despesas operacionais durante um período de tempo. Normalmente em um provedor de serviços em nuvem esse cálculo é baseado em um período médio de 3 a 5 anos, que é o período tradicional de ciclos de atualização de hardware e software. Os gastos com investimento em um provedor de serviços em nuvem para o ambiente de armazenamento de dados é calculado com base nas compras de hardware para servidores, unidades de armazenamento dedicadas, equipamentos de rede, licenças de software, gastos com energia, instalações e assim por diante. Estudos demonstraram que a economia em gastos de investimento ao usar sistemas hiperconvergentes em vez de sistemas tradicionais está na faixa de 30%, considerando economias em hardware, software e energia [21].

Além disso, a facilidade de implantação e operação e o conhecimento geral existente das tecnologias envolvidas em um ambiente hiperconvergente podem resultar em cerca de 50% de economia em despesas operações [21]. Esses números variam amplamente dependendo de quais produtos são comparados.

Em resumo conclui-se que hiperconvergência executada em sistemas baseados arquiteturas tradicionais são mais econômicos do que as soluções tradicionais que utilizam equipamentos dedicados para armazenamento, agregando os benefícios de ciclos de implementação mais curtos e maior receita gerada pela com eficiência na alocação de recursos para consumidores.

Com a popularização da hiperconvergência, empresas com serviços em nuvem privada estão percebendo que com produtos melhor integrados e melhores ferramentas de automação, é possível duplicar o modelo de nuvem pública para seus consumidores. Dessa forma não é preciso pagar por serviços em nuvens públicas se uma nuvem privada hiperconvergente permite iniciar servidores virtuais ou containers em minutos, alocar armazenamento sob demanda, compartilhar a infraestrutura com vários departamentos, pagar conforme você cresce, automatizar tarefas ou duplicar um modelo em diferentes regiões. Uma nuvem privada hiperconvergente nesse contexto se assemelha com um datacenter comum, porem com uma integração melhores, uso de armazenamento definido por software e um novo modelo de consumo.

O uso de serviços de nuvem pública e de nuvem privada não são mutualmente exclusivos. Em determinados casos é possível utilizar a noção de "nuvens híbridas" o que significa que alguns dos serviços podem ser executados em nuvem privada e alguns outros serviços pode ser executados em uma nuvem pública. Um aplicativo pode ser executado em uma nuvem privada hiperconvergente e seu armazenamento pode estar em uma nuvem pública. Uma empresa pode ter sua própria infraestrutura de hiperconvergência executando serviços de computação, rede e armazenamento, aproveitando recursos de provedores publico tais como Amazon Web Services (AWS), Azure ou Google Cloud Platform. Exemplos de tais serviços são análise ou aprendizado de máquina para alguns dados disponíveis, onde a empresa poderia terceirizar os serviços para a nuvem pública por uma fração do custo de construí-los internamente.

VI. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma análise de desempenho do armazenamento em uma infraestrutura hiperconvergente que usa um sistema de arquivos distribuído de código aberto para armazenar e replicar dados entre vários servidores. Assim também, a infraestrutura permite usar os recursos de computação dos mesmos servidores para hospedar *containers* e armazenar seus dados persistentes.

A proposta foi avaliada de forma experimental em um cenário hiperconvergente com o uso do GlusterFS e Docker, com 2 cargas de trabalho, 3 configurações de disco, 4 tamanhos diferentes de arquivos e 2 quantidades diferentes de arquivos sendo operados simultaneamente. A comparação do cenário foi realizada com uma infraestrutura tradicional de armazenamento local de dados.

Os resultados obtidos pela proposta deste trabalho indica que cargas de trabalho com maior quantidade de operações de escrita e que manipulam arquivos grandes tem melhor desempenho na arquitetura hiperconvergente de que no armazenamento local tradicional.

Um sistema hiperconvergente adiciona recursos desejáveis para sistemas de computação em nuvem quando comparado ao armazenamento convencional, tais como escalabilidade, elasticidade e tolerância a falhas. Um sistema hiperconvergente

pode crescer facilmente adicionando mais servidores, que podem ter diferentes configurações de computação e armazenamento. Enquanto isso em uma arquitetura tradicional, novos servidores individuais podem ser adicionados à infraestrutura, mas os recursos de computação e armazenamento estarão confinados em cada servidor.

Com base nos resultados obtidos nos experimentos pode-se concluir que a proposta de uma infraestrutura hiperconvergente é viável para provedores de serviços de nuvem na prestação de serviços de IaaS (por exemplo, máquinas virtuais e *containers*) e serviços de PaaS (por exemplo, bancos de dados em *containers*). Uma aplicação de usuário que requerer mais recursos de armazenamento do que um único servidor pode oferecer tem desafios para dimensionada em um sistema não convergente, mas teria os recursos alocados facilmente em uma arquitetura hiperconvergente. Esse tipo de recurso é desejado por *datacenters* de provedores em nuvem, pois otimiza a infraestrutura, permitindo que dois tipos de recursos (computação e armazenamento) se colapsem em um, aumentando a eficiência.

Como trabalhos futuro pretende-se avaliar a proposta com clusters maiores, outras soluções de hipervisor, diferentes tipos de discos, configurações de servidor e cargas de trabalho.

REFERÊNCIAS

[1] Z. Liu, F. Dong, J. Zhang, P. Zhou, Z. Xu, and J. Luo, "A client-side directory prefetching mechanism for glusterfs," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 003 942–003 947.

[2] L. M. Roch, T. Aleksiev, R. Murri, and K. K. Baldridge, "Performance analysis of open-source distributed file systems for practical large-scale molecular ab initio, density functional theory, and gw+ bse calculations," *International Journal of Quantum Chemistry*, vol. 118, no. 1, 2018.

[3] T. D. Thanh, S. Mohan, E. Choi, S. Kim, and P. Kim, "A taxonomy and survey on distributed file systems," in *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, vol. 1. IEEE, 2008, pp. 144–149.

[4] J. Blomer, "A survey on distributed file system technology," in *Journal of Physics: Conference Series*, vol. 608. IOP Publishing, 2015, p. 012039.

