PREA: A Replication Strategy for Software Defined Storage

Josino Rodrigues
IFPE , UFPE
jrn4@cin.ufpe.br,
josino@palmares.ifpe.edu.br

Vinicius Garcia
UFPE
vcg@cin.ufpe.br

Rodrigo Elia Assad
UFRPE
assad@deinfo.ufrpe.br

ABSTRACT

The Data Storage industry is facing challenges in the management of heterogeneous storage devices, which in most cases, have their own protocols and proprietary management interfaces. Software Defined Storage(SDS) has emerged as a solution to abstract this heterogeneity. Among the existing requirements in a storage system, availability is a requirement that if not met, prevents the use of the storage solution. For some scenarios, performance requirements can be as important as the availability. In such cases, the search for algorithms and efficient strategies for SDS has entered the agenda of not only professionals but also researchers. This work aims to present the PREA, a strategy for data replication in SDS environments focused on availability and performance. The proposed solution takes into account local replication environments, remote replication and hybrid replication (local and remote). This work is in the development stage and in the course of this article will be presenting the advantages, disadvantages, strengths and weaknesses of the proposed approach.

Keywords

Cloud Storage, Software Defined Storage, SDS, Replication

1. INTRODUÇÃO

Várias das tecnologias de armazenamento utilizadas atualmente no mercado foram projetadas para uma era da computação com necessidades diferentes das atuais. Muitas foram projetadas para uma era onde aplicações e cargas de trabalho sempre executavam em um servidor dedicado. De acordo com um estudo da EMC [9], a previsão é que até 2020 o volume de dados digitais dobre a cada 2 anos, chegando a 40.000 exabytes em 2020. Nesse cenário surge cada vez mais a necessidade de sistemas de armazenamento de dados escaláveis, confiáveis e gerenciáveis.

Com a elevação nos custos de manutenção dos datacenters o software de gerenciamento se torna cada vez mais importante que o hardware subjacente, algoritmos que atendam a requisitos funcionais e não funcionais de armazenamento precisam evoluir e se consolidar para prover soluções de armazenamento eficientes. Software Defined Storage (SDS) vem sendo proposto como uma nova categoria de produtos de software de armazenamento que tem o objetivo de abstrair a complexidade e heterogeneidade no gerenciamento de dispositivos de armazenamento em um datacenter [13].

Em ambientes corporativos, os dados são frequentemente replicados ou espelhados em múltiplos data centers para evitar perda de dados no caso de catástrofes. Outra necessidade latente é o desempenho de armazenamento e acesso aos dados. Alguns fornecedores (como Solid Fire) oferecem soluções totalmente baseadas em discos SSD (Solid-tate Drive), mas essas soluções ainda possuem um alto custo (aproximadamente $1000/TB). O custo de armazenamento nesse caso pode ser ainda maior em ambientes de alta disponibilidade onde os dados são replicados. A partir desse cenário formulamos a seguinte pergunta de pesquisa: *Como garantir disponibilidade de dados e desempenho em sistemas de armazenamento a um baixo custo?*.

Esse trabalho se propõe a apresentar uma possível solução para a pergunta de pesquisa apresentada anteriormente e para isso parte da seguinte hipótese: *A implementação de um SDS com uma estratégia de replicação eficiente, se combinado com um ambiente de armazenamento híbrido (discos FLASH e mecânicos) pode aumentar o desempenho de armazenamento e disponibilidade com um custo reduzido.*

Esse artigo está organizado da seguinte forma: a seção 2 descreve a caracterização do problema, a seção 3 apresenta a fundamentação teórica necessária para o entendimento desse trabalho, a seção 4 apresenta os trabalhos relacionados, a seção 5 apresenta a solução proposta nesse trabalho, a seção 6 apresenta o estado atual desse trabalho e por fim são apresentadas as considerações finais e trabalhos futuros na seção 7.