[5] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive iot data and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.

[6] N. Mills, F. A. Feltus, and W. B. Ligon III, "Maximizing the performance of scientific data transfer by optimizing the interface between parallel file systems and advanced research networks," *Future Generation Computer Systems*, vol. 79, pp. 190–198, 2018.

[7] S. Kaneko, T. Nakamura, H. Kamei, and H. Muraoka, "A guideline for data placement in heterogeneous distributed storage systems," in *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*. IEEE, 2016, pp. 942–945.

[8] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawak, and C. V. Packer, "Beowulf: A parallel workstation for scientific computation," in *Proceedings, International Conference on Parallel Processing*, vol. 95, 1995, pp. 11–14.

[9] V. Tarasov, L. Rupprecht, D. Skourtis, A. Warke, D. Hildebrand, M. Mohamed, N. Mandagere, W. Li, R. Rangaswami, and M. Zhao, "In search of the ideal storage configuration for docker containers," in *Foundations and Applications of Self* Systems (FAS* W), 2017 IEEE 2nd International Workshops on*. IEEE, 2017, pp. 199–206.

[10] Docker, "About storage drivers," https://docs.docker.com/storage/storagedriver/, 2018, acessado em 15/03/2019.

[11] Q. Xu, M. Awasthi, K. T. Malladi, J. Bhimani, J. Yang, and M. Annavaram, "Docker characterization on high performance ssds," in *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 133–134.

[12] ——, "Performance analysis of containerized applications on local and remote storage," in *Proc. of MSST*, 2017.

[13] C.-T. Yang, W.-H. Lien, Y.-C. Shen, and F.-Y. Leu, "Implementation of a software-defined storage service with heterogeneous storage technologies," in *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*. IEEE, 2015, pp. 102–107.

[14] W.-H. Cheng, C.-I. Chiang, C.-T. Yang, S.-T. Chen, and J.-C. Liu, "The implementation of supporting uniform data distribution with software-dened storage service on heterogeneous cloud storage," in *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*. IEEE, 2016, pp. 610–615.

[15] T. Kontzer, "What's holding up the adoption of containers?" http://www.baselinemag.com/storage/slideshows/whats-holding-up-the-adoption-of-containers.html, 2017, acessado em 04/02/2019.

[16] T. Fattahi and R. Azmi, "A new approach for directory management in glusterfs," in *Information and Knowledge Technology (IKT), 2017 9th International Conference on*. IEEE, 2017, pp. 166–174.

[17] M. Veras and A. Carissimi, *Virtualização de Servidores*. Escola Superior de Redes, 2015.

[18] N. Martin, "A brief history of docker containers' overnight success," https://searchservervirtualization.techtarget.com/feature/A-brief-history-of-Docker-Containers-overnight-success, 2015, acessado em 15/03/2019.

[19] Docker, "The definitive guide to docker," https://www.innovate-systems.de/downloads/docker/The_definitive_Guide_to_Docker.pdf, 2018, acessado em 15/03/2019.

[20] J. Huang, A. Badam, L. Caulfield, S. Nath, S. Sengupta, B. Sharma, and M. K. Qureshi, "Flashblox: Achieving both performance isolation and uniform lifetime for virtualized ssds." in *FAST*, 2017, pp. 375–390.

[21] S. Halabi, *Hyperconverged Infrastructure Data Centers: Demystifying HCI*. Cisco Press, 2019.

ANEXO 3

Artigo submetido no JCIT

(Journal of Convergence Information Technology)

Busan / Coréia

Performance Analysis of Storage Scalability in a Hyperconverged System based on Docker and GlusterFS

[1]Rodrigo Leite, [1]Priscila Solis
[1]*Department of Computer Science, University of Brasilia (UnB), Brasília, Brazil,*
rodrigo123@gmail.com, pris@unb.br

Abstract

The adoption of hyperconverged infrastructures is a trend in datacenters, because it merges different type of computing and storage resources. Hyperconverged infrastructures use distributed file systems (DFS) to store and replicate data between multiple servers while using computing resources of the same servers to host virtual machines or containers. In this work, the distributed file system GlusterFS and the hypervisor VMware ESXi are used to build an hyperconverged system to host Docker containers, with the goal of evaluate the storage scalability performance of this system compared to traditional approach where data is stored directly on the server's disks. The performance of the container's persistent storage is evaluated using the benchmark tool FIO (Flexible I/O Tester) under different workloads against several volumes built with different numbers of disks and then compared with a traditional system with local storage.

Keywords*: Hyperconverged Infrastructures, DFS, Containers, GlusterFS, Docker, FIO, Cloud, Storage*

1. Introduction

In the last years, the amount of data generated by different devices and users is growing constantly and exponentially. This brings a challenge to the traditional storage solutions that need to cope with massive data storage and processing. To address this problem, in the last years several large-scale Distributed File Systems (DFS) were developed to overcome these limitations. Some benefits of DFS are scalability, fault-tolerance, parallelism and ability to run on commodity hardware. These systems assemble many nodes across a network and provide a distributed file system with large storage capacity, where data can be transparently accessed [1]. The ability to use commodity hardware makes scalability economically viable, allowing the addition of more devices to scale up the system in an incremental fashion. Because the file system runs on several commodity hardware components, which are highly prone to failure, techniques such as replication and erasure codes are used to avoid data loss and increase fault-tolerance in case of hardware failures.

Traditionally, computing and storage resources in datacenters are separated into different pods. The computing pod usually uses virtualization technologies to optimize resources, while the storage pod is usually based on monolithic hardware. Recently, hyperconverged infrastructures is an emerging paradigm that has been gaining popularity, both in academia and industry [2]. This paradigm allows to merge computing and storage components in on single system using hypervisors and DFS. While traditional storage systems use dedicated hardware to store data, a hyperconverged system uses a DFS and a hypervisor to aggregate the storage capacity and computing resources, providing a unique pod of compute and storage.

In a cloud computing environment, containers can benefit from a hyperconverged infrastructure. The traditional container deployment uses local storage on the node to start and run. Then, the ability to store container's persistent data on a hyperconverged system may improve scalability, elasticity and fault-tolerance in these environments.

Considering the above, the motivation of this work is to evaluate storage scalability performance of a proposed hyperconverged system, based on GlusterFS as the DFS, VMware ESXi as the hypervisor and Docker as container engine. The goal is to evaluate the storage performance scalability by running inside Docker containers several I/O workloads ranging from 100% read to 100% write. The results of the

proposed hyperconverged system are compared with a traditional configuration using local storage, with the intention of verifying the viability of its use in place of direct-attached storage.

This paper is organized as follows: Section 2 presents related works and the literature review; Section 3 presents and describes the proposed architecture; Section 4 details the experiments and results; Section 5 discuss about the use of hyperconverged infrastructures on cloud providers; and finally, Section 6 presents the conclusions and future work of this research.

2. Theoretical Concepts and Related Work

Permanent storage consists of a named set of objects that are explicit created, are immune to temporary failures of the system, and persist until explicitly destroyed. A file system is a refinement that describes a naming structure, a set of operations on objects described by explicit characteristics. DFS allow multiple users who are physically dispersed in a network of autonomous computers share in the use of a common file system. Cloud storage is a system that provides data storage and assembles a large number of different types of storage devices through the application software which are based on the functions of cluster applications, grid techniques and DFS. Cloud storage is a cloud computing system with large capacity storage. An hyperconverged environment combines compute, storage and networking components into a single system or node based on commodity servers, with the aim to reduce hardware costs when compared to specialized storage arrays, servers and other equipment, which appears today as an interesting option to cloud providers.

Studies on performance of hyperconverged environments are still scarce. The fundamental concept behind this architecture is based on the Beowulf Systems, that already been vastly studied by academia. In a Beowulf system a cluster of commodity-grade computers are networked to run some software that can perform parallel computing and storage [3]. A hyperconverged system is a recent evolution of this concept, unifying virtualized servers and virtualized storage onto the same cluster with the goal of simplifying infrastructure.

In one of the first studies specifically about hyperconvergence, the General Parallel File System (GPFS) is presented as a software defined storage solution for hyperconverged systems [4]. In this work, several concepts of hyperconvergence and software defined storage are outlined, and the proposed solution is to use GPFS within a hyperconverged architecture using virtual storage appliances (VSA) that manage physical storage resources. The VSA allow the increase of capacity in a scale-out model and data protection through the “GPFS native RAID” (GNR) that makes copies of the data in several nodes. Although the article is produced by IBM and presents a case of application of its products, several concepts of hyperconvergence and software defined storage are well defined. The architecture presented for the use of GPFS can be replicated for use in another distributed file system in order to achieve the same results.

The amount of data generated by IoT devices is addressed in another study, in which the storage paradigms for store and process data from IoT devices are analyzed. The conclusion points to the hyperconverged architecture as one solution because of its scalability, fault-tolerance and use of commercial off-the-shelf hardware [2].

Hyperconvergence relies heavily on the use of software defined storage (SDS). The terms “Distributed File Systems” and “Software Defined Storage” are similar concepts, sometimes used synonymously. A DFS is by definition a software that stores data across multiple nodes. A SDS system relies on the same definition, emphasizing the ability to manipulate data flow, hardware abstraction, scalability, reliability, fault tolerance, management interface and integration with other software [5].

In the area of DFS and SDS, recent studies evaluated the common problem of how to store and handle large amounts of data. In one of this works [1], there is a comparison between two DFS for storing scientific data. In this study the DFS GlusterFS and Ceph were tested, using 8 nodes distributed in 8 different combinations. The experiments were performed with 7 different I/O patterns. The conclusion was that GlusterFS outperforms Ceph during large I/O workloads and that GlusterFS administration is simplified, requiring less maintenance and installation time. The conclusions about Ceph were that it has greater complexity and needs separate servers for metadata (data about other data), which increases the complexity of the solution.

Another DFS study in a similar area investigates the problem of how to store and transfer large amounts of scientific data between two institutions [6]. This study shows that the best way to transfer data between two research institutions is to use a combination of a high-speed communications network

and a high-performance file system. Finding a balance between these two items enables data to be read from the source at high speed, transmitted quickly, and then recorded at high speed at the destination. The DFS analyzed in this study were BeeGFS, Ceph, GlusterFS and OrangeFS. Experiments were performed with 4 to 8 servers at the source transmitting data via FTP to another 4 to 8 servers at the destination, with BeeFS getting the best performance.

In the work of [7] a guideline for data allocation in DFS is proposed, recommending that data stored in multiple nodes with few disks have a better performance than a few nodes with many disks. In the experiments, volumes built with disks on different servers were compared with volumes built with several disks on the same server. The comparison was made analyzing the read and write throughput while copying files to and from the volumes. In the results it was verified that the proposed guideline obtained better performance in larger multiplicities. The multiplicity is defined by the author as being the number of clients accessing simultaneously a number of volumes. For example, multiplicity 24 means 4 clients simultaneously accessing 6 volumes. The analysis concluded that distributing the volume across several servers results in improved data throughput because of the parallelism in data access. This guideline can be considered a good practice for building DFS. Building volumes with few servers the advantages of a parallel file system are lost and server resources (processor, disk controller, network card, etc.) can impact the performance. Spreading a volume across multiple servers allowed them to be accessed by different servers, thinning resource consumption between each server and improving data throughput.

In the area of containers data storage, there are very few recent studies dedicated to analyze the use of a SDS to store container's persistent data. Recent studies on containers storage performance analyze the possible configurations for storage access [8] [9], how to provide encryption in image manipulation [10] and performance using solid state disks (SSD) [11] [12]. In another works [8] [9] [10] the mechanisms for containers access storage areas are described and analyzed, but both are limited to local storage. In other study the authors analyze the use of remote storage for containers, using non-volatile memory express over fabrics (NVMF) technology being accessed by a remote direct access memory (RDMA) over a 40Gbps Ethernet network [11].