2. CARACTERIZAÇÃO DO PROBLEMA

Em um cenário onde o software de gerenciamento se torna cada vez mais importante que o hardware subjacente, algoritmos que atendam a requisitos funcionais e não funcionais de armazenamento precisam evoluir e se consolidar. De todos os requisitos existentes em um sistema de armazenamento a disponibilidade é o requisito que, se não atendido, inviabiliza o uso da solução. Para alguns cenários requisitos de desempenho podem ser tão relevantes quanto a disponibilidade. Nesse cenário a busca por algoritmos e estratégias eficientes para SDS tem entrado na agenda de trabalho não só de profissionais mas também de pesquisadores [15, 19, 12].

Uma das vantagens de SDS é permitir que o hardware de

In: Workshop de Teses e Dissertação (WTD), 16., 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.

ISBN: 978-85-7669-332-1

armazenamento subjacente seja heterogêneo. Isso viabiliza a implementação de um *tiered storage* [8]. Um *Multi-tiered storage* usa *tiers* (camadas) de discos SSD e mecânicos para flexibilizar a decisão de prover maior desempenho ou maior capacidade de armazenamento. Nesse caso os dados que são mais comumente acessados ou que exigem maior desempenho são armazenados nos discos SSD e os dados que exigem maior espaço de armazenamento ou que não exigem grandes velocidades de leitura ou escrita são armazenados nos HDs convencionais. O grande desafio em um SDS é tomar essa decisão de forma automática. Esse problema é conhecido como problema de localização de dados [18, 2].

Outro problema também associado a replicação é a otimização do espaço de armazenamento utilizado. Uma das abordagens para tentar resolver esse problema é a utilização de dereplicação. Dereplicação é uma abordagem que elimina réplicas de um dado para otimizar a utilização de espaço de armazenamento, levando em consideração a carga de acesso atual e demendas futuras para o dado em questão. Um dado altamente acessado normalmente não é um candidato a dereplicação. Dereplicação é considerada parte do processo de gerenciamento de recursos, enquanto a replicação é considerada como parte do processo de alocação de recursos[16]. Uma solução de alta disponibilidade deve levar em conta tanto o gerenciamento dos recursos de armazenamento como a alocação desses recursos.

3. FUNDAMENTAÇÃO TEÓRICA

3.1 Software Defined Storage

De acordo com o SNIA, para ser considerado um SDS, uma solução deveria oferecer os seguintes recursos [3]: automação, interfaces padronizadas, virtualização de acesso, escalabilidade e transparência. Embora esses possam ser considerados elementos chave na definição de um SDS, alguns outros recursos ainda são importantes para o bom funcionamento e confiabilidade do serviço de armazenamento. Dentre eles podemos citar [14]:

- **camadas de desempenho dinâmicas:** devido ao surgimento de novas tecnologias de hardware, sistemas de armazenamento geralmente combinam dispositivos de alto desempenho, como discos SSD e FusionIO[1], e dispositivo mais lentos, mas com capacidade de armazenamento maior, como é o caso dos discos SATA. O SDS deve permitir que os dados sejam movidos automaticamente entre as camadas de desempenho para otimizar a velocidade de acesso.
- ***Caching***: normalmente essa é uma funcionalidade importante em grandes sistemas de armazenamento. A diminuição do custo dos discos SSD tem viabilizado a criação de caches mais rápidos no lado do servidor e a criação de arrays de discos híbridos. Mas embora o preço dos discos tenha caído bastante, ainda não é viável para muitas empresas a criação de sistemas de armazenamento totalmente SSD para prover uma melhor desempenho.
- **Replicação**: a replicação é a chave para prover alta disponibilidade e tolerância a falhas em sistemas de armazenamento.
- **Qualidade de Serviço(QoS)**: o alvo da QoS é garantir o atendimento aos requisitos de desempenho de cada aplicação. Por exemplo, o perfil de acesso a disco de um banco de dados é diferente de um sistema de compartilhamento de arquivos.
- ***Snapshots***: snapshots provêem uma cópia do *storage* em um determinado momento, para que se possa reverter o *storage* para esses pontos em caso de alguma falha no sistema.
- **Deduplicação**: embora o custo dos dispositivos de armazenamento esteja caindo, as empresas ainda lutam para alcançar a redução nos seus custos de armazenamento. Quando o usuário armazena o mesmo dado mais de uma vez no sistema de armazenamento, o algoritmo de deduplicação evita que o dado ocupe 2 vezes mais espaço.
- **Compressão**: compressão de dados é outra funcionalidade que visa a redução nos custos de armanzenamento das organizações.