In one of these studies the performance of the different storage drivers used by the container engine Docker were analyzed [8]. Storage drivers are file systems mounted on top of the host file system to store images and containers [13]. In this study a file system benchmark tool was installed inside a container that performed read and write synthetic workloads on each of the storage drivers. The conclusion was that there is no storage driver with good performance in all scenarios analyzed, and the driver must be chosen according to the I/O patterns of the application that will be packaged inside the container. In this paper the issue of persistent storage was not addressed.

In similar study, the performance analysis of different storage drivers was analyzed, but focused on the use of high-performance solid-state storage (SSD) drives installed on the host [9]. In this study both image, container storage and persistent storage were analyzed. In an experimental analysis similar to that used in previous work [8], a file system benchmark tool was also installed inside a container that performed a read and write synthetic workloads on each of the storage drivers on the same SSD. In addition, a persistent storage was presented to a container and the same benchmark tool executed the workloads. The experiment was repeated with different storages drivers, keeping the same persistent storage. The conclusion was that "overlay" and "aufs" storage drivers performed better when using an SSD and that operations in persistent storage are not affected by the storage driver.

In another work involving storing data in containers, the use of high performance solid state storage (SSD) drives are analyzed, both locally and remotely [11]. Different from the previous two works [8] [9], in this study we have an experiment where an external unit to the host is used for store container data, being accessed via a high speed network. In this configuration, the NVMF (Non-Volatile Memory Express over Fabrics) protocol is used to extend the host's PCI-e bus through an Ethernet network. In the experimental analysis several containers with Cassandra database were executed and a synthetic workload was applied to it. The Cassandra data storage location has been configured in different ways to evaluate both local and remote performance. The conclusion was that using this technology, both local and remote storage achieved near performance.

There are few studies in the area of distributed file systems focused on providing performance, reliability and low cost storage for cloud services, specifically for the storage of containers data. Fewer studies are still found in the area of hyperconvergence. However, in the software-defined storage area,

problems related to virtual machine are studied more often than those related to containers, usually addressing the issue of intercepting and manipulating disk operations for some purpose.

Turns out that the most studies are based on local storage, with the exception of one work where the hyperconvergence is used to host and store containers [14]. In this work, the SDS and a hypervisor are used to build an hyperconverged system to host containers, with the goal of evaluate the storage performance of this system compared to traditional approach where data is stored directly on the server's disks. This study is close to our proposal, since the container's data will be stored in a DFS, but the scalability of the proposed architecture were not studied and analyzed.

3. Proposed Architecture

3.1. Architecture

The proposed architecture is presented in Figure 1 and works as follows:

- The server runs a Type I Hypervisor with at least two disk controllers: one for hypervisor, VSA and container host ❶; and one for the hyperconverged system ❷.
- On the disks connected to the disk controller managed by the hypervisor on path ❸ are stored the hypervisor itself and both VSA and the Container Host virtual machines, as can be seen on path ❺.
- Using the "PCI Passthrough" feature on the hypervisor, the second disk controller ❷ is directly managed by the VSA virtual machine via path ❹.
- Each VSA, with its own storage resources, is configured to participate in an existing cluster or start a new one. The nodes provide their individual resources ❻ to create a distributed virtual volume across the nodes of the cluster, as presented on Figure 2. Data replication between VSA nodes is performed over a low-latency, high-bandwidth Ethernet network.
- After the virtual volume is created, it can be accessed by the container engine as a scalable and fault-tolerant data storage unit, as can be seen on path ❼.

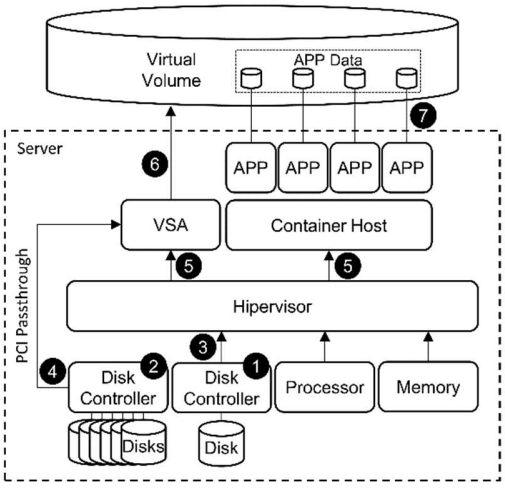

Figure 1. **Proposed hyperconverged architecture.**

The use of a disk controller dedicated to the VSA is a desirable characteristic because it allows direct control of the disks, dropping an abstraction layer in the hypervisor. Another desirable requirement is a high bandwidth and low latency network between the servers to avoid congestion or any kind of degradation during storage operations. The use of disks with same model to build a volume is another important aspect to get a better performance [7].

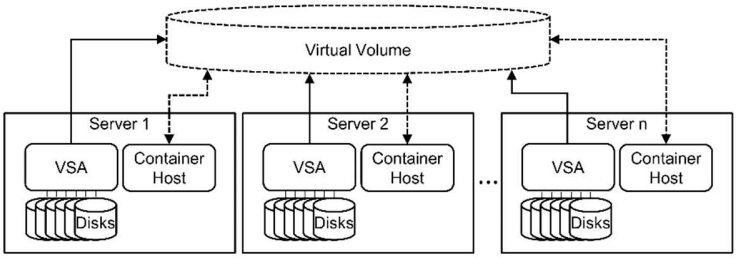

Figure 2. **Virtual volume built by the VSAs.**

3.2. Implementation

In the next subsections, we describe the tools that were integrated to implement the proposed architecture.

3.2.1. GlusterFS

GlusterFS is an open-source DFS that allows to distribute a large set of data across multiple servers. It has simple operation, being supported by most Linux distributions. It does not separate metadata/data and therefore does not need additional servers for this task [15]. It has a native mechanism of redundancy based on the replication of files between a set of "bricks" (a basic unit of storage) that form a resilient logical unit [1]. It allows the use of heterogeneous nodes, enabling nodes with different configurations on the same distributed volume. In these cases, it is recommended to group nodes with the same characteristics for better performance [7].

Bricks are the basic storage unit in GlusterFS, represented by an export directory on a server. A collection of blocks forms a "Volume" in which GlusterFS operations take place. GlusterFS allows you to group Bricks to form five different types of volumes:

1) Distributed Volume: Distribute files between bricks. The final volume capacity is the sum of bricks' capacity. It has no redundancy. Failure on a brick corrupts the volume. Recommended for cases where availability is not important or is provided by another mechanism.
2) Replicated Volume: The files are replicated between the volume bricks. Recommended for cases where high availability and reliability are required.
3) Distributed Replicated Volume: Distributes files between replicated subvolumes. Recommended for cases where scalability, high availability and good reading performance are required.
4) Dispersed Volume: They are based on erasure codes and provide efficient protection against disk or server failures. A coded fragment of 'the original file is stored in each brick, so that from a set of fragments it is possible to retrieve the original file.
5) Distributed Dispersed Volume: Distributes files between dispersed subvolumes. It has the advantages of a distributed volume, but using dispersed storage within the subvolume.

3.2.2. ESXi

VMware ESXi is a hypervisor that runs directly on the host hardware, i.e., it is a virtual machine monitor (VMM) type I (bare metal). It takes minimal memory from the physical server and provides a robust, high-performance virtualization layer that abstracts the server's hardware resources and enables the sharing of these resources across multiple virtual servers.

3.2.3. Docker

Docker is an application that performs operating-system-level virtualization also known as containerization. It uses the resource isolation features of kernel to allow independent containers to run within a single host, avoiding the overhead of starting and maintaining virtual machines (VMs) [9]. Containers and virtual machines have similar resource isolation and allocation benefits but different architectural approaches, which allows containers to be more portable and efficient compared to bare metal and virtual machines [12].

3.2.4. Virtual Storage Appliance

The core component of our proposed architecture is the Virtual Storage Appliance (VSA), a virtual machine that runs a minimal version of the Linux operating system and the GlusterFS software. GlusterFS is configured to manage the physical storage resources presented to the VSA and use these to build a DFS. The choice of GlusterFS for the DFS is due to the fact that data and metadata are stored together in the nodes, without a specific node to handle metadata [15]. This feature makes GlusterFS ideal for use in a hyperconverged architecture since the nodes are independent and can run without external dependencies, simplifying the architecture and allowing the deployment of just one VSA per hypervisor and nothing else. The VSA nodes, each with its own storage resources, allow the creation of DFS volumes that span across many server and can be used by hypervisor and container engine to store data with scalability and fault-tolerance.

4. Experimental Analysis

4.1. Testbed

In our experiments we built a hyperconverged system using three IBM HS23 servers with dual Intel Xeon CPU E5-2670 2.60GHz CPUs, 96GB of RAM, one 16GB SSD (to host ESXi, VSA and Docker Host), two 1TB HDD (to the hyperconverged system), hypervisor ESXi 6.5 and network connection via two gigabit Ethernet ports. The VSA on each hypervisor has 4 vCPU, 8 GB of RAM, runs CentOS Linux release 7.5.1804, GlusterFS 3.12.15 and XFS as the file system for disks managed by GlusterFS. The Container Host has 28 vCPUs, 84 GB of RAM, runs CentOS Linux release 7.5.1804 and Docker 18.09. The connection between the Docker Host and the virtual volume built by the VSAs is made through Gluster Native Client. To evaluate the storage I/O performance of the hyperconverged system we choose FIO 3.1 running in Docker containers and storing persistent data on the hyperconverged virtual volume.

To compare results from the hyperconverged system we built a traditional system where storage resources are provided by local disks on the server. For this system we used another IBM HS23 server with same configuration as the hyperconverged servers, but the persistent storage for the containized FIO were provided by the hypervisor using the local server's disk.

4.2. Workload

To evaluate our proposal, we used the Flexible I/O Tester (FIO), a tool designed to simulate a given I/O workload. We choose a wide array of I/O patterns inspired on the application workloads from Yahoo and Microsoft Datacenters [16], where the I/O patterns vary from 100% read to 100% writes. To evaluate our propose we elaborate 11 different workloads, ranging from only writes to one with only reads, in steps of 10%, as can be seen summarized in Table 1.

Table 1. Workloads.

Workload	I/O Pattern
0/100	100% write
10/90	10% read, 90% write
20/80	20% read, 80% write
30/70	30% read, 70% write
40/60	40% read, 60% write
50/50	50% read, 50% write
60/40	60% read, 40% write
70/30	70% read, 30% write
80/20	80% read, 20% write
90/10	90% read, 10% write
100/0	100% read

4.3. Metrics, Validation and Factors

The metric evaluated during the execution of the experiments the kilobytes per second (KiB/sec). The validation of the results was performed evaluating the results from the proposed hyperconverged system in comparison with the results from a traditional system where data is stored on the local server's disks. For our experiments, the factors we chose to analyze were the number of disk on the volume.

4.4. Evaluation Scenarios and Experiments

In the hyperconverged system built for this study, GlusterFS can be configured to create five types of volumes: Distributed, Replicated, Distributed Replicated, Dispersed and Distributed Dispersed. Since the last four types does not scale in steps of 1 disk, we did not used these types of disk configurations in the experiments, testing only the "Distributed" volume with all workloads.

Inside the hyperconverged system we build 18 different volumes, each one with a unique amount of disks, ranging from 1 to 18, as can be seen summarized in Table 2. Each grey cell in Table 2 represent a disk been in use by the volume.

Table 2. Servers and disks allocation per volume.