Idealmente, SDS permite aplicações e produtores de dados gerenciar a distribuição dos seus dados através da infraestrutura de armazenamento sem a necessidade de intervenção do administrador do storage, sem operações explícitas de provisionamento, e com gerenciamento automático a nível de serviço[3].

Através da abstração dos recursos físicos, empresas podem adotar diferentes tipos de hardware subjacente. A abstração provida por SDS permite às organizações adotar hardware mais barato para atender suas necessidades. Outras tecnologias tradicionais de armazenamento como RAID e LUNs também permitem uma certa abstração de dados, mas em contrapartida tornam o gerenciamento do storage mais complexo. As soluções SDS se propõem a eliminar ou esconder essa complexidade das equipes que gerenciam o datacenter.

3.2 Replicação de dados

Nem todos os sistemas de armazenamento funcionam da mesma forma. Em geral, eles são divididos *File Storage, Object Storage* e *Block Storage*[14]. Em um *File Storage* o nível de granularidade dos dados é o arquivo. Esse é o tipo de armazenamento mais comum, já que os arquivos são armazenados utilizando uma estrutura de diretórios, no entanto é mais difícil de escalar principalmente quando se trabalha com arquivos muito grandes. Protocolos comumente utilizados nesse tipo de storage são CIFS (Common Internet File System), NFS (Network File System) e HTTP (Hypertext Transfer Protocol).

Em um *Object Storage* o nível de granularidade mais básico é o objeto, que é composto de dados (por exemplo uma imagem, vídeo ou array de bytes), metadados (por exemplo data, tamanho, tipo de câmera que gerou o dado) e um identificador global para cada objeto. Os dados em um *object storage* são normalmente acessados utilizando protocolo HTTP ou APIs REST. Esse tipo de armazenamento é comumente utilizado em provedores de computação em nuvem como IBM SoftLayer, Amazon S3, Google e Facebook.

Já em *Block Storage* os dados são armazenados em sequências de bytes de tamanho fixo conhecidas como blocos. Nesse tipo de armazenamento o sistema operacional do servidor se conecta aos discos físicos através de uma variedade de protocolos como Fibre Channel, SCSI (Small Computer System

[1]http://www.fusionio.com

Interface), iSCSI (Internet protocol SCSI), SAS (Serial Attached SCSI) e ATA (Advanced Technology Attachment).

Independente do tipo de dado e da forma que ele é armazenado, um sistema de armazenamento deveria esta sempre disponível. O mecanismo padrão para atender a esse requisito é a replicação de dados. A replicação é uma técnica para melhorar a qualidade de serviços de armazenamento de dados distribuídos. As motivações para a replicação são: desempenho, aumentar a disponibilidade dos dados e tolerância a falhas.

As técnicas de replicação podem ser classificadas em duas grandes categorias: replicação estática e replicação dinâmica[1]. Nas estratégias de replicação estática o número de réplicas e hosts onde essas réplicas podem ser armazenadas são configurados de forma estática. Já as estratégias de replicação dinâmicas se adaptam a mudanças no padrão de requisições dos usuários, na capacidade de armazenamento corrente e utilização de rede. Essas estratégias podem tomar decisões mais inteligentes de replicação como adicionar ou remover réplicas de um dado de acordo com o perfil de acesso por exemplo.

Em ambientes altamente distribuídos, alta disponibilidade, tempo de resposta, custo de acesso, consumo de banda, confiabilidade e escalabilidade são metricas muito importantes e devem ser levadas em consideração em soluções de replicação.