	Server 1		Server 2		Server 3		Server 4		Server 5		Server 6		Server 7		Server 8		Server 9	
	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6	Disk 7	Disk 8	Disk 9	Disk 10	Disk 11	Disk 12	Disk 13	Disk 14	Disk 15	Disk 16	Disk 17	Disk 18
Volume 1	■																	
Volume 2	■		■															
Volume 3	■		■		■													
Volume 4	■		■		■		■											
Volume 5	■		■		■		■		■									
Volume 6	■		■		■		■		■		■							
Volume 7	■		■		■		■		■		■		■					
Volume 8	■		■		■		■		■		■		■		■			
Volume 9	■		■		■		■		■		■		■		■		■	
Volume 10	■	■	■		■		■		■		■		■		■		■	
Volume 11	■	■	■	■	■		■		■		■		■		■		■	
Volume 12	■	■	■	■	■	■	■		■		■		■		■		■	
Volume 13	■	■	■	■	■	■	■	■	■		■		■		■		■	
Volume 14	■	■	■	■	■	■	■	■	■	■	■		■		■		■	
Volume 15	■	■	■	■	■	■	■	■	■	■	■	■	■		■		■	
Volume 16	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		■	
Volume 17	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
Volume 18	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

To evaluate the hyperconverged storage I/O performance of each volume we used a containerized FIO to submit the workloads listed on Table 1 inside the container. For comparison purposes, the workloads have also run on a traditional system where the container stores data directly on the server's local disks. The choice for FIO was due to the fact that can run custom workload to measure storage performance and also being popular among recent studies in both academia and industry [17].

In each workload 500 I/O operations were made in 10 files of 100MB in random blocks of 4KB. The mean resulted of the amount of operations made in each workload were calculated with a 95% confidence interval. In the experiments, all the workloads listed on Table 1 ran on the volumes listed on Table 2. The results can be seen on Figure 3a and 3b.

4.5. Analysis of Results

The experimental results shown on Figures 3a and 3b indicates that the proposed architecture can scale without losing performance. As the number of servers and disk increases in a hyperconverged volume, the storage performance increases either. The results also show that the hyperconverged storage architecture has better performance than local storage, getting better throughput after 3-4 disk on the volume.

The overall behavior of the architecture can be seen on Figure 4, where the result's mean of all read and write results were grouped by volume. A regression was made for both read and write graphs, with the exponential regression better fitting the results. The analysis of the regression function confirms that the increase in the number of disk increase the storage performance.

The result's analysis also show that the proposed solution has a better read performance than write as can be seen on Figure 5, where the global behavior of the architecture grouped by workload. The inverted

workload I/O pattern always have a better read performance. This can be explained by the fact that write operations are more complex because the fault-tolerant algorithms require copying files on different disks.

Figure 3a. **Experiments results.**

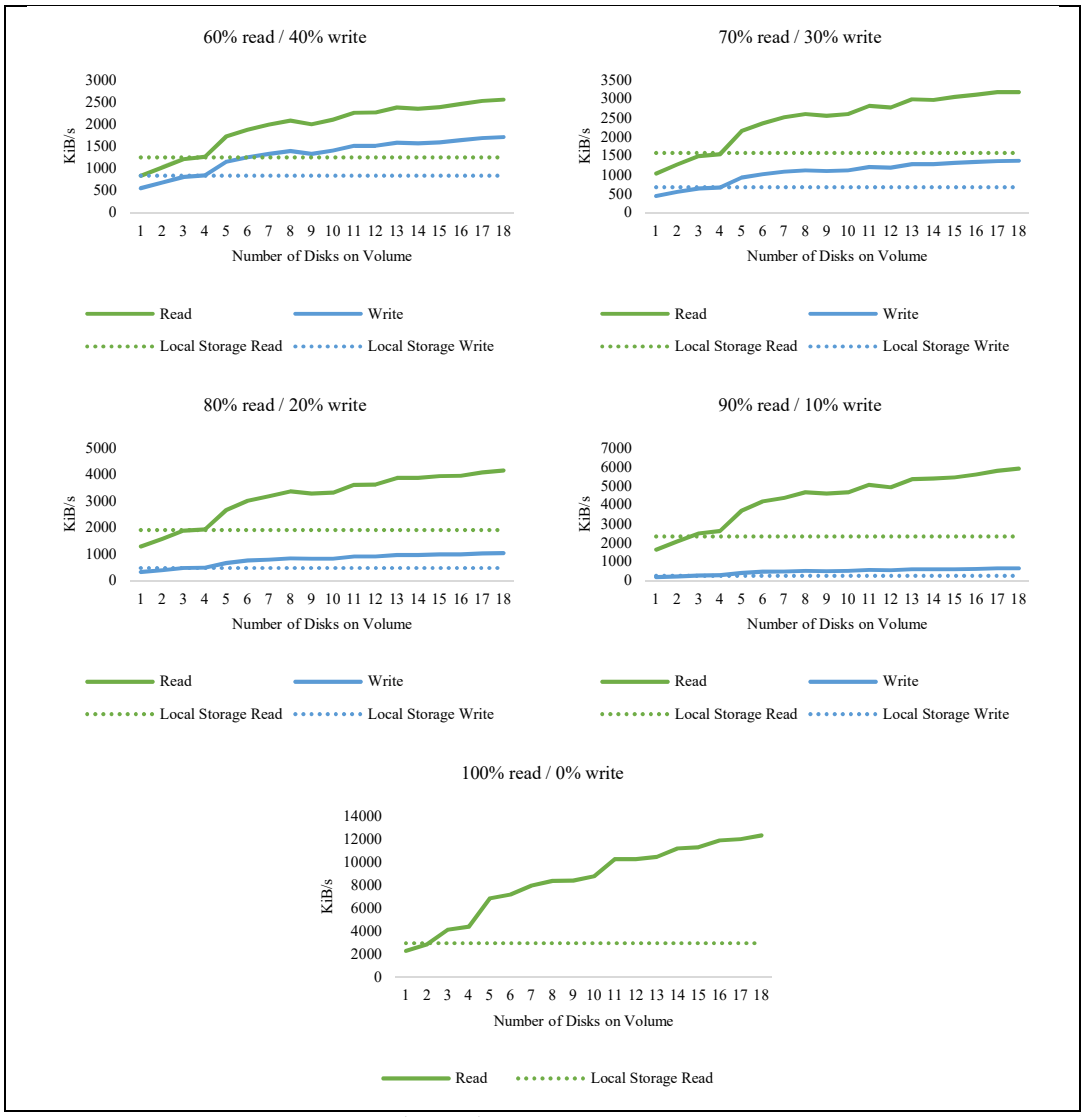

Figure 3b. **Experiments results.**

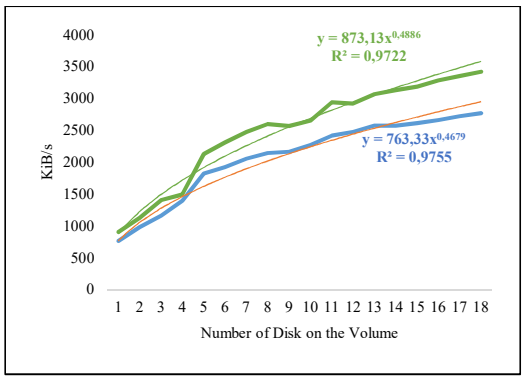

Figure 4. **Results grouped by number of disks on the volume.**

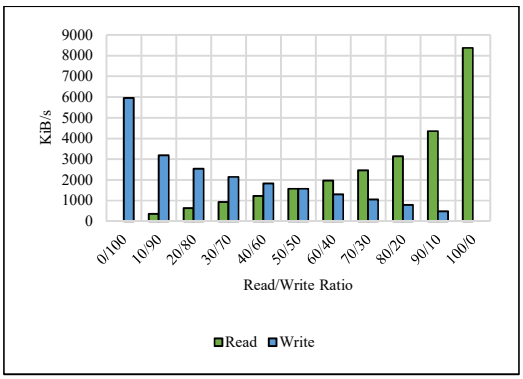

Figure 5. **Results grouped by workload.**

5. Hyperconverged Infrastructures on Cloud Providers

One of the characteristics of hyperconverged is the ability to enable agility in public and private clouds. One of the dilemmas for consumers of cloud services is to decide whether to move applications and storage to a public cloud or keep them in a private cloud. There are many parameters that need to be analyzed to make this kind of decision, cost being an important element, but not the only one. The agility and flexibility of public clouds are attractive because a consumer can develop applications or deploy a DevOps model for faster releases of features and functionality. However, for many consumers need to follow complex processes to acquire hardware and software and integrate the components of a traditional architecture, and the applications delivery time are compromised. Hyperconvergence allows these cloud service consumers to have that kind of agility in their private cloud. The installation time and provisioning of a hyperconverged platform is faster than compared to a traditional architecture because it basically uses "servers" instead of "servers and dedicated storage equipment interconnected by a SAN". In this case a cloud service provider can use its hyperconverged infrastructure as a dynamic pool of resources that are used by many consumers. Each consumer can have their own resources to work with and they can go back to the main pool once the project is completed. This represents cost savings by avoiding public cloud charges, allowing you to invest in others projects.

Another feature is a greater availability with lower costs. Hyperconverged systems are architected for high availability, which means that the architecture predicts that components will fail but data and applications will be protected. The availability on hyperconverged system is "many-to-many," not "one-to-one" as in traditional dedicated storage array. In storage arrays is common to have two controllers, in the case of one fails the other can continue with disk operations. In addition to this fault tolerance layer, storage volumes are created by grouping disks in traditional RAID 5 or RAID 6 configurations. However, in this storage model there is no protection for faults that can occurred between a server and the storage array, requiring the use of another layer of protection on the connections. This means that to have high availability it is necessary to double connections between servers and storage arrays and the use of software to manage multiple paths to the data volume (multipath). Even with all these layers of fault tolerance, when a controller of a storage arrays fails, 100% of the disk operations load goes to the other controller. During this time, applications can suffer a major performance impact if individual controllers are running at more than 50 percent load. Basically, achieving high availability in traditional storage arrays is expensive. Because a hyperconverged system is a distributed architecture, data is protected by default, being replicated to multiple servers and disks. Protection is "many-to-many," which means that after a server crashes, the rest of the servers absorb the load and continue to provide access to the data. In addition, hyperconverged systems automatically rebalance when a server or capacity is added or removed. Therefore, hyperconverged systems are highly available and have a better cost-benefit.

Hyperconverged infrastructures also have low cost to start an operation. A hyperconverged system can start with two or three servers and grow to hundreds. This eliminates the decision-making that cloud service providers must make in choosing how large or small a storage array needs to be. A low cost entry infrastructure encourages consumers to try out new applications and grow as needed. Traditional architectures are usually marketed in small, medium and large configurations, with cost differences between different configurations and a large portion of the cost being allocated to the storage equipment. This is not mean that dedicated storage arrays cannot be upgraded, but long refresh cycles and the cost of upgrading could influence cloud service providers to get only new, higher capacity equipment. In this line of action, providers would invest in a new equipment with extra capacity that could remain inactive for years only to avoid the update cycles. In a hyperconverged architecture, systems cost less by nature because they are built on x86 architecture. In addition, the system can start small and be smoothly updated by adding more servers. When servers are added, the system rebalances to take advantage of added storage, memory, and processing.

A total cost of ownership is a measure of investment spending and operating expenses over a period of time. Typically, in a cloud service provider this calculation is based on an average period of 3 to 5 years, which is the traditional period of hardware and software refresh cycles. Investments in a cloud provider are calculated based on hardware purchases for servers, dedicated storage units, network equipment, software licenses, energy costs, facilities, and so on. Studies have shown that the savings in investment using hyperconverged systems rather than traditional systems is in the range of 30%, considering hardware, software and energy savings [18].

In addition, the ease of deployment and operation and the existing general knowledge of the technologies involved in a hyperconverged environment can result in about 50% savings in operations expenses [18]. This number varies widely depending on which products are compared. In summary, can be concluded that hyperconverged running on x86-based systems are more economical than traditional solutions that use dedicated storage arrays, adding the benefits of shorter deployment cycles and greater revenue generated by efficiently allocating resources to consumers.