4. TRABALHOS RELACIONADOS

4.1 Software-Defined Storage

Nos últimos anos várias arquiteturas de SDS têm sido propostas pela academia e pela indústria. Uma dessas propostas é o IOFlow [17], que tem como objetivo otimizar o fluxo de IO em um datacenter para obter ganho de desempenho. IOFlow se baseia nos mesmos princípios de SDN (*Software Defined Networks*) para criar rotas para fluxos de IO existentes no sistema. A ideia é dar um tratamento diferenciado para requisições de IO que exigem melhor desempenho.

Outras propostas tentam implementar o conceito de SDS através do uso de tecnologias existentes como foi feito por um grupo de pesquisa da universidade de Taiwan [21]. Em seu trabalho eles usam OpenStack para construir e gerenciar um serviço de nuvem, e utilizaram o EMC ViPR para integrar diferentes dispositivos de armazenamento para prover um array de discos e construir um *pool* de armazenamento.

Algumas arquiteturas tentam focar em problemas específicos de SDS, como o problema de localização de dados [18], criação de estratégias de experimentação em SDS [5] e implantação de SDS em ambientes de Cloud Storages[11].

4.2 Algoritmos de replicação

Técnicas de replicação vem sendo desenvolvidas e aprimoradas desde o surgimento dos dispositivos de armazenamento. Nos últimos anos estratégias de replicação têm ganhado bem mais relevância devido a larga adoção de sistemas de nuvem. Em [4], os autores apresentam um algoritmo para grids computacionais dinâmicos baseado em popularidade e confiança. Esse algoritmo associa o dado e suas replicas a locais no grid de forma que haja redução na latência de acesso.

Outros trabalhos vem tentando reduzir o consumo de rede através da combinação de estratégias de replicação com algoritmos de deduplicação. Em [20], a deduplicação ocorre antes do envio dos dados para o sistema de armazenamento em nuvem. O Algoritmo deduplicação verifica se já existe uma cópia do dado que se pretende enviar, caso exista, o algoritmo envia apenas os metadados para que haja um apontamento lógico para o dados já existente no sistema de armazenamento.

Como a deduplicação tem sido utilizada para reduzir consumo de banda, a dereplicação tem sido utilizada para reduzir o consumo de espaço em disco. Em [10], um algoritmo de dereplicação é utilizado para realizar o controle das réplicas geradas pelo algoritmo de replicação. Caso exista excesso de réplicas, as réplicas excedentes são removidas sem que haja prejuízo para a disponibilidade do dado.

5. PROPOSTA

PREA é uma estratégia para armazenamento SDS implementado a partir do projeto USTORE [7]. O projeto USTORE consiste em uma solução de baixo custo para armazenamento de arquivos de forma distribuída utilizando redes P2P. Nele, os arquivos são separados em chunks (pedaços com tamanho fixo pré-definido) e gravados em outros nós conectados na rede de acordo o algoritmo de replicação baseado apenas na disponibilidade de cada nó.

A arquitetura do projeto USTORE é composta por cinco componentes básicos:

- ***Superpeers***: esse tipo de host tem o papel de gerenciar os nós conectados na rede P2P e gerenciar as federações existentes na rede P2P.
- **Server peers**: são nós que oferecem serviços necessário para o funcionamento da rede P2P, como autenticação, gerenciamento de arquivos, disponibilidade dos peers, busca de peers, busca de arquivos, etc.
- **Proxies**: atuam como um catalogo de serviços. Guardam informações sobre cada serviço disponível na rede e qual server oferece esse serviço.
- **Bancos de dados relacionais e NoSQL**: armazena os metadados gerenciados por cada server.
- ***SimplePeers***: são responsáveis por armazenar os chunks dos arquivos existentes na rede P2P.

A partir da arquitetura apresentada anteriormente, esse trabalho apresenta como principais contribuições:

- Implementação de um algoritmo que utilize o conceito de multi-tier para garantir o armazenamento inteligente em hardware de maior desempenho ou menor custo, dependendo da demanda existente.
- Implementação de um algoritmo de replicação que leve em consideração o tipo de dado armazenado (bloco, arquivo ou objeto), a camada de desempenho mais adequada e a localização apropriada para garantir disponibilidade e tolerância a falhas.
- Implementação de um algoritmo de dereplicação para otimização do espaço de armazenamento utilizado nos nós

A seguir, cada contribuição da proposta será detalhadamente discutida.

5.1 Tipos de armazenamento

Em nosso trabalho a forma de armazenamento dos dados é identificada através da interface de comunicação utilizada entre o sistema de armazenamento e a aplicação ou cliente que utiliza o storage. Considerando que existem 3 tipos de *storage* (*File Storage, Block Storage* e *Object Storage*), um conjunto de protocolos de comunicação é associado a cada um desses tipos.

Em nossa solução, o armazenamento baseado em arquivos é feito sempre que o usuário acessa o SDS pelo cliente desktop, que atua como software de backup e sincronização de arquivos. Ou através da interface web, onde é possivel visualizar a árvore de diretórios através do browser e recuperar ou armazenar arquivos.

O armazenamento baseado em blocos é utilizado sempre que a aplicação cliente se conecta através do protocolo iSCSI. Esse tipo de armazenamento é bastante apropriado quando se trabalha com arquivo grandes, como por exemplo máquinas virtuais. Nesse cenário, nem sempre é necessário carregar o arquivo inteiro para sua utilização.

Já o armazenamento baseado em objetos ocorre através do uso de uma API REST que possui uma interface de manipulação de objetos baseada em chamadas HTTP. Esse tipo de armazenamento é muito utilizado em aplicações de streamming ou que precisam apenas armazenar um array de bytes com alguns metadados associados a ele.

5.2 Implementação de Data Peers

Uma das característica chave de um SDS é a abstração do hardware subjacente. Em nossa proposta adotamos o conceito de *data peer*, que representa um nó de armazenamento P2P, que pode ser um disco rígido, um disco ssd ou uma máquina conectada na rede. A implementação desse conceito ocorre através de um módulo autônomo de software que representa um espaço de armazenamento no SDS.

Um *data peer* funciona como um processo executando no sistema operacional do host que contém o dispositivo de armazenamento. Dessa forma é possível executar vários *data peers* em um mesmo host, cada um gravando dados em um disco diferente. Além disso é possível utilizar apenas uma parte de um disco para armazenar dados, nesse caso, durante a inicialização do *data peer* é necessário apenas definir a capacidade máxima de armazenamento que o *data peer* irá utilizar do dispositivo de armazenamento.

Sempre que um data peer é inicializado, ele se conecta à rede P2P e anuncia seu espaço disponível para armazenamento. A partir desse ponto um *data peer* pode receber blocos ou chunks oriundos de objetos ou arquivos. Dessa forma a capacidade de armazenamento do SDS pode funcionar de forma elástica, aumentando ou diminuindo de acordo com a necessidade.

5.3 Multi-tier

A criação de *tiers* (camadas) de desempenho parte do princípio de que os dados armazenados no SDS podem ter requisitos de QoS diferentes. Os *tiers* variam em desempenho, custo e capacidade de armazenamento. Cada *tier* possui data peers que armazenam dados em dispositivos de armazenamento com tecnologia específica. Por exemplo, um *tier* de alto desempenho pode ser composto de *data peers* que armazenam dados em placas FusionIO ou discos SSD, que são dispositivos de armazenamento de alta velocidade, mas possuem um custo mais alto.

Cada *data peer* possui um metadado que o associa a uma das camadas existentes no SDS. Essa informação é anunciada na rede P2P no memento da inicialização do *data peer*. A quantidade de tiers do SDS é configurável e novos *tiers* podem ser criados sem a necessidade de reiniciar todo o sistema de armazenamento.