With the popularization of hyperconverged infrastructures, companies with private cloud services are realizing that with better integrated products and better automation tools it is possible to duplicate the public cloud model for their consumers. That way you do not have to pay for services in public clouds if a hyperconverged private cloud lets you start virtual servers or containers in minutes, allocate storage on demand, share infrastructure with multiple departments, pay as you grow, automate tasks, or duplicate one model in different regions. A hyperconverged private cloud in this context resembles a common datacenter, but with better integration, use of software-defined storage, and a new consumption model.

The use of public or private cloud services is not mutually exclusive. In some cases, you can use "hybrid clouds" which means that some of the services can run privately and some other services can run in a public cloud. An application can run on a private, hyperconverged cloud and its storage may be in a public cloud. A company can have its own hyperconverged infrastructure running computing, networking, and storage services, leveraging resources from public providers such as Amazon Web Services (AWS), Azure, or the Google Cloud Platform. An example of such services is data mining where one company outsource the services to the public cloud for a fraction of the cost of building them in-house.

6. Conclusions and Future Work

This paper presented a storage performance analysis on the scalability of a hyperconverged infrastructure that uses an open-source DFS to store and replicate data between multiple servers while using the computing resources of the same servers to host containers and store its persistent data. The proposal was evaluated under several different scenarios, using 11 workloads, 18 disk configurations and compared to a traditional direct-attached storage infrastructure.

The results show that the performance of the proposed architecture scales as the number of disks and servers are increased in the hyperconverged volume. Besides that, a hyperconverged system adds another desired features for cloud computing systems when compared to conventional storage, like elasticity and fault-tolerance. A hyperconverged system can grow easily by adding more servers, which may have different computing and storage configurations. Meanwhile in a traditional architecture, new stand-alone servers can be added to the infrastructure, but computing and storage resources are confined to each server. Besides that, a small hyperconverged system can easily bypass the resources of a traditional local storage-based system.

We believe that the use of our proposal of hyperconverged infrastructure is feasible for cloud service providers in the provision of IaaS services (e.g. vitual machines and containers) and PaaS services (e.g. containerized databases and applications in general). A user's application that requires more storage resources than a single server can provide, has challenges for scaling in a non-convergent system but would have the resources allocated easily in a hyperconverged architecture. This type of feature is desired by datacenters from cloud providers because it optimizes infrastructure by allowing two resource pods (computing and storage) to collapse into one, increasing efficiency.

In further research, we intend to evaluate our proposal with bigger clusters, other hypervisor solutions, different type of disks (e.g., SSD), server configurations and workloads.

7. References

[1] L. M. Roch, T. Aleksiev, R. Murri and K. K. Baldridge, "Performance analysis of open-source distributed file systems for practical large-scale molecular ab initio, density functional theory, and GW+ BSE calculations," *International Journal of Quantum Chemistry,* vol. 118, 2018.

[2] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Communications Surveys & Tutorials,* vol. 19, pp. 1457-1477, 2017.

[3] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawak and C. V. Packer, "BEOWULF: A parallel workstation for scientific computation," in *Proceedings, International Conference on Parallel Processing*, 1995.

[4] A. C. Azagury, R. Haas, D. Hildebrand, S. W. Hunter, T. Neville, S. Oehme and A. Shaikh, "GPFS-based implementation of a hyperconverged system for software defined infrastructure," *IBM Journal of Research and Development,* vol. 58, pp. 6-1, 2014.

[5] P. Sharma, "Understanding Software-Defined Storage," The New Stack, 2015. [Online]. Available: https://thenewstack.io/understanding-software-defined-storage/. [Accessed on 06/09/2019].

[6] N. Mills, F. A. Feltus and W. B. Ligon III, "Maximizing the performance of scientific data transfer by optimizing the interface between parallel file systems and advanced research networks," *Future Generation Computer Systems,* vol. 79, pp. 190-198, 2018.

[7] S. Kaneko, T. Nakamura, H. Kamei and H. Muraoka, "A guideline for data placement in heterogeneous distributed storage systems," in *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*, 2016.

[8] V. Tarasov, L. Rupprecht, D. Skourtis, A. Warke, D. Hildebrand, M. Mohamed, N. Mandagere, W. Li, R. Rangaswami and M. Zhao, "In search of the ideal storage configuration for Docker containers," in *Foundations and Applications of Self* Systems (FAS* W), 2017 IEEE 2nd International Workshops on*, 2017.

[9] Q. Xu, M. Awasthi, K. T. Malladi, J. Bhimani, J. Yang and M. Annavaram, "Docker characterization on high performance SSDs," in *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*, 2017.

[10] I. Giannakopoulos, K. Papazafeiropoulos, K. Doka and N. Koziris, "Isolation in Docker through Layer Encryption," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[11] Q. Xu, M. Awasthi, K. T. Malladi, J. Bhimani, J. Yang and M. Annavaram, "Performance analysis of containerized applications on local and remote storage," in *Proc. of MSST*, 2017.

[12] J. Bhimani, J. Yang, Z. Yang, N. Mi, Q. Xu, M. Awasthi, R. Pandurangan and V. Balakrishnan, "Understanding performance of I/O intensive containerized applications for NVMe SSDs," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, 2016.

[13] Docker, "About storage drivers," Docker, 2019. [Online]. Available: https://docs.docker.com/storage/storagedriver/. [Accessed on 06/09/2019].

[14] R. Leite, P. Solis and E. Alchieri, "Performance Analysis of an Hyperconverged Infrastructure using Docker Containers and GlusterFS," in *Proceedings of the 9th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, 2019.

[15] T. Fattahi and R. Azmi, "A new approach for directory management in GlusterFS," in *Information and Knowledge Technology (IKT), 2017 9th International Conference on*, 2017.

[16] J. Huang, A. Badam, L. Caulfield, S. Nath, S. Sengupta, B. Sharma and M. K. Qureshi, "FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs.," in *FAST*, 2017.

[17] J. Axboe, "Flexible I/O Tester," [Online]. Available: https://github.com/axboe/fio. [Accessed on 06/09/2019].

[18] S. Halabi, Hyperconverged Infrastructure Data Centers: Demystifying HCI, Cisco Press, 2019.