5.4 Estratégia de replicação

O algoritmo de replicação existente no projeto USTORE foi concebido para um cenário P2P no qual os nós da rede podem se conectar e desconectar constantemente. Por esse motivo o principal fator avaliado durante a escolha dos nós para replicação é a disponibilidade [6]. Isso porque, a proposta inicial do projeto era utilizar o recurso ocioso de armazenamento de estações de trabalho corporativas para compor uma nuvem de armazenamento privada. No contexto de SDS, é possivel ter o sistema de armazenamento composto por servidores de grande porte ou racks contendo vários dispositivos de armazenamento. O desafio nesse cenário é criar uma estratégia de replicação que utilize o poder de processamento e armazenamento desse hardware de forma eficiente, sem perder a essência de um sistema de armazenamento P2P. Para resolver esse problema, alguns conceitos novos são adicionados à estratégia de replicação. Para garantir a disponibilidade, consideramos algumas premissas básicas a respeito do hardware utilizado no *storage*: discos podem falhar, controladoras de disco podem falhar, um rack inteiro pode ficar indisponível e um datacenter inteiro pode ficar indisponível.

Partindo desses pricípios definimos o conceito de tipos de localização. A escolha do tipo de localização ajuda a garantir a disponibilidade do dados em caso de falha. Em nossa abordagem definimos cinco tipos de localização:

- **Discos**: os dados são gravados em discos diferentes, garantindo que em caso de falha no disco a informação ainda estará disponível em outro
- **Controladoras**: quando uma controladora falha, todos os discos ligados a ela ficam indisponíveis. Para evitar perda de dados nesse caso pode-se exigir que o algoritmo de replicação crie réplicas em controladoras diferentes.
- ***Hosts***: em um mesmo *rack* é possível existir vários hosts diferentes. Para proteção contra a falha de um host, pode-se exigir que o algoritmo de replicação leve em consideração a replicação entre hosts diferentes.
- ***Rack***: a definição desse tipo de localização permite que o sistema de armazenamento possa se recuperar no caso de falha de um rack inteiro. Para que isso aconteca é necessário que a mesma informação esteja armazenada em racks diferentes.
- ***Site***: em um ambiente altamente distribuídos, é possivel ter um sistema de armazenamento composto de vários datacenters. Assim, em caso de perda de um datacenter inteiro, causado por algum desastre natural por exemplo, pode-se ter o dado disponível caso ele esteja replicado entre datacenters diferentes.

A decisão de qual a melhor localização para um dado depende de qual nível de disponibilidade e desempenho o dado exige. Armazenar réplicas em discos diferentes na mesma

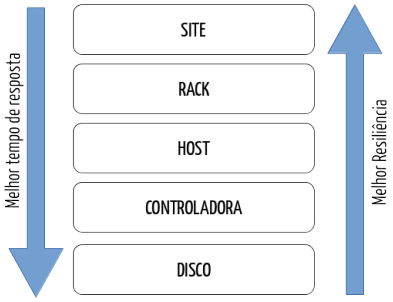

Figure 1: Tempo de Resposta versus Resiliência

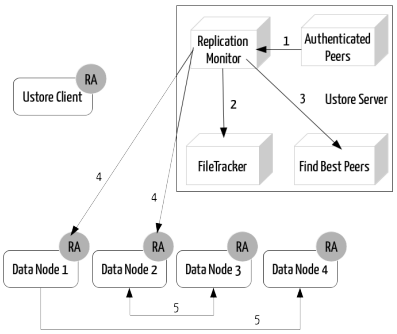

Figure 2: Estratégia de Replicação

controladora ou no mesmo host pode permitir o balanceamento de carga, mas possui um nível de disponibilidade limitado. Quanto mais remoto é o local onde a réplica é armazenada maior a disponibilidade obtida em caso de falha e maior é a latência de acesso ao dado. Esse tradeoff é descrito no diagrama apresentado na Figura 1.

Após a etapa de escolha da localização dos dados a serem replicados, dois componentes básicos da arquitetura entram em ação, o *Replication Agent* e o *Replication Monitor*. Ambos cooperam para realizar a tarefa de replicação na rede P2P. As responsabilidades do *Replication Monitor* são:

- Manter e checar a disponibilidade de um dado
- Descobrir quais dados estão com baixa disponibilidade
- Identificar quais dados devem ser enviados para o data peer escolhido para substituir o data peer indisponível.

Já as responsabilidades do *Replication Agent* são:

- controlar a operação de replicação e a transferência dos dados replicados entre os *data peers*
- checar se todos os dados informados pelo *Replication Monitor* foram copiados.

Conforme mostrado na Figura 2, nossa estratégia de replicação segue os seguintes passos:

1. Assim que o *AuthenticatedPeers* detecta que um peer saiu da rede P2P, ele informa o evento ao *ReplicationMonitor*
2. O *ReplicationMonitor*, por sua vez, tem o objetivo de manter e checar a disponibilidade dos dados existentes nesse peer. Quando um peer sai da rede, o *ReplicationMonitor* pede ao *FileTracker* uma lista com os dados que estavam armazenados no *peer* e quais outros *peers* possuem essa informação. Com essa informação é possível identificar quais dados tiveram a disponibilidade afetada.
3. O *ReplicationMonitor* busca no *FindBestPeers* informações sobre quais os melhores peers para armazenar os dados a serem replicados.
4. O ReplicationMonitor se comunica com o *ReplicationAgent* dos peers que tem a informação a ser replicada e envia uma ordem para iniciar a replicação nos peers informados pelo *FindBestPeers*.
5. Cada *ReplicationAgent* acionado inicia o processo de replicação solicitado.

5.5 Estratégia de dereplicação

Quando nós de armazenamento que ficaram indisponíveis por um determinado intervalo de tempo voltam a funcionar, passa a existir um excesso de réplicas de alguns dados no sistema de armazenamento. Esse excesso de réplicas utiliza espaço de armazenamento que poderia estar sendo utilizado para receber dados novos. Para evitar o desperdício de espaço, o *ReplicationMonitor* decide quais réplicas precisam ser removidas do sistema de armazenamento e ordena ao *ReplicationAgent* dos peers que contém esses dados que removam essas réplicas.

6. ESTADO ATUAL DO TRABALHO

O trabalho encontra-se na fase e implementação e realização de experimentos com o objetivo de avaliar e validar o modelo proposto. A primeira etapa desse trabalho consistiu da revisão de literatura, com o objetivo de entender a área de *Software Defined Storage* na academia e na indústria, bem como fundamentar a proposta do projeto e identificar trabalhos relacionados. A revisão bibliográfica foi conduzida a partir de em uma revisão sistemática da literatura que contou com buscas manuais e automáticas realizadas em quatro fontes de pesquisa (IEEE Explorer, ACM Digital Library, Science Direct e Scopus). Essa revisão sistemática foi utilizada para sintetizar conceitos e tendências da área de pesquisa. Considerou-se também que parte do conhecimento gerado na área de SDS é gerado no âmbito da indústria, por esse motivo também foi realizado uma busca para mapear até que ponto a indústria de armazenamento vem utilizando os conhecimento gerados pela academia.

A partir dos resultados da primeira etapa desse trabalho, iniciou-se a implementação de uma plataforma SDS que será utilizada como base para a realização das outras etapas desse trabalho.

7. CONSIDERAÇÕES FINAIS

Este artigo apresentou o PREA, uma estratégia para replicação de dados em ambientes SDS focada em disponibilidade e desempenho. O projeto se encontra em fase de implementação, e posteriormente será submetido a experimentos que serão realizados em um ambiente real de armazenamento. O objetivo é avaliar o modelo proposto, e comparar os resultados obtidos utilizando benchmarks existentes na literatura

para avaliar o desempenho. Além disso, serão simulados vários cenários de desastre para verificar a disponibilidade dos dados armazenados utilizando a solução proposta.

8. REFERENCES

[1] T. Amjad, M. Sher, and A. Daud. A survey of dynamic replication strategies for improving data availability in data grids. *Future Generation Computer Systems*, 28(2):337–349, 2012.

[2] A. A. Bertossi, D. Diodati, and C. M. Pinotti. Storage placement in path networks. *IEEE Transactions on Computers*, 64(4):1201–1207, 2015.

[3] M. Carlson, A. Yoder, L. Schoeb, D. Deel, C. Pratt, C. Lionetti, and D. Voigt. Software defined storage. *Storage Networking Industry Assoc. working draft, Apr*, 2014.

[4] Z. Cui, Z. Zhang, et al. Based on the correlation of the file dynamic replication strategy in multi-tier data grid. *International Journal of Database Theory and Application*, 8(1):75–86, 2015.

[5] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. Sdstorage: a software defined storage experimental framework. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pages 341–346. IEEE, 2015.

[6] M. P. Duarte, R. E. Assad, F. S. Ferraz, L. P. Ferreira, and S. R. d. L. Meira. An availability algorithm for backup systems using secure p2p platform. In *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, pages 477–481, Aug 2010.

[7] F. Durão, R. Assad, A. Fonseca, J. Fernando, V. Garcia, and F. Trinta. Usto. re: A private cloud storage software system. In *International Conference on Web Engineering*, pages 452–466. Springer, 2013.

[8] A. Elnably, H. Wang, A. Gulati, and P. J. Varman. Efficient qos for multi-tiered storage systems. In *HotStorage*, 2012.

[9] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007:1–16, 2012.

[10] P. Gupta, M. Vardhan, A. Goel, A. Verma, and D. S. Kushwaha. On the fly file dereplication mechanism. In N. Meghanathan, D. Nagamalai, and N. Chaki, editors, *ACITY (1)*, volume 176 of *Advances in Intelligent Systems and Computing*, pages 709–718. Springer, 2012.

[11] M.-J. Huang, C.-F. Huang, and W.-S. E. Chen. Architecting a software-defined storage platform for cloud storage service. In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 379–386. IEEE, 2015.

[12] N. S. Islam, X. Lu, M. Wasi-ur Rahman, D. Shankar, and D. K. Panda. Triple-h: A hybrid approach to accelerate hdfs on hpc clusters with heterogeneous storage architecture. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 101–110. IEEE, 2015.

[13] C. S. Li, B. L. Brech, S. Crowder, D. M. Dias, H. Franke, M. Hogstrom, D. Lindquist, G. Pacifici, S. Pappe, B. Rajaraman, J. Rao, R. P. Ratnaparkhi, R. A. Smith, and M. D. Williams. Software defined environments: An introduction. *IBM Journal of Research and Development*, 58(2/3):1:1–1:11, March 2014.

[14] S. D. Lowe. Software defined storage for dummies. *New Jersey*, pages 15–16, 2014.

[15] S. Matsui, H. Miyoshi, H. Araki, N. Iwasaki, and T. W. Bish. Improving the performance of asynchronous replication with mixed-mode copying. In *Proceedings of the Posters and Demos Session of the 16th International Middleware Conference*, page 8. ACM, 2015.

[16] J. Paiva and L. Rodrigues. On data placement in distributed systems. *ACM SIGOPS Operating Systems Review*, 49(1):126–130, 2015.

[17] E. Thereska, H. Ballani, G. O'Shea, T. Karagiannis, A. Rowstron, T. Talpey, R. Black, and T. Zhu. Ioflow: a software-defined storage architecture. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 182–196. ACM, 2013.

[18] Y. Wu and Y. Zhang. Ga based placement optimization for hybrid distributed storage. In *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pages 198–203. IEEE, 2015.

[19] L. Xiao, K. Ren, Q. Zheng, and G. A. Gibson. Shardfs vs. indexfs: replication vs. caching strategies for distributed metadata management in cloud storage systems. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 236–249. ACM, 2015.

[20] L. Xu, A. Pavlo, S. Sengupta, J. Li, and G. R. Ganger. Reducing replication bandwidth for distributed document databases. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 222–235. ACM, 2015.

[21] 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*, pages 102–107. IEEE, 2015.